UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cristiano Antonio de Souza

Detecção e prevenção de intrusão em computação de nevoeiro e Internet das Coisas

Florianópolis
2023

Cristiano Antonio de Souza

Detecção e prevenção de intrusão em computação de nevoeiro e Internet das Coisas

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de doutor em Ciência da Computação.
Orientador: Prof. Carlos Becker Westphall, Dr.
Coorientador: Prof. Renato Bobsin Machado, Dr.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Souza, Cristiano Antonio de
Detecção e prevenção de intrusão em computação de nevoeiro e Internet das Coisas / Cristiano Antonio de Souza ; orientador, Carlos Becker Westphall, coorientador, Renato Bobsin Machado, 2023.
247 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciência da Computação. 2. Detecção de Intrusão. 3. Internet das Coisas. 4. Aprendizado de Máquina. 5. Computação em nevoeiro. I. Westphall, Carlos Becker . II. Machado, Renato Bobsin. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

Cristiano Antonio de Souza

Detecção e prevenção de intrusão em computação de nevoeiro e Internet das Coisas

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Fabiano Silva, Dr.
Universidade Federal do Paraná (UFPR)

Prof. Marcelo Ricardo Stemmer, Dr.
Universidade Federal de Santa Catarina (UFSC)

Prof. Jean Everson Martina, Dr.
Universidade Federal de Santa Catarina (UFSC)

Prof. Joel José Puga Coelho Rodrigues, Dr.
Universidade Federal do Piauí (UFPI)

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de doutor em Ciência da Computação.

Documento assinado digitalmente
Cristina Meinhardt
Data: 11/01/2024 08:21:22-0300
CPF: ***.850.380-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Márcio Bastos Castro, Dr.
Coordenador do Programa

Documento assinado digitalmente
Carlos Becker Westphall
Data: 09/01/2024 23:01:04-0300
CPF: ***.747.990-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Becker Westphall, Dr.
Orientador

Florianópolis, 2023.

Este trabalho é dedicado aos meus pais.

AGRADECIMENTOS

Agradeço primeiramente a minha família, meus pais Valderi Francisco de Souza e Maria Tereza Rama de Souza, e a meu irmão Cristian Robson de Souza, pelo amor e carinho demonstrado, e pela dedicação em fornecer todas as condições para alcançar meus objetivos, por isso e por muito mais, sou eternamente grato.

Agradeço ao professor, orientador e amigo Carlos Becker Westphall pelas suas orientações, apoio, conselhos e convivência, durante toda a elaboração deste trabalho. Sem o apoio recebido este trabalho não seria possível.

Agradeço ao professor, coorientador e amigo Renato Bobsin Machado pelas suas orientações e seu apoio durante toda minha trajetória acadêmica.

Agradeço a minha amada Mariana Machado Vieira que me acompanhou durante grande parte da elaboração deste trabalho, sempre me incentivando e fornecendo apoio. Obrigado por compartilhar a vida comigo e sempre me ajudar no que for preciso.

Agradeço a todos os amigos que tive o prazer de conhecer em Florianópolis, dentre os quais: Yago, Christian, Artur, Aron, Vinicius, Nelson e Gustavo, por toda companhia e parceria fora da universidade.

Agradeço aos amigos Gustavo dos Santos Vieira, Gustavo Johann e Jean Douglas Valêncio por toda a parceria desde os tempos de Foz do Iguaçu.

Agradeço aos demais pesquisadores e amigos do Laboratório de Redes e Gerência (LRG) da UFSC, Leandro Loffi, Rodolfo Borges, Caciano Machado, Hugo Sampaio, Ricardo Boing por toda a convivência e troca de conhecimento, que proporcionou um ambiente extremamente amigável e favorável para o desenvolvimento deste trabalho.

Agradeço sinceramente ao Programa de Pós-Graduação em Ciência da Computação e à Universidade Federal de Santa Catarina (UFSC). Além disso, este trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC), sob o edital 03/2017 -23038.013359/2017-71, e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

RESUMO

Técnicas especiais de segurança, como mecanismos de detecção de intrusão, são indispensáveis nos sistemas computacionais modernos. No entanto, o contexto da Internet das Coisas apresenta desafios para a aplicação dessas técnicas devido às restrições de recursos envolvidas. É importante detectar e identificar ataques em categorias específicas para aplicar contramedidas adequadas a cada tipo de ameaça. No entanto, as abordagens de detecção multiclasse existentes nesse contexto apresentam alguns pontos fracos, principalmente relacionados aos desafios na identificação de categorias específicas de ataques, ao custo computacional envolvido e a problemas com falsos positivos. Este trabalho aborda esse problema de pesquisa e avança no estado da arte, propondo uma abordagem de detecção multi-etapas chamada DNNET-Ensemble, que combina detecção binária e multiclasse. Nessa abordagem, o tráfego benigno é liberado rapidamente no primeiro nível de detecção, enquanto o tráfego intrusivo é submetido a um método *Ensemble* para análise mais robusta sem causar problemas de latência. Para o primeiro nível de detecção é proposto o método DNNET, com a capacidade de oferecer detecção binária precisa e rápida. Também é apresentada uma estratégia federada para treinar o modelo neural do método DNNET sem enviar dados para a nuvem, preservando assim a privacidade dos dados locais. Além disso, algumas melhorias nos processos de seleção de atributos e no balanceamento de classes são propostas, visando a redução de custos no treinamento dos métodos de detecção. Por fim, é realizada uma breve análise a respeito das abordagens de execução de contramedidas existentes. Os resultados obtidos a partir de experimentos com conjuntos de dados renomados, contendo tráfego de rede intrusivo, demonstram que a abordagem proposta pode alcançar taxas de detecção superiores em comparação com outras abordagens do estado da arte, além de gerar uma baixa taxa de falsos positivos.

Palavras-chave: Internet das Coisas. Computação em nevoeiro. Detecção de Intrusão. Aprendizado de Máquina. Aprendizado Ensemble.

ABSTRACT

Special security techniques, such as intrusion detection mechanisms, are indispensable in modern computing systems. However, the context of the Internet of Things presents challenges for the application of these techniques due to the resource constraints involved. It is important to detect and identify attacks in specific categories to apply countermeasures appropriate to each type of threat. However, existing multi-class detection approaches in this context have some weaknesses, mainly related to challenges in identifying specific categories of attacks, the computational cost involved, and problems with false positives. This work addresses this research problem and advances the state of the art by proposing a multi-step detection architecture called DNNET-Ensemble, which combines binary and multi-class detection. In this approach, benign traffic is quickly released at the first level of detection, while intrusive traffic is subjected to an Ensemble approach for more robust analysis without causing latency issues. For the first level of detection, the DNNET approach is proposed, with the ability to offer accurate and fast binary detection. A federated strategy is also presented to train the neural model of the DNNET method without sending data to the cloud, thus preserving the privacy of local data. Furthermore, some improvements in the attribute selection and class balancing processes are proposed, aiming to reduce costs in training detection methods. Finally, a brief analysis of existing countermeasure execution approaches is carried out. The results obtained from experiments with renowned datasets containing intrusive network traffic demonstrate that the proposed approach can achieve superior detection rates compared to other state-of-the-art approaches, in addition to generating a low false positive rate.

Keywords: Internet of Things. Fog Computing. Intrusion Detection. Machine Learning. Ensemble Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ACC	Acurácia
AE	*AutoEncoder*
ADT	*Alternate Decision Tree*
ANN	*Artificial Neural Networks*
BACC	Balanced Accuracy
CNN	*Convulotional Neural Network*
DNN	*Deep Neural Networks*
DNNET	*Abordagem híbrida proposta com DNN e ET*
DNNET-Ensemble	*Abordagem hierárquica de duas etapas*
DNNKNN	*Abordagem híbrida proposta com DNN e KNN*
DoS	*Denial of Service*
DDoS	*Distributed DoS*
DL	*Deep Learning*
DT	*Decision Tree*
XGBoost	*Extreme Gradient Boosting*
EL	*Ensemble Learning*
ELM	*Extreme learning machine*
ET	*Extra Tree*
FP	Falso Positivo
FN	Falso Negativo
GMM	*Gaussian Mixture Models*
HIDS	*Host-Based Intrusion Detection System*
IA	Inteligência Artificial
IDS	*Intrusion Detection Systems*
IG	*Information Gain*

INB	*Improved Naïve Bayes*
IoT	*Internet of Things*
IPS	*Intrusion Prevention Systens*
KNN	*K-Nearest Neighbors*
LSTM	*Long Short-Term Memory*7
MCC	*Matthews Correlation Coefficient*
MITM	*Man-In-The-Middle*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
MQTT	*Message Queue Telemetry Transport Transport*
NB	*Naïve Bayes*
NIDS	*Network-Based Intrusion Detection System*
NIST	*National Institute of Standards and Technology*
OCC	*One Class Classification*
OS-ELM	*Online Sequential ELM*
PCA	*Principal Component Analysis*
PSO	*Particle Swarm Optimization*
PRE	Precisão
R2L	*Remote to Local*
ReLU	*Rectified Linear Unit*
RF	*Random Forest*
RFE	*Recursive Feature Elimination*
RFID	*Radio-Frequency IDentification*
RNN	*Recurrent Neural Network*
RSL	Revisão Sistemática da Literatura
SAE	*Stacked AutoEncoder*

SDN	*Software Defined Networks*
SFS	*Sequential Feature Selector*
SMO	*Spider-Monkey Optimization*
SMOTE	*Synthetic Minority Oversampling Technique*
SOM	*Self Organizing Map*
SVM	*Suport Vector Machine*
TNR	True Negative Rate
U2R	*User to Root*
VAE	Variational AE
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
WSN	*Wireless Sensor Networks*

SUMÁRIO

1 INTRODUÇÃO

Com o desenvolvimento dos recursos tecnológicos e com a popularização da *Internet*, observou-se um crescimento expressivo do número de aplicações computacionais. Diante deste novo contexto tecnológico, surgiram dificuldades para se manter a segurança de aplicações e dos dados, tendo em vista que as técnicas para explorar vulnerabilidades destas infraestruturas computacionais são constantemente aperfeiçoadas, com o objetivo de adquirir acesso a sistemas, assim como para obter e usar indevidamente informações sensíveis.

As vulnerabilidades dos sistemas computacionais podem ser exploradas por usuários maliciosos para realizar atividades ilícitas. A motivação principal dos invasores é obter conteúdo digital privilegiado que possa trazer algum benefício ao invasor e/ou causar algum dano significativo ao alvo dos ataques.

Atualmente a Internet das Coisas (*Internet of Things* - IoT) está se difundindo em todas as áreas que aplicam recursos computacionais. A IoT possibilita a conexão de objetos do cotidiano à *Internet*, bem como a computadores e *smartphones*. O objetivo é cada vez mais integrar o mundo físico ao digital por meio da comunicação entre objetos, data centers e nuvens.

Os dispositivos de IoT possuem recursos limitados (Atzori; Iera; Morabito, 2010). Portanto, é necessário transferir os dados gerados por esses dispositivos através da *Internet* para processamento e armazenamento em centros computacionais com maior capacidade (Miorandi et al., 2012). Muitas aplicações IoT utilizam a computação em nuvem (*Cloud Computing*) para realizar essas tarefas (Al-Fuqaha et al., 2015). No entanto, com o crescimento da IoT, a geração de grandes quantidades de dados se tornou um desafio significativo. Essa quantidade de dados pode exigir recursos computacionais substanciais, como largura de banda. Além disso, a distância física entre os dispositivos de IoT e os *datacenters* pode resultar em atrasos e congestionamento na rede durante a comunicação (Roman; Lopez; Mambo, 2018). Para superar esses desafios, surgiu o conceito de computação em nevoeiro (*Fog Computing*) (Bonomi et al., 2012; Roman; Lopez; Mambo, 2018). O objetivo é realizar o processamento e armazenamento temporário de dados próximo aos dispositivos de IoT, reduzindo assim o tráfego enviado para a nuvem (Roman; Lopez; Mambo, 2018). Isso permite que as aplicações obtenham uma resposta mais rápida, especialmente aquelas que requerem processamento em tempo real (Al-Fuqaha et al., 2015). Essa abordagem descentralizada permite que as aplicações IoT realizem tarefas de processamento na borda da rede, de forma mais eficiente e com menor latência.

A IoT permite que diversos objetos físicos possam interagir com o ambiente em que estão inseridos, adquirindo a capacidade de ver, ouvir, sentir, pensar e se comunicar entre si (Atzori; Iera; Morabito, 2010). Isso significa que cada objeto pode compartilhar informações com outros objetos e tomar decisões com base nessas informações para executar determinadas tarefas. No entanto, essa conectividade constante e a troca de informações em tempo real trazem consigo desafios de segurança e privacidade. Uma grande quantidade de dados é processada a todo momento nas aplicações IoT, e muitos desses dados são confidenciais e privados, pois em muitas aplicações dados sensíveis dos usuários são coletados, processados, transmitidos e

armazenados (Roman; Lopez; Mambo, 2018). Nesse contexto, os sistemas IoT estão sujeitos a ataques cibernéticos, nos quais os atacantes podem comprometer a segurança do sistema. Um exemplo seria um atacante invadindo o sistema de uma casa inteligente e obtendo acesso a informações sigilosas sobre os hábitos da família, como horários em que os residentes dormem ou quando a casa está vazia (Ni et al., 2018). Essas informações podem ser utilizadas posteriormente para causar danos significativos às vítimas. Portanto, a proteção dos dados sensíveis e a garantia da segurança e privacidade na IoT são desafios cruciais que precisam ser abordados de maneira eficaz para garantir a confiança e a integridade dos sistemas IoT.

As restrições de recursos dos dispositivos IoT os tornam suscetíveis a falhas e ataques mal-intencionados, especialmente no que se refere à integridade dos dados (Neshenko et al., 2019). Isso pode resultar em falta de confiabilidade nos dispositivos e, em casos extremos, até mesmo no colapso do sistema. Um dos principais objetivos de um ataque direcionado a uma rede IoT é interromper a disponibilidade dos dados enviados pelos dispositivos IoT para os aplicativos e serviços. Essa interrupção pode ser alcançada de várias maneiras, como sobrecarregar os dispositivos com solicitações excessivas de informações ou comprometer a estrutura da rede por meio de descarte de pacotes (Roman; Lopez; Mambo, 2018).

1.1 MOTIVAÇÕES

Os ambientes inteligentes estão se tornando reais e possíveis por meio da IoT, porém, conforme supracitado, também não estão livres de ameaças à segurança e das vulnerabilidades. Neste contexto, em paralelo ao crescimento tecnológico, surgem também dificuldades para se manter a segurança das aplicações e infraestruturas computacionais, tendo em vista que conjuntamente com o aumento da quantidade de serviços disponíveis, cresce também as vulnerabilidades. Um incidente importante envolvendo dispositivos IoT ocorreu em outubro de 2016, onde um ataque, envolvendo dispositivos IoT através da *botnet Mirai*, contra o provedor de serviços *Dyn*, deixou *offline* por várias horas, centenas de *sites*, incluindo *Twitter*, *Netflix*, *Reddit* e *GitHub* (Kolias et al., 2017; Tanaka; Yamaguchi, 2017). Essas vulnerabilidades e incidentes destacam a necessidade de técnicas especiais de segurança para os sistemas computacionais modernos. De acordo com Roman, Lopez e Mambo (2018), a segurança é um dos maiores desafios para garantir um ambiente ideal de IoT e de computação em nevoeiro, onde os dispositivos possam usufruir dos serviços fornecidos pelo paradigma. Os Sistemas de Detecção de Intrusão, em inglês *Intrusion Detection Systems* (IDS) são essenciais para a segurança, tendo como objetivo identificar tentativas de ataques.

1.2 PROBLEMAS DE PESQUISA

Através da revisão sistemática da literatura realizada neste trabalho, é possível observar que existem várias abordagens no estado da arte para detectar intrusões em ambiente IoT. Conforme apresentado na Seção 3 (Página 61), alguns trabalhos focam em detecção por as-

sinatura, entretanto, estas abordagens não conseguem detectar novos ataques ou variações de ataques conhecidos (Amaral et al., 2014; Arshad et al., 2019; Zhou; Guo; Deng, 2019). Além disso, métodos baseados em especificação também foram propostos. No entanto, estas abordagens necessitam de especialista humano para especificar o comportamento esperado para a rede. Por fim, outras abordagens propuseram métodos baseados em anomalia para detectar intrusões (Prabavathy; Sundarakantham; Shalinie, 2018; Xu; Qian; Hu, 2019; Miranda et al., 2020). A detecção por anomalia considera que todo comportamento anômalo é uma intrusão e, portanto, consegue detectar novos ataques ou variações dos conhecidos (Boukerche et al., 2007). Os métodos de aprendizado de máquina (*Machine Learning* - ML) são comumente aplicados nesse contexto (Buczak; Guven, 2016). No entanto, abordagens baseadas em anomalia geralmente requerem recursos que os dispositivos IoT não possuem. Desse modo, grande parte dos estudos trabalha com a implantação do módulo de detecção nos dispositivos da computação em nevoeiro.

Os estudos que abordam a detecção por anomalia no nevoeiro podem ser divididos em abordagens de detecção binária e abordagens de detecção multiclasse. Várias abordagens encontradas no estado da arte focaram em métodos baseados em anomalia para detecção binária (ataque ou não-ataque) (Shafi et al., 2018; Miranda et al., 2020; Verma; Ranga, 2020; Kumar; Tripathi; Gupta, 2021; Priyadarshini; Barik, 2022). Estas abordagens são capazes de detectar que uma intrusão está ocorrendo, pois identificaram o tráfego como anômalo, mas não conseguem identificar o tipo ou a categoria do ataque.

Nesse contexto de detecção de intrusão, é importante que a abordagem seja capaz de mitigar a invasão de modo que a mesma não obtenha sucesso (Nobakht; Sivaraman; Boreli, 2016). Portanto, é importante identificar a categoria do ataque para que contramedidas mais específicas possam ser aplicadas para cada tipo de ameaça. A identificação do tipo ou categoria do ataque também é importante para auxiliar na tomada de decisão por parte do responsável pela rede.

No campo das abordagens multiclasse, diversas propostas foram apresentadas no estado da arte para lidar com essa necessidade. No entanto, as abordagens de detecção multiclasse geralmente são mais complexas, possuem maior custo computacional e apresentam taxas de acurácia inferiores em comparação aos métodos binários (Prabavathy; Sundarakantham; Shalinie, 2018; Nguyen et al., 2019; Sarwar et al., 2022; Zhao et al., 2022; Lazzarini; Tianfield; Charissis, 2023). Isto é atribuído principalmente aos desafios na identificação de tipos específicos de ataques (Prabavathy; Sundarakantham; Shalinie, 2018; Diro; Chilamkurti, 2018c; Almiani et al., 2020; Du et al., 2020; Abdel-Basset et al., 2021). Além disso, algumas abordagens apresentam problemas relacionados à identificação normal do tráfego, gerando muitos falsos positivos (Ieracitano et al., 2020; Dat-Thinh; Xuan-Ninh; Kim-Hung, 2022). A métrica de falsos positivos é extremamente importante porque indica quanto do tráfego normal está sendo identificado corretamente. Abordagens com problemas de falsos positivos podem degradar o desempenho da rede, já que a abordagem de detecção estará bloqueando uma grande quantidade de tráfego legítimo.

Diante desse contexto, elaborou-se a seguinte pergunta de pesquisa:

Uma abordagem de detecção de duas etapas distribuídas em uma arquitetura multicamadas, com a combinação de detecção binária e multiclasse, é capaz de alcançar maior robustez de detecção e baixa taxa de falsos positivos em relação aos demais métodos do estado da arte, além de operar sem custos proibitivos?

Essa pergunta de pesquisa se justifica, pois a principal lacuna no estado da arte é a falta de uma abordagem baseada em anomalia para detecção multiclasse que obtenha bom desempenho na identificação de ataques em IoT, mantendo baixa taxa de falsos positivos e sem sobrecarregar o nevoeiro com custos proibitivos. Algumas abordagens apresentam problemas relacionados à alta taxa de falsos positivos (Ieracitano et al., 2020; Moustafa et al., 2021) e dificuldades em manter boas taxas de detecção para os ataques (Vinayakumar; Soman; Poornachandran, 2017; Prabavathy; Sundarakantham; Shalinie, 2018; Sarwar et al., 2022). A taxa de falsos positivos é extremamente importante, pois indica em que medida o tráfego legítimo está sendo identificado incorretamente. Uma alta taxa de falsos positivos é um grande problema e, em alguns casos, pode prejudicar o desempenho da rede.

O custo proibitivo é um ponto bastante relevante a ser considerado. Os ambientes de computação IoT e Fog têm restrições de recursos significativas, o que limita o design de abordagens robustas. Realizar uma análise multiclasse robusta e demorada na fog pode sobrecarregar o dispositivo e causar atraso no fluxo da rede (Nguyen et al., 2019). Além disso, a utilização de técnicas como seleção de atributos e balanceamento de classes, que são úteis para melhorar o desempenho de detecção, tende a aumentar o custo de treinamento. Portanto, é essencial encontrar um equilíbrio entre a eficácia da detecção e os recursos disponíveis nesses ambientes.

Outra questão importante nesse contexto é a implementação de contramedidas para mitigar os ataques detectados. Muitos estudos se concentram apenas em um tipo específico de ataque, o que limita a eficácia das contramedidas propostas (Tu et al., 2018; Priyadarshini; Barik, 2022; Maharaja; Iyer; Ye, 2019; Pacheco et al., 2020). Alguns trabalhos apresentaram uma arquitetura baseada em redes definidas por software (*Software Defined Networks* - SDN), onde utiliza-se o controlador SDN para instalar regras de fluxo apropriadas nos *switches* da borda da rede para controlar a interrupção do tráfego de ataque (Shafi et al., 2018; Nguyen et al., 2019; Rangisetti; Dwivedi; Singh, 2021; Javanmardi et al., 2023). Levando em consideração que a maioria dos trabalhos do estado da arte não apresenta soluções para a implementação de contramedidas para quando o ataque é detectado, este trabalho busca realizar uma breve análise a respeito desse tema para fornecer direções de trabalhos futuras.

1.3 OBJETIVOS

Após identificar as limitações existentes no estado da arte, o objetivo geral definido para este trabalho foi:

Propor uma abordagem hierárquica de duas etapas baseada em computação em nevoeiro e em nuvem, capaz de combinar detecção binária e multiclasse para detecção e prevenção de intrusão em ambientes IoT.

O objetivo geral foi definido com base na hipótese de que a utilização de duas etapas de detecção em uma camada hierárquica pode ser eficaz para aplicar técnicas complexas de aprendizado de máquina sem gerar custos proibitivos em cenários com dispositivos de baixa capacidade, como o ambiente de nevoeiro e IoT. A ideia é que o tráfego legítimo possa ser prontamente liberado na primeira etapa de detecção, enquanto o tráfego intrusivo possa ser submetido a uma análise mais detalhada sem causar atrasos. Além disso, o uso de duas etapas de detecção pode ajudar a recuperar falsos positivos gerados pelo primeiro nível de detecção.

Para alcançar o objetivo deste trabalho os seguintes objetivos específicos foram definidos:

- Realizar uma revisão bibliográfica sistemática da literatura sobre o estado da arte em detecção e prevenção de intrusões em computação de nevoeiro e IoT;
- Propor uma abordagem hierárquica de duas etapas de detecção, combinando métodos binários e multiclasses, em uma arquitetura capaz empregar métodos de *Machine Learning*, abordagens robustas de seleção de atributos e balanceamento de classes, sem gerar custos proibitivos;
- Propor método binário para a etapa de detecção capaz de alcançar taxas de acurácia e *recall* superiores aos demais trabalhos de detecção binária do estado da arte;
- Propor método multiclasse robusto para a etapa de identificação.
- Realizar uma análise a respeito da implementação de ações de contramedidas para mitigar os ataques detectados.
- Realizar avaliação da capacidade de detecção e identificação de intrusões da abordagem proposta, bem como dos custos relacionados a treinamento e operação;

1.4 CONTRIBUIÇÕES

Os objetivos definidos para este trabalho foram alcançados, conforme apresentado ao longo desta tese. As principais contribuições são resumidas a seguir.

Destaca-se, primeiramente, a contribuição por meio da Revisão Sistemática da Literatura (RSL), que proporciona uma visão abrangente do estado da arte atual e oferece orientações de pesquisa para a detecção e prevenção de intrusões em *fog computing* e IoT. Essa revisão permite identificar as limitações e desafios atuais do estado da arte, servindo como base para o desenvolvimento desta pesquisa. A RSL inclui mais de 100 estudos, selecionados após um rigoroso processo de inclusão e exclusão, com critérios bem definidos aplicados a mais de dez

mil artigos pesquisados em sete conceituadas bases de dados de artigos científicos. O processo de execução foi baseado em uma estratégia de pesquisa definida claramente, permitindo a reprodução, replicação e avaliação da integridade do trabalho (Kitchenham; Brereton, 2013). Mais detalhes sobre a RSL são apresentados no Apêndice B.

Este trabalho avança no estado da arte propondo uma abordagem hierárquica de detecção em duas etapas que combina detecção binária e multiclasse para identificar ataques no contexto de computação em nevoeiro e IoT. A abordagem apresenta taxas de precisão superiores em relação a trabalhos anteriores e opera sem gerar custos proibitivos. A arquitetura hierárquica da abordagem é composta por duas camadas: uma camada formada pelos dispositivos do nevoeiro e outra pela nuvem. O primeiro nível da abordagem monitora a rede IoT e realiza uma primeira análise, realizando uma detecção binária do tráfego na camada de nevoeiro. Se for tráfego benigno, será permitido automaticamente. Por outro lado, caso o tráfego seja classificado como intrusivo, ele é enviado para a segunda etapa de análise, na camada de nuvem, onde um classificador multiclasse realiza a identificação da categoria do ataque. Desta forma, informações mais precisas sobre o ataque podem ser obtidas.

A maioria dos eventos será analisada apenas por essa primeira etapa da abordagem, uma vez que serão considerados legítimos e terão seus fluxos liberados. Portanto, é necessário que o método de detecção binária a ser implantado no primeiro nível da abordagem possua alta taxa de precisão e de *recall*, ou seja, que o maior número possível de eventos intrusivos seja detectado como ataque. As redes neurais profundas (*Deep Neural Networks - DNN*) têm apresentado resultados interessantes na detecção de intrusão (Diro; Chilamkurti, 2018c; Ravi; Shalinie, 2020). No entanto, problemas de instabilidade podem ocorrer com classificadores únicos, como dificuldades durante o treinamento e o modelo ficar sobreajustado. Esses fatores tornam desafiador o uso de classificadores únicos no primeiro nível da abordagem proposta, onde busca-se uma baixa taxa de falsos negativos. Outra técnica interessante é o algoritmo *k-Nearest Neighbors* (KNN), que já demonstrou bom desempenho de detecção em experimentos e trabalhos anteriores, embora possa exigir um alto custo computacional. Com o objetivo de melhorar a robustez do método e reduzir o custo associado ao algoritmo KNN, propusemos o método híbrido DNNKNN (Souza et al., 2020), uma abordagem híbrida para detecção binária que combina DNN com o algoritmo KNN. A hipótese foi que um método híbrido conseguiria um desempenho de classificação melhor do que os classificadores únicos. Esta abordagem conseguiu reduzir em aproximadamente 97% o custo do KNN. Apesar desta melhoria alcançada através do método DNNKNN, o custo computacional ainda se apresentava elevado. Como o método binário é projetado para operar nos dispositivos do nevoeiro e irá atuar em um primeiro nível de análise empregado sobre todo o tráfego da rede IoT, é crucial que seja leve para não causar atrasos significativos no tráfego. Assim, a abordagem foi aprimorada substituindo o KNN pelo método *Extra Tree* (ET). Dessa forma, o novo método proposto, chamado DNNET, baseia-se na arquitetura do DNNKNN e combina as técnicas DNN e ET para classificar o tráfego como benigno ou intrusivo. O ET é um método *ensemble* de Árvores de Decisão, em inglês *Decision Tree* (DT). Esta abordagem conseguiu manter o bom desempenho do DNNKNN e reduziu o seu

custo em aproximadamente 36%.

A segunda etapa de detecção opera na nuvem. Nessa etapa, o tráfego previamente identificado como intrusivo será analisado por métodos mais robustos. O objetivo da análise é classificar o tráfego em uma categoria específica de intrusão. Para essa etapa, propõe-se um método *ensemble* multiclasse, que consiste na combinação de três técnicas diferentes de aprendizado de máquina: *Extra Tree*, *Random Forest* (RF) e um modelo neural DNN. Os métodos *ensemble* têm como principal objetivo combinar as classificações de vários classificadores de aprendizado de máquina, para melhorar a generalização e a robustez em relação a classificadores individuais (Goodfellow et al., 2016). Embora os métodos *ensemble* sejam mais robustos, eles tendem a exigir maior capacidade de processamento durante a etapa de detecção e também de treinamento, em comparação com modelos de classificadores individuais. A abordagem proposta contorna essa limitação executando esta abordagem apenas na segunda etapa de detecção, impactando apenas os eventos previamente identificados como intrusivos. Como a primeira etapa já buscou detectar e liberar os fluxos legítimos, o tempo de resposta para os eventos enviados a esta segunda etapa de classificação pode ser flexibilizado, uma vez que há uma alta probabilidade de que sejam intrusivos. Isso possibilita a aplicação de métodos mais robustos para essa classificação.

Este trabalho também fornece contribuições em outras etapas ao longo da arquitetura proposta. As dificuldades de detecção de ataques, em muitos casos, estão relacionadas ao desequilíbrio dos dados existentes. Alguns trabalhos utilizaram a técnica *Synthetic Minority Oversampling Technique* (SMOTE) para balancear os dados (Qaddoura et al., 2021a; Qaddoura et al., 2021b). No entanto, aplicar a estratégia SMOTE completa em cenários extremamente desequilibrados, com muitas classes, criará um número muito grande de registros sintéticos, aumentando o custo do treinamento e gerando o risco de reduzir o desempenho do modelo de aprendizado de máquina. Neste ponto, fornecemos como contribuição a proposta do *Soft-SMOTE*, uma abordagem de balanceamento mais suave, a qual permite operar com um conjunto de dados minimamente balanceado sem gerar um aumento relevante no tempo de treinamento, mesmo em cenários com grande número de classes e grande desequilíbrio entre elas.

A escolha das características do tráfego utilizadas para a análise e detecção é outro ponto importante, alguns trabalhos tentaram filtrar as melhores características de tráfego com técnicas de seleção de atributos *wrapper* (Guerra-Manzanares; Bahsi; Nõmm, 2019), onde métodos de classificação são incorporados ao seletor. Comparativamente, os métodos *wrapper* obtêm conjuntos de atributos de maior qualidade para detecção do que os métodos de *filter*, os quais são baseados em métricas estatísticas e são independentes do classificador (Stiawan et al., 2020). No entanto, as abordagens de *wrapper* exigem mais processamento e geram custos computacionais mais altos, o que pode ser proibitivo ao lidar com grandes quantidades de dados. A estratégia Seleção Híbrida de Atributos proposta pode encontrar um subconjunto ótimo de atributos através de um método *wrapper*, tendo um custo de treinamento menor devido à pré-seleção com um método de filtro (Valencio, 2022).

Este trabalho fornece também uma breve análise a respeito de soluções existentes para

a execução de ações de mitigação de ataques no contexto de computação em nevoeiro e IoT. São apresentadas algumas estratégias de mitigação e as suas limitações. A análise tem como foco principal os estudos baseados em Redes Definidas por Software (*Software Defined Networks*), os quais se mostraram promissores nesse contexto. A partir disso, são discutidas direções de pesquisas futuras nesse tema.

Os principais componentes do sistema foram avaliados em termos de desempenho de detecção e custo de tempo por meio de experimentos quantitativos. Esses experimentos consideraram o cenário de detecção binária para validar a abordagem DNNET e também o cenário multiclasse para validar a abordagem completa de dois níveis de detecção. Os resultados obtidos nos experimentos com diversos conjuntos de dados de intrusão demonstraram que a abordagem alcançou um desempenho superior em relação a outras técnicas clássicas de aprendizado de máquina e abordagens do estado da arte. A abordagem proposta obteve médias de acurácia, precisão e taxas de *recall* balanceado superiores às demais abordagens. Além disso, cabe destacar que também alcançou um desempenho superior em relação à identificação de tráfego benigno, indicando uma baixa taxa de falsos positivos e exigindo menor custo computacional.

Parte das contribuições desta tese foram publicadas em artigos científicos em periódicos e eventos referenciados a seguir:

- Souza, C. A. et al. *Hybrid approach to intrusion detection in fog-based IoT environments*. *Computer Networks*, p. 107417, 2020.

- Souza, C. A.; Cardoso, J. V.; Westphall, C. B. *Multiclass decomposition and Artificial Neural Networks for intrusion detection and identification in Internet of Things environments*. In: Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. SBC, 2021. p. 85-98.

- Souza, C. A. et al. *Two-step ensemble approach for intrusion detection and identification in iot and fog computing environments*. *Computers Electrical Engineering*, v. 98, p. 107694, 2022.

- Souza, C. A. et al. *Intrusion detection and prevention in fog based iot environments: A systematic literature review*. *Computer Networks*, p. 109154, 2022.

- Souza, C. A. et al. *DNNET-Ensemble approach to detecting and identifying attacks in IoT environments*. In: Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. SBC, 2023. p. 435-448.

- Souza, C. A.; Westphall, C. B.; Machado, R. B. *Intrusion detection with Machine Learning in Internet of Things and Fog Computing: problems, solutions and research*. Sociedade Brasileira de Computação, 2023.

1.5 ORGANIZAÇÃO DA TESE

O restante da tese está organizado conforme descrito a seguir. O Capítulo 2 apresenta os conceitos fundamentais envolvidos na temática deste trabalho. São apresentados conceitos a respeito das camadas envolvidas em uma arquitetura IoT-Fog-Cloud e também são discutidas as ameaças existentes nesses ambientes. Posteriormente, são apresentados os Sistemas de Detecção de Intrusão e as suas principais categorias. Por fim, são abordados os principais conceitos relacionados a Aprendizado de Máquina para detecção de intrusão.

No Capítulo 3 (Página 61), é exposto o estado da arte levantado por meio de uma revisão sistemática da literatura, e são apresentadas as abordagens propostas pelos principais trabalhos relacionados. Por fim, nesse capítulo são discutidos os principais problemas de pesquisa.

O Capítulo 4 (Página 79) apresenta a proposta de solução completa para detecção de intrusão em ambientes de computação em nevoeiro e IoT. Inicialmente é apresentada a proposta da abordagem de detecção binária DNNET. Em seguida, são apresentados detalhes a respeito de uma abordagem *ensemble* multiclasse. Posteriormente, é apresentado como ambas as abordagens operam em um módulo de duas etapas para detecção e identificação de intrusão. Também são apresentados detalhes a respeito do treinamento dessa solução completa. Por fim, é realizada uma breve análise a respeito das ações de mitigação de intrusão.

No Capítulo 5 (Página 109), é descrita a metodologia de avaliação executada. São apresentados detalhes a respeitos dos conjunto de dados e métricas de desempenho utilizadas nos experimentos de simulação realizados. Todos os experimentos realizados são descritos em detalhes nesse capítulo. Primeiramente, é realizado um experimento para avaliação da abordagem DNNKNN e DNNET em relação a abordagens binárias do estado da arte. Posteriormente, é realizado um experimento para avaliar a abordagem multiclasse proposta de duas etapas em relação a métodos clássicos e abordagens do estado da arte. Por fim, no último experimento a estratégia de treinamento federado é avaliado em relação a abordagem centralizada clássica.

Os resultados obtidos por meio dos experimentos de simulação são exibidos e discutidos no Capítulo 6 (Página 131). Os resultados são apresentados em três seções, uma para cada experimento. Este capítulo também apresenta um seção de discussões a respeito dos experimentos realizados.

Por fim, o Capítulo 7 conclui a tese, apresentando considerações finais, as limitações existentes no trabalho e as direções para trabalhos futuros.

2 BACKGROUND

Neste capítulo são abordados conceitos relacionados a temática deste trabalho, contextualizando os sistemas de detecção de intrusão (*Intrusion Detection System* - IDS), a *Internet of Things* (IoT), a computação em nuvem (*Cloud Computing*) e a computação em nevoeiro (*Fog Computing*). Na Seção 2.4 (Página 41) são discutidas as principais ameaças presentes nos ambientes IoT. Por fim, a Seção 2.6 (Página 46) apresenta os conceitos básicos de *Machine Learning* (ML) e uma breve descrição da sua aplicabilidade para a detecção de intrusão por anomalia.

2.1 INTERNET OF THINGS

A *Internet of Things* (IoT) tem como característica básica a presença pervasiva de uma grande variedade de objetos inteligentes no cotidiano das pessoas, como: sensores, *tags* de *Radio-Frequency IDentification* (RFID), telefones móveis, dentre outros (Atzori; Iera; Morabito, 2010). A IoT conecta dispositivos físicos à *Internet*, permitindo a comunicação e atuação de maneira inteligente.

Do ponto de vista conceitual a IoT baseia-se em três princípios básicos relacionados as características dos objetos inteligentes: serem identificáveis, comunicáveis e capazes de interagir com o ambiente em que estão inseridos (Miorandi et al., 2012). A IoT permite que vários objetos físicos possam ver, ouvir, sentir, pensar e se comunicar de modo a compartilharem informações entre si e tomarem decisões para realizarem determinadas tarefas.

As aplicações IoT possuem potencial para melhorar a vida das pessoas, a maneira como vivem, trabalham, aprendem e se divertem. Por exemplo, as casas inteligentes podem fornecer aos moradores abertura automática da garagem, preparação automática do café, controle dos sistemas de climatização, dentre outros.

Os dispositivos IoT são objetos físicos que geralmente possuem tamanho pequeno, com capacidades limitadas de processamento e armazenamento. Devido a grande quantidade de dados gerados por esses dispositivos, surgiu a necessidade de maior capacidade computacional. Além disso, a quantidade de dispositivos conectados a *Internet* segue crescendo. A *Cisco* prevê que a quantidade de dispositivos interconectados no planeta poderá alcançar a marca de 500 bilhões até 2025 (Camhi, 2015). A computação em nuvem pode ser uma solução para resolver estas necessidades de maior capacidade de processamento.

2.2 CLOUD COMPUTING

O *National Institute of Standards and Technology* (NIST) define a Computação em Nuvem (*Cloud Computing*), como um modelo para permitir acesso de rede onipresente e conveniente a um conjunto compartilhado de recursos computacionais configuráveis que podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação entre prestadores de serviços (Mell; Grance et al., 2011).

Segundo os autores Takabi, Joshi e Ahn (2010), a *Cloud Computing* é um paradigma importante, com o potencial de reduzir significativamente os custos através de otimização e aumento das eficiências operacionais e econômicas. Além disso, destacam que este paradigma pode melhorar significativamente a colaboração, a agilidade e a escalabilidade, possibilitando um modelo de computação verdadeiramente global sobre a infraestrutura da *Internet*.

A Computação em Nuvem possui 5 características essenciais: o autosserviço sob demanda, acesso de rede onipresente, *pooling* de recursos, independência de localidade, elasticidade rápida e serviço medido, todos voltados para o uso de nuvens de forma transparente. Na *cloud* os recursos computacionais do provedor são agrupados para atender a vários consumidores usando um modelo de multi-locação, com diferentes componentes físicos e virtuais atribuídos e reatribuídos dinamicamente de acordo com a demanda do consumidor. Os provedores devem fornecer elasticidade rápida, permitindo ao consumidor aumentar ou diminuir os recursos, e atendimento sob demanda, para que o cliente possa alocar unilateralmente recursos de forma dinâmica (Takabi; Joshi; Ahn, 2010). Os provedores de *cloud* são capazes de lidar com grandes quantidades de dados e altas taxas de processamento.

As características básicas da *cloud computing* a tornam um importante mecanismo de processamento para aplicações IoT que realizam a captura de grandes quantidades de informações. No entanto, a sua utilização também possui desvantagens, pois esta centralização dos recursos de processamento e armazenamento implica em uma grande separação entre os dispositivos físicos de IoT e os *datacenters* da *cloud*. Fato este, que de acordo com Satyanarayanan (2015), resulta no crescimento da latência média e no *jitter*.

Surgiu então a Computação em Nevoeiro (*Fog Computing*), capaz de resolver os problemas supracitados para aplicações IoT. Ela estende a *cloud* para mais perto do usuário, de modo que tarefas de acesso, processamento e armazenamento de dados são realizadas por recursos locais, como roteadores, *gateways* e *switches*. Assim sendo, o processamento e armazenamento de dados temporários e execução de análises locais são realizadas sem a necessidade de longas transmissões através da Internet. Dessa maneira, *fog computing* não sofre com problemas de alta latência e *jitter* (Ni et al., 2018).

2.3 FOG COMPUTING

Os autores Bonomi et al. (2012) definem a *Fog Computing* como uma plataforma altamente virtualizada que prove computação, armazenamento e serviços de rede entre os dispositivos IoT e os *datacenters* da nuvem. Além disso, estando geralmente próximo dos dispositivos IoT na borda da rede.

A ideia principal desse paradigma é estender a computação em nuvem para mais próximo dos dispositivos finais, de modo a prover acesso, processamento e armazenamento eficiente de dados. Portanto, a característica marcante da computação em nevoeiro é a distribuição dos recursos, serviços de comunicação, processamento e armazenamento próximo dos usuários (Marín-Tordera et al., 2017).

A computação em nevoeiro não surgiu para substituir a computação em nuvem para processamento e armazenamento remoto, mas sim para complementar. Permitindo criar uma infraestrutura hierárquica onde os dados locais são processados e armazenados pela computação em nevoeiro e o armazenamento permanente e análises globais são realizadas nos *datacenters* da nuvem (Ni et al., 2018).

Por ser um paradigma recente, as pesquisas em segurança e em problemas de privacidade ainda estão em estágio inicial.

2.4 VULNERABILIDADES E AMEAÇAS EM COMPUTAÇÃO EM NEVOEIRO E IOT

Este trabalho considera o contexto de ambientes inteligentes baseados em computação em nevoeiro e IoT. A segurança nesses ambientes é fundamental, pois os dispositivos IoT geralmente são incorporados ao cotidiano das pessoas e lidam com informações confidenciais. Além disso, alguns sistemas realizam monitoramentos e realizam ações críticas, que precisam ter uma operação ininterrupta.

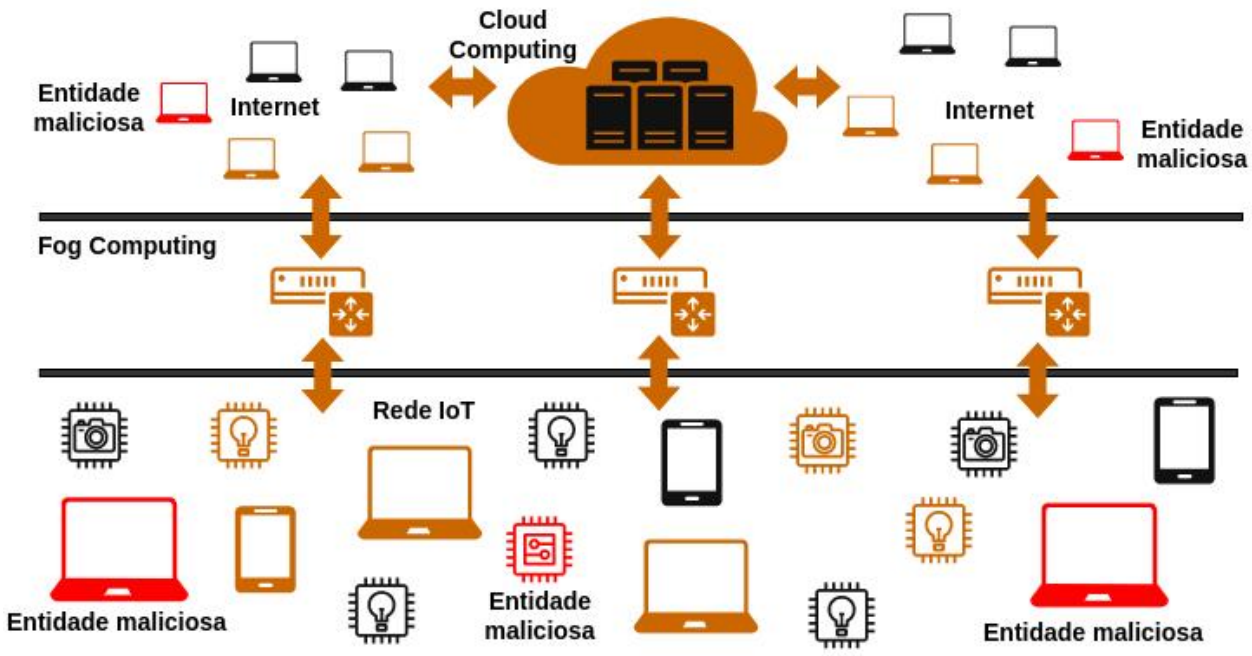
As soluções de IoT e o nevoeiro são compostas por várias tecnologias, serviços e padrões, cada um com seus próprios requisitos de segurança e privacidade (Zarpelão et al., 2017). O paradigma IoT apresenta várias vulnerabilidades de segurança que as redes de comunicação, serviços em nuvem e a Internet possuem (Zarpelão et al., 2017). No entanto, as ferramentas de segurança tradicionais têm dificuldades de serem aplicadas diretamente neste contexto devido a três aspectos fundamentais: o poder computacional limitado dos componentes de IoT, o alto número de dispositivos interconectados e o compartilhamento de dados entre objetos e usuários (Sicari et al., 2015). Além disso, a rápida expansão das soluções de IoT deixou essas redes vulneráveis a riscos de segurança e privacidade. Os autores Kolias et al. (2016) descobriram várias vulnerabilidades de segurança criando casos de uso de IoT usando produtos e serviços comerciais populares. A computação em nevoeiro surgiu não só para fornecer maiores recursos computacionais para a IoT, mas também baixa latência e uso intensivo de computação. Estes fatos a tornam um ótimo local para implantar aplicativos de segurança da IoT como detectores de intrusão (Nguyen et al., 2019). No entanto, a pesquisa em segurança relacionada a aplicações de computação em nevoeiro e IoT ainda está em estágio inicial (Ni et al., 2018). Este fato, aliado aos grandes danos que podem ser gerados por ataques nesse ambiente, gera a necessidade de concentração de esforços nessa área.

Os autores (Garcia-Morchon et al., 2013) organizaram as ameaças à segurança de ambientes IoT nas seguintes categorias: clonagem de dispositivos, substituição maliciosa de dispositivos, substituição de *firmware*, extração de parâmetros de segurança, interceptação, ataque do homem do meio (*Man-In-The-Middle - MITM*), ataque de roteamento, negação de serviço (*Denial of Service* - DoS) e negação de serviço distribuída (*Distributed DoS* - DDoS). As ameaças relacionadas a clonagem, substituição e extração ocorrem geralmente durante a fabricação, instalação, manutenção e atualização dos dispositivos (Zarpelão et al., 2017). Desse modo, não serão consideradas neste trabalho.

Kolias et al. (2016) destaca como principais ataques presentes na IoT: DoS, DDoS, MITM, ataques de roteamento e ataques convencionais. Ameaças de segurança relacionadas a tecnologias convencionais que fazem parte do ambiente IoT, também podem se aplicar, por exemplo, conexões não seguras, injeção de código malicioso, sondagem, interceptação, fabricação e modificação de mensagens (Muhammad; Anjum; Mazhar, 2015).

Conforme ilustrado na Figura 1, esses ambientes estão sujeitos a ataques de fontes externas, da Internet, e ataques internos, por dispositivos maliciosos na rede IoT. Entidades maliciosas fora da rede podem tentar obter acesso privilegiado a dispositivos IoT para controlá-los (Muhammad; Anjum; Mazhar, 2015). Através desse acesso, é possível realizar ataques de *botnet*, onde dispositivos IoT comprometidos podem ser usados como *bots* ou zumbis, para realizar diversas tarefas maliciosas (Aversano et al., 2021). Além disso, os ataques de *spoofing* envolvem a personificação de dispositivos legítimos, explorando suas identidades para obter acesso à rede IoT e, em seguida, lançando outros tipos de ações maliciosas (Aversano et al., 2021), como roubar informações confidenciais manipuladas na rede IoT. Os ataques DoS são bastante comuns e visam afetar a disponibilidade da vítima. Isso pode ser feito por meio de inundações com um volume muito grande de solicitações ou esgotando recursos como memória e poder de computação. No contexto da IoT, o dispositivo pode fazer parte da rede sob ameaça ou ser usado como zumbi para lançar um DDoS em outra rede. Ataques de sondagem, em que o invasor examina uma rede para coletar informações e descobrir vulnerabilidades, também são considerados comuns. Geralmente é um ataque que coleta informações do alvo antes de iniciar outro tipo de ataque mais severo (Arshad et al., 2019).

Figura 1 – Ilustração de possíveis ameaças em um ambiente de computação em nevoeiro e IoT.



No caso de um ataque interno, presume-se que o invasor seja uma entidade maliciosa

que se autenticou com sucesso no nevoeiro ou um dispositivo IoT legítimo que se tornou malicioso ao longo do tempo, ele pode realizar vários ataques, incluindo negação de serviço, envio uma alta taxa de pacotes de dados no nevoeiro. Isso permite sobrecarregar os dispositivos e as camadas superiores, conforme ilustrado na Figura 1, prejudicando ou mesmo causando a interrupção dos serviços prestados pelos sistemas, serviços que em muitos casos são extremamente importantes. Portanto, por meio dessas ações, todo o sistema IoT pode sofrer danos e usuários externos que acessam serviços e informações geradas pela solução IoT podem ser prejudicados.

Baseado na classificação apresentada pelos autores Zarpelão et al. (2017), são apresentados abaixo os principais ataques encontrados no ambiente de nevoeiro e IoT.

- *Denial of Service* (DoS): ataques de negação de serviço tem como objetivo afetar a disponibilidade da vítima. Isto pode ser feito através de inundação com um volume enorme de pedidos ou esgotando os recursos como memória e poder computacional (Zarpelão et al., 2017). Tanto um nodo IoT, quanto o nevoeiro, podem ser vítimas desse ataque.

- *Distributed DoS* (DDoS): os ataques distribuídos de negação de serviço possuem o mesmo objetivo de um DoS. No entanto, são executados por um conjunto de *hosts*, enquanto, em um DoS comum um único *host* é o atacante (Yan et al., 2016). Além disso, no contexto da internet das coisas, o nodo IoT pode fazer parte da rede sob ameaça ou ser usado como um zumbi para lançar um DDoS em outra rede.

- *Man-In-The-Middle* (MITM): um ataque de homem do meio é realizado quando um invasor interfere na comunicação entre uma entidade A e uma entidade B, sem que A e B percebam (Zarpelão et al., 2017). Os autores Navas et al. (2018) demonstram os riscos de ataques MITM em redes IoT provocados por dispositivos internos maliciosos.

- Ataques de roteamento: os ataques de roteamento consistem em falsificar, modificar informações de roteamento da rede para criar *loops*, atrair ou rejeitar o tráfego, estender ou encurtar rotas e assim por diante. Outros possíveis ataques de roteamento incluem *sinkhole attack*, *selective forwarding*, *wormhole attack* e *sybil attack* (Zarpelão et al., 2017).

- Ataques convencionais: as ameaças à segurança, relacionadas a tecnologias convencionais que fazem parte do ambiente de IoT, também podem se aplicar aos sistemas IoT, por exemplo, conexões não seguras, injeção de código malicioso, interceptação, sondagem, fabricação e modificação de mensagens (Muhammad; Anjum; Mazhar, 2015).

Além das ameaças já existentes relacionadas a redes de computadores, a IoT precisa lidar com as restrições de recurso que seus dispositivos possuem. A seguir são apresentados os conceitos básicos relacionados a sistemas de detecção e prevenção de intrusão. Esses sistemas podem ser utilizados para proteger redes IoT.

2.5 DETECÇÃO DE INTRUSÃO

Uma intrusão pode ser definida como um conjunto de ações realizadas com o intuito de ultrapassar as barreiras de defesa de uma aplicação para comprometer a integridade, confidencialidade e disponibilidade de recursos (Heady et al., August 1990). Os IDSs têm como objetivo reconhecer ações e comportamentos intrusivos para alertar os administradores ou executar contramedidas automaticamente (Bace; Mell, 2001).

Os esforços em pesquisa de detecção de intrusão são conduzidos desde 1980, quando (Anderson, 1980) apresentou um modelo de ameaça e um sistema de monitoramento de segurança baseado na detecção de anomalias no comportamento do usuário. Os IDSs são inseridos como a última linha de defesa dentro de uma arquitetura computacional, o que os torna de grande importância, possibilitando inferir sobre a legitimidade de ações realizadas e possuindo comportamento pró-ativo em situações de ataque (Patel; Qassim; Wills, 2010). A estrutura dos IDSs pode variar em relação ao modo que é implantada, pela frequência de operação, pelos dados que analisam ou pelas análises realizadas sobre eles (Campello; Weber, 2001).

Os IDSs podem ser classificados de acordo com os métodos de detecção empregados. Desse modo, podem ser classificados em análise por assinatura ou análise por comportamento, está ultima também conhecida como análise por anomalia. Na detecção por assinatura, as ações monitoradas são comparadas com eventos intrusivos predefinidos, normalmente armazenadas em uma base de dados. Esses padrões previamente conhecidos são chamados de assinaturas e apesar de permitir uma rápida detecção e diminuir a ocorrência de alarmes falsos, possui a limitação de apenas detectar esses ataques conhecidos (Northcutt et al., 2001). Este método é utilizado pela maioria dos sistemas de antivírus comerciais (Bace; Mell, 2001). A detecção de anomalias assume que qualquer atividade anômala é necessariamente uma intrusão. Qualquer atividade que não se enquadre nos modelos de comportamento definidos como normais é considerada um ataque. A grande vantagem da técnica de detecção de anomalias é que permite detectar novos ataques e/ou variações dos já conhecidos, uma vez que não é necessário conhecê-los previamente. No entanto, esta técnica tem maior chance de sofrer com problemas relacionados a falsos positivos (Boukerche et al., 2007). Está estratégia geralmente é modelada com a utilização de técnicas de Aprendizado de Máquina, maiores detalhes a respeito dessas técnicas são apresentados na Seção 2.6. Além disso, alguns trabalhos consideram uma ramificação da análise por comportamento chamada de análise por especificação (Mitchell; Chen, 2014). Este tipo de solução emprega regras e limites que definem o comportamento padrão esperado para os componentes monitorados. É semelhante à detecção de anomalias. Ambos detectam intrusões quando o comportamento da rede se desvia do especificado. A principal diferença entre eles é que na análise baseada em especificações, um especialista humano é quem estabelece as regras (Mitchell; Chen, 2014; Zarpelão et al., 2017). A grande desvantagem deste tipo de análise é a especificidade necessária e o conhecimento necessário do domínio para especificar o comportamento benigno.

Os IDSs podem analisar dados de várias origens e podem ser implantadas em diferentes

locais. Esses dados geralmente estão relacionados com a forma de implantação da abordagem. Existem duas categorias principais de implantação relacionadas ao foco de captura de informações, os IDSs baseados em *host Host-Based Intrusion Detection System* - HIDS) que buscam analisar informações capturadas do próprio *host* onde estão implantados e os IDSs baseados em rede (*Network-Based Intrusion Detection System* - NIDS) que analisam o tráfego capturado da rede monitorada (Zarpelão et al., 2017). Além disso, no contexto deste trabalho as abordagens podem ser implantadas em diferentes níveis: nos próprios dispositivos IoT, nos dispositivos do nevoeiro ou na nuvem. Em um IDS baseado em *host*, todos os componentes, desde a coleta dos eventos até a classificação, estão localizados no mesmo *host*. Os mecanismos de monitoramento de eventos e de análise utilizam apenas informações do próprio computador. Os eventos podem ser originados a partir de *logs* do sistema, dados sobre usuários, serviços e processos. Essa abordagem possibilita a independência da rede e a detecção de ataques internos. No entanto, as soluções baseadas em *host* empregadas nos próprios nós IoT podem sofrer com restrições de memórias. Já as empregadas nos dispositivos do nevoeiro permitem a detecção de ataques contra o próprio dispositivo, mas podem ter dificuldade em lidar com ataques à rede e aos dispositivos IoT. Por outro lado, as abordagens NIDS são implantadas em um dispositivo com capacidade de capturar o tráfego da rede que se pretende monitorar. Os eventos e atividades são obtidos capturando o tráfego da rede em modo promíscuo. Esses IDSs geralmente monitoram uma rede composta de vários dispositivos. Podem ser utilizados também sensores de modo a capturar informações de maneira distribuída em vários pontos da rede. Uma das dificuldades dessa abordagem é determinar os melhores locais para posicionar os sensores de captura de informações. O método de análise, nessas abordagens, foi geralmente incluído nos dispositivos de névoa. Esta estratégia permite detectar ataques externos e é mais independente da plataforma (Mukherjee; Heberlein; Levitt, 1994). A computação em nevoeiro é uma das alternativas mais promissoras para a implantação de abordagens para monitoramento da rede IoT. Além disso, é interessante dividir as tarefas de detecção ao longo da arquitetura completa, considerando dispositivos IoT, nevoeiro e nuvem.

Além de detectar as intrusões, é muito importante ter mecanismos para executar ações de contramedidas, com o objetivo de bloquear e evitar que a intrusão obtenha sucesso. Dentre as ações existentes estão a emissões de alerta para o gerenciador da rede. A emissão de apenas um alerta não configura uma ação de prevenção, pois apenas deixa o gerenciador ciente que ocorreu uma intrusão, mas não a evita. A emissão de alerta é considerada uma pós-detecção passiva. Outra classe de abordagens pós-detecção é a ativa, onde as ações executadas tem como objetivo interromper um ataque em andamento e em seguida bloquear o acesso do atacante (Bace; Mell, 2001). Os IDSs que possuem contramedidas ativas são conhecidos como Sistemas de Prevenção de Intrusão (*Intrusion Prevention Systens* - IPS) (Birkinshaw; Rouka; Vassilakis, 2019).

Nas seções a seguir, são apresentados outros conceitos envolvidos no contexto deste trabalho. A Seção 2.6 apresenta os conceitos básicos de ML e uma breve descrição da sua aplicabilidade para a detecção de intrusão por anomalia.

2.6 MACHINE LEARNING

Em Russell e Norvig (2010) são apresentadas várias definições para o termo Inteligência Artificial (IA), que de forma geral, apontam esta como a capacidade de fazer máquinas reproduzirem atividades inteligentes e capacidades cognitivas encontradas em humanos. Ao analisarmos de modo mais enfático as soluções atualmente propostas na área de segurança computacional, verifica-se cada vez mais forte a pesquisa e a aplicação de métodos de aprendizado de máquina para melhorias no processo de detecção de intrusões.

De acordo com Goodfellow, Bengio e Courville (2016), *Machine Learning* é a capacidade de uma determinada técnica adquirir seu próprio conhecimento, extraindo informações de dados brutos e os representando através de algum tipo de modelo matemático. A ML possui diversas sub-áreas, dentre elas a de classificação.

2.6.1 Técnicas de classificação

Uma tarefa de classificação consiste em classificar dados e objetos em determinadas classes de maneira automatizada. Os métodos de aprendizado de máquina são capazes de realizar esse tipo de tarefa. Eles geralmente precisam ser treinados para adquirir conhecimento e gerar um modelo com conhecimento agregado. A seguir são apresentados os tipos de aprendizagem mais comuns empregados em soluções de aprendizado de máquina. Quatro categorias são destacadas: aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem por reforço e aprendizagem semi-supervisionada.

A principal característica das abordagens baseadas em aprendizagem supervisionada é a existência de rótulos no subconjunto de dados de treinamento. Este tipo de aprendizagem reflete a capacidade de um algoritmo de generalizar o conhecimento a partir de dados disponíveis com casos alvo ou rotulados, para que o algoritmo possa ser usado para prever novos casos não rotulados (Berry; Mohamed; Yap, 2019). Assim, o processo de treinamento do método utiliza esse conhecimento prévio para treinar e gerar os modelos de classificação. Após o treinamento, os métodos podem classificar novos dados. A dificuldade desta abordagem reside na necessidade de dados rotulados para treinar os modelos (Russell; Norvig, 2009).

A aprendizagem não supervisionada refere-se ao agrupamento de dados em dados não rotulados usando métodos ou algoritmos automatizados. Nessa situação, os algoritmos precisam compreender os relacionamentos ou recursos subjacentes dos dados disponíveis e agrupar casos com recursos ou características semelhantes (Berry; Mohamed; Yap, 2019).

As abordagens semi-supervisionadas treinam um classificador específico usando dados rotulados e não rotulados. Devido ao esforço e tempo necessários para rotular registros, o aprendizado semi-supervisionado torna-se interessante para treinar modelos usando uma combinação de um grande número de registros não rotulados e um pequeno número de registros rotulados (Berry; Mohamed; Yap, 2019).

Outro tipo de aprendizagem é a por reforço, onde o método aprende como atingir uma

meta em um ambiente incerto e potencialmente complexo. Inicialmente, ele usa tentativa e erro para encontrar uma solução para o problema. Recompensas ou penalidades são fornecidas para as ações realizadas pela abordagem. Seu objetivo é maximizar a recompensa total (Russell; Norvig, 2009; Sutton; Barto, 2018).

Observa-se que esta capacidade dos métodos de classificação de aprendizado de máquina se enquadra perfeitamente no contexto de detecção de intrusão, uma vez que as abordagens de detecção têm a tarefa de analisar as informações capturadas da rede ou dos hosts, verificar a ocorrência de comportamentos anormais e realizar a classificação de informações em benignas ou intrusivas. Além disso, há também a necessidade de classificar os ataques em tipos ou categorias. Portanto, as técnicas de classificação são ótimas para compor abordagens de detecção baseadas em anomalias. Existem muitos algoritmos de classificação que podem ser empregados no contexto de detecção de intrusão, dentre eles as Redes Neurais Artificiais (*Artificial Neural Networks* - ANN).

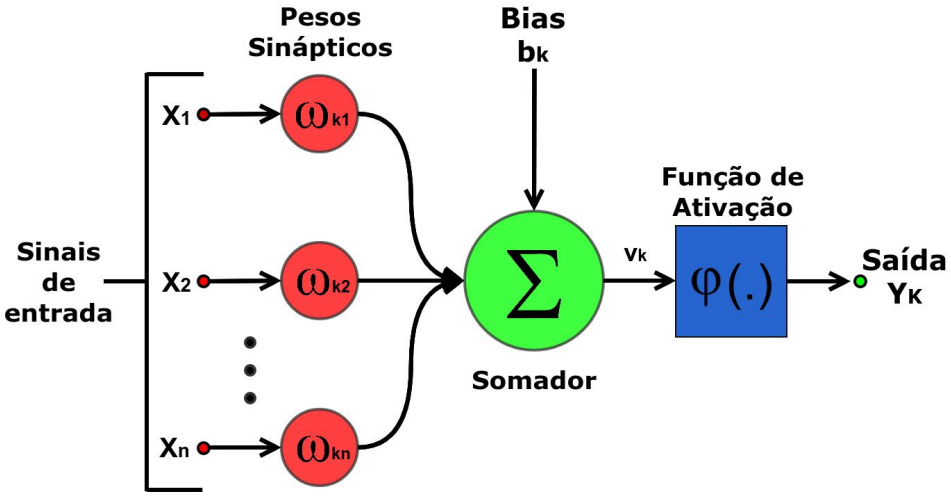
2.6.1.1 Redes Neurais Artificiais

O cérebro possui neurônios densamente interconectados formando uma estrutura altamente complexa, inspiradas nela, as ANN foram propostas (Haykin, 2001). O neurônio artificial é uma estrutura lógico matemática que tem como objetivo simular a forma, o comportamento e as funções do neurônio biológico. Os dentritos são substituídos por entradas, e as ligações dessas entradas com o corpo celular artificial são conhecidas como pesos, os quais simulam as sinapses. Os estímulos recebidos pelas entradas são processados pela função de soma e o limiar de disparo do neurônio biológico é simulado pela função de ativação no neurônio artificial (Chua; Yang, 1988). De acordo com Dalton e Deshmane (1991), os pesos sinápticos possuem importante papel em neurônios artificiais. O propósito dos pesos é ponderar a influência dos sinais de entrada nos neurônios pós-sinápticos. Os pesos positivos tendem a incrementar o nível de ativação de um neurônio, consistindo em uma conexão excitatória. Já os pesos negativos tendem a diminuir o nível de ativação, chamadas então de conexões inibitórias.

A Figura 2 apresenta o modelo simplificado de um neurônio artificial. Onde o neurônio k recebe uma entrada x $(x_1, x_2, ..., x_n)$ que entra pela sinapse j. Cada sinal x_j da entrada x que entra pela sinapse j é multiplicada por um peso w_{kj}. O resultado deste processo passa por um somador que soma os sinais de entrada ponderados pelos pesos das respectivas sinapses com um *bias* externo (b_k). Em relação a determinado peso sináptico w_{kj}, o primeiro índice (k) refere-se ao neurônio em questão e o segundo índice (j) refere-se a entrada da sinapse ao qual o peso está relacionado (Haykin, 2001). O *bias* b_k citado anteriormente tem o papel de aumentar ou diminuir o valor gerado pelo somador antes de repassá-lo para uma função de ativação, que então irá transformar a saída em um intervalo fechado geralmente entre $[0, 1]$ ou $[-1, 1]$ para ser repassado para outros neurônios (Haykin, 2001).

A função de ativação, que na Figura 2 está representada por $\varphi(.)$ tem extrema importância no modelo neural, pois define qual será a saída do neurônio conforme a entrada recebida

Figura 2 – Modelo simplificado de um Neurônio Artificial.



Fonte: Adaptado de Haykin (2001).

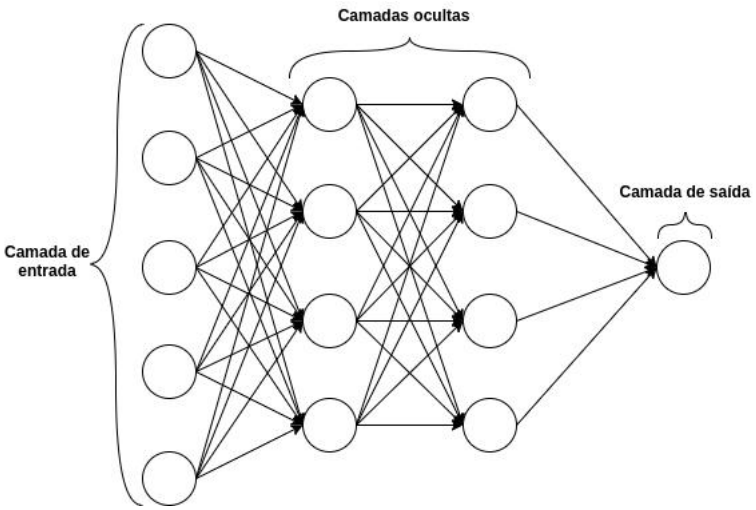
(Haykin, 2001). Portanto, temos que a função de ativação $\varphi(v)$ define a saída do neurônio conforme o resultado do somatório (v) das entradas ponderadas. Existem vários tipos de função de ativação utilizadas em redes neurais, entre elas a Limiar, Linear por partes, Sigmoide logística, Tangente hiperbólica, *Rectified Linear Unit* (*ReLU*) e a *Softmax*. A função de ativação sigmoide tem gráfico em forma de "S" e assume um intervalo contínuo de valores entre 0 e 1. Ela é uma das mais comuns funções de ativação usadas em ANN e exibe um balanceamento adequado entre comportamento linear e não linear. A função tangente hiperbólica é similar a sigmoide logística, porém seus valores contínuos variam de -1 à 1. O fato de permitir que uma função de ativação assuma valores negativos fornece benefícios analíticos e vantagens durante a fase de treinamento (Haykin, 2001). A *softmax* é uma generalização da função sigmoide para casos não-binários. Ela não costuma ser aplicada às camadas escondidas da rede neural, mas sim na camada de saída de problemas de classificação multiclasse. A função *softmax* transforma as saídas para cada classe para valores entre 0 e 1 e as divide pela soma das saídas. Isso essencialmente dá a probabilidade de a entrada estar em uma determinada classe. Por fim, a *ReLU* é uma abreviação para *Rectified Linear Unit* (ReLU), ou unidade linear retificada. Ela retorna 0 para todos os valores negativos, e o próprio valor para valores positivos. Desse modo, se a entrada for negativa, o neurônio não será ativado. Isso significa que, simultaneamente, apenas alguns neurônios são ativados, tornando a rede esparsa e eficiente. Portanto, é uma função computacionalmente leve (Goodfellow; Bengio; Courville, 2016).

Uma característica essencial para uma ANN é a habilidade de aprender através de seu ambiente e de melhorar seu desempenho através do processo de treinamento. Uma ANN aprende mais sobre o ambiente em que está inserida por um processo iterativo de ajuste dos seus pesos sinápticos e níveis de *bias*. Este processo é definido como treinamento. A ANN se torna mais conhecedora de seu ambiente após cada iteração do processo de treinamento (Haykin, 2001). O conhecimento da ANN é representado por meio dos pesos sinápticos, formando

uma representação compacta e distribuída, desse modo, proporcionando capacidades de generalização e adaptabilidade à rede neural. As ANN conseguem alcançar ótimos desempenhos de classificação. Entretanto, podem sofrer com instabilidades causadas por ruídos e variança no treinamento. Essa instabilidade significa que pequenas alterações nos dados de treinamento usados para construir o modelo podem resultar em modelos muito diferentes (Cunningham; Carney; Jacob, 2000).

A maneira que os neurônios estão dispostos em uma ANN está diretamente relacionada com o algoritmo de aprendizagem. Para efeito do presente trabalho, abordaremos a classe das redes *Multilayer Perceptron* (MLP) *feedforward* (Russell; Norvig, 2010). Este modelo não possui ciclos de realimentação entre os neurônios, portando o fluxo do processo sináptico ocorre da camada de entrada em direção a saída. A arquitetura disposta na Figura 3 ilustra uma estrutura básica comum de redes neurais MLP. Embora possa haver diversas variações no *design* das camadas ocultas, tanto em termos do número de camadas, quanto em relação à quantidade de neurônios existentes em cada uma delas. O *design* dessas camadas intermediárias, ou ocultas, é ponto crucial na definição de uma rede neural, principalmente das *Deep Neural Networks* (DNN). As DNNs são redes profundas, com mais de uma camada oculta. A área de que estuda redes profundas é denominada Aprendizado Profundo *Deep Learning (DL)* (Goodfellow; Bengio; Courville, 2016).

Figura 3 – Arquitetura simplificada de uma rede neural *feedforward* profunda.



O treinamento de redes MLP é geralmente realizado através do algoritmo *backpropagation*. O treinamento se dá mediante propagação dos pesos sinápticos da camada de entrada para a saída, passando por cada uma das camadas ocultas, sendo que nesse momento os pesos mantêm-se inalterados. Após, com base no erro calculado utilizando-se do resultado esperado (aprendizado supervisionado) e do valor de saída da última camada, os pesos são ajustados e realiza-se uma nova iteração do treinamento. Considera-se que o treinamento foi concluído quando o erro for suficientemente pequeno. A partir disso, a rede passa a operar apenas em sentido *forward* para a classificação de novos exemplos.

As redes neurais podem apresentar um custo computacional elevado a medida que o número de camadas ocultas é aumentado. Este custo esta atrelado principalmente a tarefa de treinamento do modelo neural. A *Extreme Learning Machine* (ELM) surgiu para superar as limitações da ANN (Huang; Zhu; Siew, 2006). ELM é uma rede neural *feedforward* de camada oculta única que escolhe aleatoriamente os pesos de entrada e o viés da camada oculta sem ajuste e determina os pesos de saída de forma analítica. Esse algoritmo tende a fornecer bom desempenho de generalização em velocidade de aprendizado extremamente rápida.

As redes neurais profundas são avanços sobre as redes tradicionais simples. Adicionando mais camadas e mais unidades dentro de uma camada, uma rede profunda pode representar funções de complexidade crescente (Goodfellow et al., 2016).

Recurrent Neural Network (RNN) é um tipo de rede neural profunda que estende os recursos da rede neural tradicional *feed-forward* e é projetada para modelar dados de sequência. Elas podem usar seu estado interno de memória para processar sequências de comprimento variável de entradas (Ahmad et al., 2021). As RNNs possuem algumas dificuldades. A RNN básica normalmente é capaz de lidar com sequências de comprimento limitado e sofrerá de memória de curto prazo se o comprimento da sequência for longo (Ahmad et al., 2021). Além disso, está sujeita a problemas de gradiente que desaparecem. Desse modo, variações foram propostas para superar esses problemas (Samy; Yu; Zhang, 2020).

A *Long Short Term Memory* (LSTM) consiste em uma variação de RNN projetada para lidar com problemas de gradiente em expansão e desaparecimento (Samy; Yu; Zhang, 2020). Ela é capaz de lembrar informações por um longo período de tempo. Essa propriedade de lembrar padrões dá ao LSTM uma vantagem sobre os RNNs tradicionais (Hochreiter; Schmidhuber, 1997).

Convolutional Neural Network (CNN) é um tipo de rede neural profunda que usa convolução no lugar da multiplicação geral da matriz, em pelo menos uma de suas camadas (Goodfellow et al., 2016). Essas redes tem alcançado ótimos resultados em problemas de processamento de dados bidimensionais com topologia em grade, como imagens e vídeos (Liu et al., 2017). Redes convolucionais, normalmente têm interações esparsas. Isso é feito tornando o kernel menor do que a entrada. Desse modo, é necessário armazenar menos parâmetros, o que reduz os requisitos de memória do modelo e melhora sua eficiência. Também significa que computar a saída requer menos operações (Goodfellow et al., 2016). A arquitetura da CNN consiste em três camadas empilhadas: camada convolucional, camada de *pooling* e camada totalmente conectada. As camadas convolucionais e de *pooling* são conectadas alternadamente e formam a parte intermediária da rede. As camadas convolucionais aplicam vários filtros (*kernels*) de tamanho igual para produzir mapas de recursos. As camadas de *pooling* substituem as saídas da rede em um determinado local por estatísticas resumidas das saídas próximas. A objetivo da camada de *pooling* é diminuir o tamanho da imagem para encontrar possíveis padrões que possam ficar mais evidentes. A dimensão dos dados de entrada é reduzida usando estas camadas de *pooling*. Ele combina um cluster em um único ponto. Por fim, a camada totalmente conectada combina as entradas de todas as posições para realizar a classificação (Farukee et al., 2020).

As redes neurais profundas são mais robustas e apresentam ótimos resultados, no entanto, possuem estruturas mais complexas, o que demanda maior poder computacional.

2.6.1.2 k-Nearest Neighbors

O algoritmo *K-Nearest Neighbors* (KNN) é o método mais básico de aprendizado baseado em instâncias, o qual assume que todos os exemplos correspondem a pontos em um plano n-dimensional R^n, onde n é o número de atributos utilizados para representá-los. Apesar de ter um funcionamento simples, o KNN geralmente apresenta uma taxa de erro muito pequena (Cover; Hart, 1967; Taneja et al., 2015). Este método utiliza uma função distância para determinar o quão próximo uma instância está da outra (Mitchell, 1997).

Para calcular a similaridade, quando o conjunto de dados é descrito por atributos numéricos, são utilizadas medidas de distâncias, de modo que a menor distância corresponde a maior similaridade. Dentre as medidas comumente aplicadas, destaca-se a grande utilização da Distância Euclidiana apresentada na Equação 2.1 (Mitchell, 1997). Onde, $p = (p_1, ..p_n)$ e $q = (q_1, ..q_n)$ são dois pontos n-dimensionais.

$$D = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{1}^{n} (p_i - q_i)^2} \tag{2.1}$$

O algoritmo KNN classifica novas instâncias considerando a classe dos k exemplos mais próximos. Se $k = 1$, então é atribuída a essa nova instância a mesma classe do exemplo mais próximo segundo a medida de similaridade utilizada. Se $k > 1$, então são consideradas as classes dos k exemplos mais próximos para realizar a classificação, sendo que neste caso, a abordagem mais comum é atribuir a nova instância a classe majoritária presente no conjunto dos k exemplos mais próximos.

2.6.1.3 Support Vector Machine

Support Vector Machine (SVM) é outra técnica que pode ser utilizada em tarefas de classificação, ela cria um hiperplano de divisão nos atributos de dados entre duas ou mais classes, de forma que a distância entre o hiperplano e os pontos de amostra mais adjacentes de cada classe seja maximizada (Tong; Koller, 2002). Basicamente utiliza uma função kernel para mapear o conjunto de dados separáveis não lineares para o espaço de alta dimensão, de modo que tenha separabilidade para obter a classificação (Du et al., 2020).

2.6.1.4 Naïve Bayes

Naïve Bayes (NB) tem como conceito adquirir, por meio do treinamento, a probabilidade condicional. Basicamente, é a probabilidade de um evento ou resultado ocorrer com base em ocorrências anteriores. Ele estima a média e a variância para cada recurso usando a aborda-

gem de máxima verossimilhança. Além disso, ele assume que todos os vetores de recursos são independentes. Ele tem uma base simples e confiável para modelar dados e é bastante resiliente a valores discrepantes e ausentes (Iwendi et al., 2020; Manimurugan, 2021; Kumar; Gupta; Tripathi, 2021b).

2.6.1.5 Decision Tree

Decision Tree é um tipo de algoritmo de aprendizagem de máquina supervisionado que se baseia na ideia de divisão recursiva de um problema de maior complexidade em problemas mais simples. Os dados de entrada são divididos em grupos homogêneos onde, cada divisão realizada representa um nodo da árvore onde, os dados são separados de acordo com um critério de divisão até chegar a pontos indivisíveis. As DTs têm uma estrutura em forma de árvore sendo composta de nodos. Esses nodos podem ser divididos em um nodo raiz, um conjunto de nodos intermediários e um conjunto de nodos folha (Breiman et al., 1984). O nodo raiz corresponde à primeira divisão que especifica como os dados devem ser divididos em partições separadas. Nodos intermediários sucessivos continuam a dividir os dados em partições menores até que não seja necessário mais particionamento. Desta forma, os nodos folha da estrutura representam as partições finais (Rokach, 2016). As DTs classificam usando um conjunto de decisões hierárquicas de recursos. As decisões tomadas nos nodos internos são os critérios de divisão (Dev; Eden, 2019).

O algoritmo básico de indução da árvore de decisão constrói árvores de decisão recursivamente com base na divisão e conquista, começando de cima para baixo. Em cada iteração, procura-se o atributo capaz de dividir melhor o conjunto de dados. Uma boa divisão em uma árvore de decisão corresponde à escolha do atributo com o máximo poder de separação. Em outras palavras, o objetivo de cada nodo é criar nodos filhos onde predomina uma única classe. A seleção do atributo mais adequado é feita de acordo com critérios específicos de divisão. Os atributos são avaliados conforme o critério de divisão, sendo selecionado o melhor atributo. O processo é recursivo, então cada nodo subdivide ainda mais o conjunto de treinamento em subconjuntos menores após selecionar uma divisão apropriada. Para atributos numéricos, há muitos pontos de corte possíveis. O algoritmo de indução procura o melhor ponto de corte avaliando o critério de divisão em cada ponto de corte possível (Rokach, 2016). Quando o nodo satisfaz as regras de parada, por exemplo, porque todas as instâncias da partição atual pertencem à mesma classe ou porque nenhum atributo de divisão futura pode ser determinado, a DT termina o processo de divisão e o nodo é rotulado conforme a classe do conjunto de dados (Rokach, 2016). Um dos desafios significativos dos algoritmos de árvore de decisão é encontrar o atributo que melhor divide os dados em suas classes correspondentes. As principais métricas são o *Gini Index* e *Information Gain* (IG).

O coeficiente de Gini mede quão bem um determinado atributo separa as classes contidas em um nó. Os valores possíveis para o índice de Gini variam entre 0 e 1, onde 0 expressa a pureza da classificação, ou seja, todos os elementos pertencem a uma determinada classe

ou nela existe apenas uma classe. Além disso, 1 indica uma distribuição totalmente desigual (Sundhari, 2011). O índice de Gini é determinado subtraindo de 1 a soma dos quadrados das probabilidades de cada classe. É expresso matematicamente na Equação 2.2 (Breiman et al., 1984). Onde P_i denota a probabilidade de que um elemento seja classificado para uma classe distinta. Ao construir a árvore de decisão, são escolhidos os recursos com os menores valores do Índice de Gini (Sundhari, 2011).

$$GiniIndex = 1 - \sum_{i=1}^{C} (p_i)^2 \tag{2.2}$$

2.6.1.6 Métodos Ensemble

Ensemble Learning é a área de estudo do aprendizado de máquina que se dedica aos métodos *ensemble*. Esses métodos combinam as decisões de vários modelos de classificação para aprimorar o desempenho global. As classificações dos vários modelos são combinadas de alguma forma para gerar a classificação final (Breiman, 1996; Wang et al., 2011; Marqués; García; Sánchez, 2012; Tsai; Hsu; Yen, 2014).

Classificadores individuais podem sofrer instabilidade. Não há garantia de que um classificador sempre terá o melhor desempenho em todas as situações. No entanto, com *Ensemble Learning*, pode ser alcançado um melhor desempenho de classificação do que qualquer classificador individual (Traganitis; Pagès-Zamora; Giannakis, 2018).

A ideia principal do aprendizado conjunto é que combinar vários modelos diminui a variação, especialmente no caso de classificadores instáveis. Desta forma, é possível produzir uma classificação mais confiável do que um modelo único (Breiman, 1996).

As técnicas de *ensemble* podem ser úteis na detecção de intrusões, permitindo construir um classificador forte para identificar a classe específicas de um determinado ataque. No entanto, por utilizar diversos classificadores essas técnicas podem apresentar um maior custo de tempo de processamento e de treinamento dos modelos.

Umas das principais estratégias para criar métodos *ensemble* é o *Bagging* (*Bootstrap Aggregating*). Criado por Breiman (1996), o *Bagging* gera vários modelos de um classificador e os utiliza para obter um classificador agregado. São criadas amostras aleatórias do conjunto de dados de treinamento para cada modelo, ou seja, subconjuntos do conjunto de dados de treinamento. Cada modelo é treinado com um subconjunto. Finalmente, os resultados desses vários modelos são combinados usando votação média ou majoritária. Testes em conjuntos de dados reais e simulados usando árvores de classificação mostram que o *Bagging* pode proporcionar ganhos substanciais em precisão. A combinação de vários modelos reduz a variação, especialmente no caso de classificadores instáveis. Desse modo, é possível produzir uma classificação mais confiável do que um único modelo (Breiman, 1996).

O *Boosting* é outra estratégia *ensemble*, ele produz um classificador com alta acurácia, combinando vários modelos "fracos". Cada um desses modelos pode não ser eficaz para

todo o conjunto de dados, mas é bom para parte específica, resultando no aprimoramento do desempenho global (Schapire, 1990). Assim como o *bagging*, o *boosting* treina cada modelo usando um conjunto de treinamento diferente. Essa abordagem é iterativa e ajusta o peso de uma observação com base na última classificação. Cada modelo gerado é influenciado pelo desempenho dos modelos gerados anteriormente. A estratégia do *boosting* é focar nos exemplos mal classificados. Cada novo modelo é criado de modo a classificar bem os exemplos mal classificados pelos modelos anteriores (Meir; Rätsch, 2003). Em geral, o *boosting* reduz o erro de *bias* e cria modelos preditivos robustos. O algoritmo *AdaBoost* (Freund; Schapire, 1995) foi o primeiro algoritmo prático de *boosting* e continua sendo um dos mais amplamente utilizados e estudados, com aplicações em vários campos.

É possível combinar classificadores de aprendizado de máquina conceitualmente diferentes e usar mecanismo de votação para gerar a classificação final. Nesse esquema, todos os classificadores são treinados com o conjunto de dados completo, e as suas predições são combinadas através de votação. Os tipos de votações que podem ser empregadas são a votação *hard* e a *soft*. Na votação *hard*, cada técnica de classificação individual vota em uma classe, e a classificação final é determinada pela classe que obteve a maioria dos votos. A outra estratégia consiste em um combinador baseado na máxima soma das probabilidades das previsões. De modo simplificado, cada técnica de classificação individual fornece um valor de probabilidade de que a instância pertença a uma determinada classe. As previsões são então somadas, e a classe com a maior soma de probabilidades é definida como a classificação final.

A técnica *Stacking* também se mostra interessante. Ela consiste em um método *ensemble* que combina vários algoritmos de aprendizado de máquina por meio de um metamodelo. Os diversos algoritmos de nível básico são treinados com um conjunto de dados de treinamento completo, e o metamodelo é treinado com os resultados finais dos modelos de nível básico. Em outras palavras, as previsões feitas pelos modelos básicos servem como entrada para o metamodelo. Desse modo, o metamodelo é responsável por aprender a combinar os resultados individuais de cada classificador base em um resultado geral final (Kumar; Gupta; Tripathi, 2021b).

A seguir são aprentados alguns métodos de aprendizado de máquina baseados em *Ensemble Learning*.

2.6.1.7 Random Forest

O algoritmo *Random Forest* (RF) é um método de aprendizado do tipo *ensemble* baseado em *Decision Tree* (DT) criado para diminuir o Sobreajuste *(Overfitting)*. Ela constrói uma "floresta" de árvores de decisão não correlacionadas e combina os resultados das mesmas produzindo os resultados finais de classificação.

Uma das características principais da RF é o emprego de um grau de aleatoriedade na seleção dos atributos a serem consideradas para a divisão. Diferentemente da DT, que aplica métricas de impureza em todo o conjunto de atributos para descobrir a melhor, a RF aplica

estas métricas somente em um subconjunto de atributos candidatos selecionados aleatoriamente. Além disso, utiliza apenas um subconjunto dos dados de treinamento, com reposição, para a construção de cada árvore da estrutura. O algoritmo procura pelos atributos que geram uma melhor separabilidade em cada nodo de cada árvore. Ela seleciona aleatoriamente um conjunto de c atributos candidatos e aplica as medidas de impureza buscando encontrar o melhor ponto de corte de cada atributo e o melhor atributo dentre os c. Esse processo garante que cada árvore gera um modelo diferente. Após o treinamento da RF, a estrutura está apta para realizar a classificação de novos dados. Cada árvore gerada irá realizar a classificação do registro e os seus resultados serão combinados através da média ou de votação majoritária. O método RF possui alguns parâmetros importantes. O primeiro parâmetro é o c, ele indica o número de atributos selecionados aleatoriamente em cada nó. O segundo parâmetro é m, que representa o tamanho mínimo do conjunto de treinamento para dividir um nó. Por fim, o terceiro parâmetro é o a, o qual corresponde ao número de árvores no conjunto. Todos os três parâmetros desempenham um papel significativo. Onde c é responsável pela intensidade do procedimento de seleção de recursos, m a força do ruído de saída da média e a a força da redução da variância da agregação do modelo de conjunto (Geurts; Ernst; Wehenkel, 2006). Além disso, são utilizados alguns parâmetros relacionados com as estruturas das árvores de decisão que compõem a estrutura RF. Um dos parâmetros é o critério de escolha dos melhores atributos, onde podem ser utilizadas as métricas citadas anteriormente: *Gini Index* e *Information Gain*. O parâmetro p indica a profundidade máxima permitida da árvore, ele é utilizado para controlar esse crescimento, pois normalmente elas podem crescer até que todas as folhas sejam puras ou até que todas as folhas contenham menos de m amostras, no entanto, isso pode gerar árvores extremamente longas e custosas. O (f) indica o número mínimo de amostras necessárias para formar um nó folha, um ponto de divisão em qualquer profundidade só será considerado se deixar pelo menos (f) amostras em cada um dos ramos esquerdo e direito.

2.6.1.8 Extra Tree

Os classificadores *Extra Tree* (ET) (Geurts; Ernst; Wehenkel, 2006) são ferramentas importantes em tarefas de classificação. Assim como a RF, a *Extra Tree* consiste em um método *ensemble* que agrega os resultados de várias DTs descorrelacionadas acumuladas em uma "floresta" para produzir os resultados da classificação.

O ET se concentra em aleatorizar fortemente tanto a escolha dos atributos quanto o ponto de corte enquanto divide um nodo na árvore. Portanto, em cada nodo intermediário, uma amostra aleatória de c recursos do conjunto de recursos é selecionada. Cada árvore de decisão deve selecionar o melhor recurso para dividir os dados com base em alguns critérios matemáticos, geralmente o índice de Gini. No caso extremo, ele constrói árvores totalmente aleatórias, cujas estruturas são independentes dos valores de saída da amostra de aprendizado (Geurts; Ernst; Wehenkel, 2006). As árvores de previsão são agregadas para produzir a previsão final, por maioria de votos em problemas de classificação e média aritmética em problemas de

regressão (Geurts; Ernst; Wehenkel, 2006). É bastante semelhante em operação à RF e varia principalmente na forma de construir os DTs dentro da floresta (Kaur; Mittal, 2020). Em ET, a aleatoriedade vai um passo adiante na forma como as divisões são calculadas. Os pontos de corte são sorteados aleatoriamente para cada atributo candidato e o melhor desses pontos de corte gerados é escolhido aleatoriamente como o regra de divisão.

A lógica por trás do método é que o corte explícito e a randomização de atributos combinados com a média do *ensemble* devem conseguir reduzir a variância mais fortemente do que os esquemas de randomização mais fracos usados por outros métodos. O uso de dados de treinamento originais e completos, em vez de réplicas *bootstrap*, é motivado para minimizar o viés (Verma; Ranga, 2020).

A ET também possui alguns parâmetros importantes. O primeiro parâmetro é c indica o número de atributos selecionados aleatoriamente em cada nó. O segundo parâmetro é m, que representa o tamanho mínimo do conjunto de treinamento para dividir um nó e o terceiro parâmetro é o a, o qual corresponde ao número de árvores no conjunto. Todos os três parâmetros desempenham um papel significativo. Onde c é responsável pela intensidade do procedimento de seleção de recursos, m a força da média do ruído de saída e a a força da redução de variância da agregação do modelo *ensemble* (Geurts; Ernst; Wehenkel, 2006). Além destes, o método também trabalha com os parâmetros das árvores e decisão internas, assim como a técnica RF.

2.6.2 Técnicas de seleção de atributos

A utilização de atributos irrelevantes e redundantes pode interferir negativamente no desempenho dos classificadores em relação às métricas de classificação e ao custo computacional (Zhou et al., 2020). Os ruídos possuem a capacidade de interferir no resultado da análise.

O processo de seleção de atributos busca a otimização do uso dos recursos de dados disponíveis. Busca-se encontrar o melhor subconjunto de atributos, conforme o critério estabelecido. Desse modo, é possível tornar o classificador computacionalmente mais rápido e eficiente.

Características significativas sobre classes de dados, padrões intrusivos e combinações de atributos podem ser revelados. Além disso, padrões redundantes, atributos de falsa correlação ou atributos irrelevantes podem ser removidos. Desse modo, permite que os classificadores lidem com um menor e mais relevante conjunto de atributos (Zainal et al., 2009).

Existem diversas técnicas de seleção de atributos na literatura e elas podem ser agrupadas nas categorias: *Filter*, *Embedded* e *Wrapper*. Os métodos *Filter* são baseados em métricas estatísticas. Eles são independentes do classificador. Tais métricas são calculadas aplicando o algoritmo e buscam remover atributos irrelevantes antes da indução ocorrer. A avaliação de cada atributo é realizada individualmente baseada em sua correlação com a função alvo. Após isto então seleciona-se os k atributos com os maiores valores. Os métodos *Embedded* são caracterizados pela realização da seleção de atributos de forma dinâmica, selecionando um subconjunto de atributos no próprio processo de construção do modelo de classificação. Por fim, os méto-

dos *Wrapper* também operam com a utilização de um modelo de classificação internamente, no entanto, eles trabalham com subconjuntos de atributos gerados a cada iteração do algoritmo. O classificador é executado com o subconjunto de atributos gerando uma precisão do classificador, utilizada para avaliar a qualidade do subconjunto (Baranauskas et al., 2002).

A seguir são apresentadas algumas abordagens clássicas de seleção de atributos.

2.6.2.1 Information Gain (IG)

Uma das maneiras para se medir a qualidade de um atributo é avaliar o seu grau de associação com a classe através da medida do ganho de informação, em inglês *Information Gain*, um dos algoritmos *filter* mais conhecido e utilizado na literatura. Ele avalia o grau de associação dos atributos com a classe para encontrar os valores com o maior grau de utilização e importância através do cálculo da redução da entropia. Quanto maior a entropia, maior o grau de impureza. O ganho de informação indica a redução da entropia. Desse modo, os atributos que possuírem o maior ganho de informação serão os mais úteis para o processo de detecção.

Considerando um conjunto de dados $D(A_1,A_2,...,A_n,C)$, com $n >= 1$, onde C é o rótulo e o seu domínio é $(c_1,c_2,...,c_m)$, com $m >= 2$. Segundo Quinlan (1986), a medida do ganho de informação é baseado no conceito de entropia. Entropia é uma medida de impureza e falta de homogeneidade de um atributo. A fórmula apresentada na Equação 2.3 corresponde ao cálculo da entropia para um atributo A, cujo domínio é $(a_1,a_2,...,a_k)$, com $k >= 1$. Os valores p_i, com $1 <= i <= k$, correspondem a razão entre o número de instâncias da base em que ocorre o valor a_i para o atributo A e o número total de instâncias.

$$Entropia(A) = \sum_{i=1}^{m} p_i \log_2(p_i) \tag{2.3}$$

A entropia da classe C, representada por $Entropia(C)$, pode ser calculada da mesma forma, considerando p_j a razão entre o número de instâncias em que o valor c_j da classe, com $1 <= j <= m$, ocorre na base e o número total de instâncias.

O ganho de informação de um atributo é determinado pela redução da entropia (Karegowda; Manjunath; Jayaram, 2010). A Equação 2.4 calcula o ganho de informação de um atributo A em relação à classe C. Onde $Entropia(C)$ é a entropia da classe C, e $Entropia(C,A)$ é entropia da classe relativa ao atributo A, calculada pela Equação 2.5.

$$Ganho(C,A) = Entropia(C) - Entropia(C,A) \tag{2.4}$$

A $Entropia(C|A = A_j)$ é a entropia relativa ao subconjunto de instâncias com um valor A_j para o atributo A. Se A é um bom descritor para a classe, cada valor de A terá uma baixa entropia distribuída entre as classes, ou seja, cada valor deve estar predominantemente em uma classe.

$$Entropia(C,A) = -\sum_{j=1}^{m} p(A,C) \log_2(p(A,C)) \tag{2.5}$$

Tem-se que o valor do ganho de informação consiste na diminuição da impureza, ou seja, atributos com menor entropia possuem maior ganho de informação. Esta medida tem um *bias* natural, pois favorece atributos que possuem muitos valores. A medida de razão de ganho de informação, cuja fórmula é apresentada na Equação 2.6, tenta corrigir o *bias* do Ganho de Informação (Quinlan, 1993).

$$Razão\ \ de\ \ Ganho(C,A) = \frac{Ganho(C,A)}{Entropia(A)} = \frac{Entropia(C) - Entropia(C,A)}{Entropia(A)} \tag{2.6}$$

Após o cálculo de cada atributo pela Equação 2.6, é gerado um ranking e seleciona-se os *N* melhores atributos desse ranking de acordo um *threshold* definido para o corte.

2.6.2.2 Sequential Feature Selector (SFS)

O método *Sequential Feature Selector (SFS)* pertence à família *Wrapper* e é utilizado para a seleção de atributos. Ele adiciona atributos de forma gananciosa (seleção *forward*) ou remove atributos (seleção *backward*) para formar um subconjunto, fazendo uso de um algoritmo de aprendizado de máquina. Durante o processo de seleção de atributos, a qualidade de cada subconjunto é avaliada por meio da precisão do classificador gerado pelo algoritmo ao utilizar aquele subconjunto de atributos. O procedimento é repetido para cada subconjunto de atributos até que o critério de parada seja atingido.

A estratégia de busca *forward* refere-se a uma abordagem de busca que parte de um conjunto inicial vazio. Consiste em um procedimento ganancioso que, de forma iterativa, identifica o melhor novo recurso a ser adicionado ao conjunto de recursos selecionados. Inicialmente, o conjunto de recursos é vazio, e busca-se encontrar o recurso que maximiza a precisão do classificador quando treinado exclusivamente com esse único recurso. Após selecionar o primeiro recurso, o procedimento é repetido, adicionando um novo recurso ao conjunto de recursos selecionados. O processo continua até que o número desejado de recursos selecionados seja alcançado (Ferri et al., 1994).

Por outro lado, a estratégia de busca *backward* possui como estado inicial o conjunto total de atributos do conjunto de dados e visa encontrar o atributo menos relevante para o processo de classificação, o qual é removido do conjunto de dados. Nesse processo, cada atributo do conjunto é avaliado, e um subconjunto é gerado sem o atributo em questão, o qual é utilizado para treinar o classificador e avaliar a precisão alcançada. O atributo considerado menos relevante é, então, removido, e o procedimento é repetido até que o número desejado de recursos selecionados seja alcançado (Ferri et al., 1994).

2.6.2.3 Recursive Feature Elimination (RFE)

O método *Recursive Feature Elimination (RFE)* (Guyon et al., 2002) é também um algoritmo da família *Wrapper* utilizado para seleção de atributos. Esse método emprega um es-

timador externo capaz de atribuir pesos aos recursos, e seu objetivo é selecionar recursivamente conjuntos cada vez menores de atributos.

Inicialmente, o estimador é treinado utilizando o conjunto completo de atributos, e a importância de cada recurso é obtida. Os recursos são pontuados utilizando um modelo de aprendizado de máquina, onde alguns algoritmos, como árvores de decisão, são capazes de fornecer pontuações de importância. Em seguida, os recursos menos importantes são removidos do conjunto atual de atributos. Esse procedimento é repetido de forma recursiva até que o número desejado de atributos a serem selecionados seja alcançado.

É importante destacar que o número de atributos a serem selecionados é um parâmetro do método RFE, e não há um número ideal padrão, o que torna a definição desse parâmetro um desafio.

2.7 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os conceitos básicos envolvidos na temática do trabalho. Com o objetivo de obter uma visão abrangente do atual estado da arte, foi realizada uma revisão sistemática da literatura e os principais trabalhos relacionados são apresentados no próximo capítulo.

3 ESTADO DA ARTE

Nesta seção, é abordado o estado da arte sobre a detecção e prevenção de intrusão na computação em nevoeiro (*Fog Computing*) e em *Internet of Things* (IoT). Para construir uma boa perspectiva sobre o estado da arte foi realizada uma Revisão Sistemática da Literatura (RSL) sobre o tema.

É importante conhecer o atual estado da arte em detecção e prevenção de intrusão em computação em nevoeiro para proteger rede de sensores IoT. Conhecer à quais ataques estes ambientes estão vulneráveis e quais as soluções existentes, permite o desenvolvimento de novos mecanismos para tornar este ambiente seguro. Para isso, uma revisão da literatura é necessária. Optamos por um RSL, pois, ela sintetiza o trabalho existente de maneira justa e é realizada de acordo com uma estratégia de pesquisa predefinida. A estratégia de pesquisa deve permitir que a integridade seja avaliada. Os principais artigos obtidos durante a revisão são abordados e comparados neste capítulo. Todos os detalhes a respeito da definição, execução e resultados da RSL são apresentados no apêndice B.

O processo de seleção de estudos é uma etapa importante na revisão sistemática. Seu objetivo é obter um conjunto de artigos relevantes para o mapeamento. A Figura 4 ilustra o processo completo de seleção do estudo empregado neste trabalho.

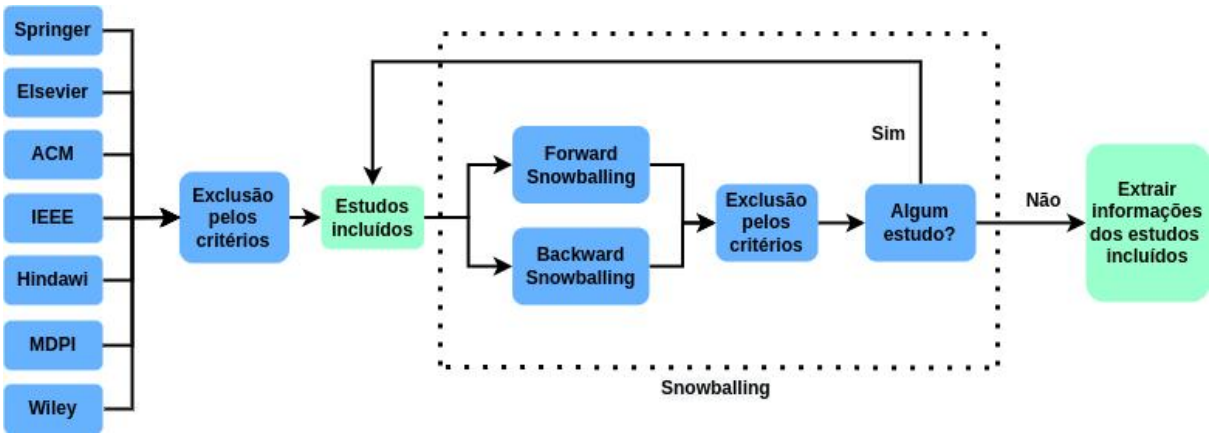


Figura 4 – Descrição do processo de busca de trabalhos.

Nesta revisão sistemática foram pesquisados trabalhos publicados em língua inglesa, a partir de 2010, disponíveis *on-line*. A busca automática foi realizada nas bases de dados ACM [1], IEEE [2], ScienceDirect [3], Springer [4], *Hindawi* [5], *Wiley* [6] e MDPI [7] com a *string* de busca *"Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR*

[1] https://dl.acm.org/
[2] https://ieeexplore.ieee.org/Xplore/home.jsp
[3] https://www.sciencedirect.com/
[4] https://link.springer.com/
[5] https://www.hindawi.com/
[6] https://onlinelibrary.wiley.com/
[7] https://www.mdpi.com/

("Fog Computing" OR Fog)). Os trabalhos duplicados são excluídos e os critérios de exclusão definidos na Seção B.3.4, são aplicados. Os artigos selecionados após este processo são incluídos no mapeamento final. Além disso, é realizado o processo *Snowballing*, que busca artigos relacionados aos estudos incluídos. As buscas são realizadas nas referências dos artigos selecionados (fase *backward*) e nos trabalhos que citam os artigos selecionados (fase *forward*). Após esse processo, se novos artigos forem incluídos, o processo *Snowballing* é reiniciado com esses novos estudos. Caso não haja novo estudo incluído, o processo seletivo é encerrado.

Com a busca nas bases de dados e a execução de 3 iterações do *snowballing* foram encontrados ao todo 10746 trabalhos, os quais foram incluídos e excluídos de acordo com diversos critérios, maiores detalhes a respeito deste processo podem ser conferidos no Apêndice B. Após a terceira iteração do *snoballing* nenhum novo artigo foi incluído e o processo geral de seleção foi encerrado resultando em um conjunto de 108 estudos selecionados ao longo da execução.

Os 108 artigos resultantes do processo de execução do protocolo e incluídos no mapeamento foram inspecionados de modo a obter as informações necessárias para responder as perguntas de pesquisa definidas na Seção B.3.1. Os resultados obtidos e as respostas as perguntas de pesquisa também são apresentados no Apêndice B. Além disso, a seguir alguns trabalhos correlatos são abordados e comparados para construir uma visão geral a respeito de detecção de intrusão em IoT e *Fog Computing*.

3.1 TRABALHOS CORRELATOS

Infelizmente, os dispositivos que compõe uma rede IoT são restritos quanto a recursos e são suscetíveis a falhas e ataques de entidades mal-intencionadas que podem corromper à integridade dos dados (Neshenko et al., 2019). Isso pode levar a falta de confiabilidade, e às vezes ao colapso do sistema. Um dos principais objetivos de um ataque a uma rede de IoT é interromper a disponibilidade de dados enviados dos dispositivos IoT para os aplicativos.

Conforme mencionado na Seção 2.5 (Página 44), os sistemas de detecção de intrusão (*Intrusion Detection System* - IDS) tradicionais analisam informações da rede e de dispositivos para identificar atividades maliciosas. No entanto, sua implantação em IoT é dificultada pelas restrições que os dispositivos deste ambiente possuem. O local de implantação da solução de detecção é uma questão séria que deve ser considerada ao projetar a abordagem, seja baseada em rede ou baseada em *host*.

3.1.1 Local de implantação no contexto IoT

As restrições de recursos, existentes nos dispositivos inseridos no contexto IoT, dificultam a implantação de abordagens de detecção nos próprios dispositivos IoT. Apesar disso, no estado da arte são encontradas abordagens de detecção de intrusão que buscam realizar a análise nos próprios dispositivos IoT e geralmente estão focadas em detectar ataques de rote-

amento através de técnicas leves. Pois, ter uma visão interna é uma vantagem para detectar inconsistências no roteamento (Raza; Wallgren; Voigt, 2013; Shreenivas; Raza; Voigt, 2017; Khan; Herrmann, 2017; Choudhary; Kesswani, 2018). Estas abordagens detectam com eficácia os ataques que provocam inconsistências no roteamento da rede IoT. No entanto, as limitações de recursos impedem a implantação de soluções complexas nestes dispositivos. Alguns trabalhos propuseram abordagens implementadas parcialmente em dispositivos IoT (Arshad et al., 2019), através do emprego de técnicas mais leves baseadas em assinatura. Outras abordagens propuseram soluções mais complexas para operar inteiramente na camada do nevoeiro (Diro; Chilamkurti, 2018b; Shafi et al., 2018; Xu; Qian; Hu, 2019; Almiani et al., 2020). No entanto, o tempo e custo de processamento de métodos de análise robustos pode ser alto, especialmente quando se considera estratégias que utilizam diversos métodos de detecção combinados. Isso pode ser um problema para a implementação de uma abordagem robusta nos dispositivos IoT e até mesmo na camada de computação em nevoeiro, pois, o dispositivo pode ser sobrecarregado e atrasar o fluxo da rede. Desse modo, alguns trabalhos delegaram parte da análise para a camada de computação em nuvem (Nguyen et al., 2019; Illy et al., 2019). Esta camada possui maior capacidade de recursos, mas possui a desvantagem da latência causada pela distância da rede IoT e dos data centers.

A implantação no nível da computação em nevoeiro evita o problema da latência e permite implantar uma estratégia distribuída nos dispositivos do nevoeiro, buscando reduzir a quantidade de tráfego monitorado e aumentar a capacidade de processamento. A grande maioria das abordagens encontradas no estado da arte foram propostas para operar na camada de computação em nevoeiro, que possui dispositivos mais próximos do sistema físico e fornece recursos de processamento e armazenamento na borda da rede tornando possível detectar ameaças em ritmo mais rápido. A computação em nevoeiro atua como uma camada intermediaria e os seus dispositivos estão inseridos próximos aos de IoT, podendo atuar como um *gateway* entre a rede de IoT sem fio e a rede cabeada (Bonomi et al., 2012). Como a computação em nevoeiro está situada em uma posição intermediária entre os dispositivos IoT e as aplicações na Internet, pode ser uma interessante aliada para executar abordagens de monitoramento do tráfego interno e externo à rede IoT, no entanto, é necessário considerar o consumo de recursos das abordagens de detecção nesse contexto. Desse modo, a colaboração entre dispositivos de IoT e as camadas superiores como nevoeiro e nuvem, considerando os níveis local e global de monitoramento, pode permitir considerar o conhecimento agregado de uma coleção de dispositivos *host* e de rede buscando obter uma precisão de detecção aprimorada em tempo hábil.

3.1.2 Abordagens de análise e detecção

As abordagens de detecção podem ser organizadas de acordo com o tipo de análise empregada para detectar as intrusões. Desse modo, a seguir são discutidos estudos do estado da arte que propuseram abordagens baseadas em assinatura ou análise por comportamento, está última também conhecida como análise por anomalia.

Na detecção baseada em assinatura, as ações monitoradas são comparadas com eventos intrusivos predefinidos, geralmente armazenados em um banco de dados. Esses padrões são chamados de assinaturas (Northcutt et al., 2001). Alguns estudos propuseram abordagens baseadas no IDS *Snort* (Sohal et al., 2018; Wang et al., 2018), que é uma ferramenta baseada em assinaturas de ataques. Embora as soluções baseadas e assinatura permitam uma detecção rápida e reduzam a ocorrência de alarmes falsos, elas possuem a limitação de não serem capazes de detectar ataques que não possuem assinatura conhecida pelo IDS.

A maioria dos estudos propôs abordagens de detecção baseadas em comportamento, muitas vezes chamada de detecção por anomalia. A detecção de anomalias assume que qualquer atividade anômala é necessariamente uma intrusão. Qualquer atividade que não se enquadre nos modelos de comportamento definidos como normais é considerada um ataque. A grande vantagem da técnica de detecção de anomalias é a capacidade de detectar novos ataques e/ou variações dos já conhecidos, uma vez que não é necessário conhecê-los previamente. No entanto, esta técnica tem maior chance de sofrer com problemas relacionados a falsos positivos (Boukerche et al., 2007).

Alguns trabalhos consideram uma ramificação da análise por comportamento chamada de análise por especificação (Mitchell; Chen, 2014). Este tipo de solução emprega regras e limites que definem o comportamento padrão esperado para os componentes monitorados. É semelhante à detecção de anomalias. Ambos detectam intrusões quando o comportamento da rede se desvia do especificado. A principal diferença entre eles é que na análise baseada em especificações, um especialista humano é quem estabelece as regras (Mitchell; Chen, 2014; Zarpelão et al., 2017). Yaseen et al. (2017) propuseram uma abordagem baseada em especificação de limites para detecção de ataques de encaminhamento seletivo em *Wireless Sensor Networks* (WSNs). Em Aliyu, Sheltami e Shakshuki (2018), uma abordagem de detecção em nevoeiro é proposta, baseada em desafio/Resposta. Os autores Potrino, Rango e Santamaria (2019) propuseram abordagem de detecção e mitigação baseada em restrições preventivas e políticas de descarte de pacotes com base em limites nos diferentes tópicos definidos por meio do *Message Queue Telemetry Transport Transport* (MQTT). As grandes desvantagens deste tipo de análise é a especificidade necessária e a necessidade do especialista humano para definir o comportamento esperado do sistema.

As abordagens de detecção por anomalia, por outro lado, geralmente são modeladas com a utilização de técnicas de Aprendizado de Máquina *Machine Learning* (ML). No entanto, conforme supracitado, as restrições de recursos são um desafio para a utilização de técnicas de aprendizado de máquina nos próprios dispositivos IoT.

Existem ameaças contra os dispositivos IoT e até mesmo contra os serviços implantados na camada de nevoeiro e de nuvem que precisam de métodos robustos para serem detectadas. Estas abordagens complexas de detecção muitas vezes não podem ser aplicadas nos dispositivos IoT devido às restrições computacionais dos mesmos (Arshad et al., 2019). Além disso, as abordagens que realizam apenas a análise nos nodos IoT, consideram apenas uma visão restrita dos eventos, sendo, portanto, limitadas em sua capacidade de lidar com ataques

complexos, de vários estágios e distribuídos.

Desse modo, vem crescendo a pesquisa em técnicas de detecção implantadas nos dispositivos da camada de nevoeiro, buscando evitar o problema da latência, implantar uma estratégia distribuída e usufruir da posição privilegiada desses dispositivos. Alguns estudos propuseram abordagens por anomalia que tem como foco detectar o comportamento anômalo do próprio nodo de nevoeiro em que estão inseridas(Pacheco et al., 2020). Khater et al. (2019) propuseram uma abordagem simples e leve baseada em redes neurais *Multi-Layer Perceptron* (MLP). A abordagem monitora chamadas do sistema do próprio dispositivo de nevoeiro para identificar eventos anômalos. Estas abordagens são baseadas em *host* e focam em analisar características do comportamento do nodo de nevoeiro, desse modo, não possuem boa habilidade de detectar o comportamento anômalo dos nodos IoT.

Por fim, muitos trabalhos têm proposto soluções implantadas no nevoeiro para realizar o monitoramento da rede de dispositivos IoT através da análise de pacotes da rede. Esta localização permite ao IDS ter uma visão global da rede IoT, monitorando todo o tráfego (Priyadarshini; Barik, 2022; Almiani et al., 2020; Zahra; Chishti, 2022; Kumar et al., 2022). Essas abordagens de detecção podem ser analisadas de acordo com o seu objetivo, algumas buscam apenas detectar a intrusão (binária) e outras realizar a classificação da intrusão em categorias (multi-classe). A seguir são apresentadas abordagens com estes dois objetivos.

3.1.2.1 Detecção binária

As abordagens de detecção binárias (ataque ou não ataque) não são capazes de identificar o tipo ou categoria de ataque. Conforme apresentado na Tabela 49 do Apêndice B, a maioria dos trabalhos apresentou abordagens com foco em detecção binária. Algumas abordagens focaram em detectar um tipo de ataque específico (Tu et al., 2018; Protogerou et al., 2020). Priyadarshini e Barik (2022) propuseram um mecanismo com redes neurais recorrentes, em inglês *Recurrent Neural Network* (RNN), do tipo *Long Short-Term Memory* (LSTM), para detectar DDoS na camada de nevoeiro. Os autores Maharaja, Iyer e Ye (2019) propuseram a abordagem FOCUS, ela adota uma unidade de análise de tráfego baseada em árvore de decisão (*Decision Tree* - DT) para detectar padrões de trafego anômalos. Farukee et al. (2020) propôs uma abordagem baseada em *Random Forest* (RF) e *Convulotional Neural Network (CNN)* para detecção de DDoS. Em Lawal, Shaikh e Hassan (2021), a abordagem usa um banco de dados que armazena assinaturas de ataques detectados anteriormente e um esquema de detecção baseado em anomalias que usa o algoritmo de classificação *K-Nearest Neighbour* (KNN) para detectar ataques DDoS. No entanto, uma desvantagem existente no KNN é o custo computacional, que pode se tornar alto porque é necessário comparar as novas instâncias com todas as instâncias armazenadas na base do exemplo.

Os trabalhos citados anteriormente se concentraram apenas na detecção de DDoS. Além destas abordagens, alguns trabalhos focaram em realizar a detecção binária buscando detectar qualquer variação do comportamento normal, sem considerar um ataque específico

(Rathore; Park, 2018; An et al., 2018c; Rahman et al., 2020; Ravi; Shalinie, 2020). Os autores Bebortta, Das e Chakravarty (2023) empregam uma ANN no nevoeiro para detectar comportamentos anômalos como intrusões. Diro e Chilamkurti (2018a) propuseram uma abordagem de detecção por anomalia baseada em redes neurais recorrentes, mais precisamente redes LSTM. Além disso, Diro e Chilamkurti (2018b) apresentou uma abordagem de detecção de intrusão por anomalia baseada em redes neurais profundas *Stacked AutoEncoder* (SAE).

Estes métodos de análise supracitados, geram como resultado a classificação dos eventos em normais ou maliciosos, não sendo possível identificar o tipo de ataque. Os métodos que realizam apenas a identificação de que uma intrusão ocorreu, ou seja, detecção binária, não são suficientes para prover segurança eficiente. Pois, é necessário que a abordagem seja capaz de mitigar a invasão de modo que a mesma não obtenha sucesso (Xu; Qian; Hu, 2019). E para isso, é importante fornecer para o módulo de mitigação mais informações a respeito do ataque, como por exemplo uma classificação mais especifica, para que sejam executadas contramedidas mais direcionadas para o determinado tipo da ameaça. Além disso, a classificação do tipo ou categoria do ataque é importante para a tomada de decisão do responsável pela rede.

3.1.2.2 Detecção multi-classe

Alguns trabalhos apresentaram abordagens para a classificação da intrusão em categorias de ataques. Estas abordagens, que visam classificar o ataque em categorias específicas, são as multiclasses. É essencial classificar o ataque em categorias para que sejam realizadas contramedidas específicas para cada tipo de ameaça. Além disso, a classificação do tipo ou categoria do ataque é importante para o gerenciador da rede. A partir da identificação da categoria de um determinado ataque que ocorre com uma frequência específica, o responsável pela rede pode decidir implementar ações para corrigir a vulnerabilidade utilizada pelo ataque.

As Redes Neurais Artificiais são uma importante técnica para tarefas de classificação, e por isso são amplamente utilizadas em tarefas de detecção e identificação de intrusão (Khater et al., 2019; Thi-Nga et al., 2020; Aliyu et al., 2022; Razaque et al., 2022). Diro e Chilamkurti (2018c) propuseram uma abordagem distribuída baseada em Redes Neurais Profundas (*Deep Neural Networks* - DNN) do tipo *feed-forward* para detectar intrusões em ambiente IoT. A abordagem de detecção é distribuída entre os nodos da camada do nevoeiro. Cada nodo possui um modelo de detecção baseado em DNN. No entanto, a abordagem apresentou dificuldades na classificação de alguns tipos de ataques.

É importante destacar que as redes neurais podem apresentar um custo computacional elevado a medida que o número de camadas ocultas é aumentado. Este custo esta atrelado principalmente a tarefa de treinamento do modelo neural. A redes neurais *Extreme learning machine* (ELM) surgiram para superar essas limitações da DNN. ELM é uma rede neural *feedforward* de camada oculta única que escolhe aleatoriamente os pesos de entrada e o viés da camada oculta sem ajuste e determina os pesos de saída de forma analítica. Esse algoritmo tende a fornecer bom desempenho de generalização em velocidade de aprendizado extremamente rápida. Os

autores Prabavathy, Sundarakantham e Shalinie (2018) propuseram uma técnica de detecção de intrusão implantada no nevoeiro e baseada em *Online Sequential ELM* (OS-ELM). O treinamento e a detecção são realizados no nevoeiro. OS-ELM fornece um modelo de aprendizado rápido que pode se adaptar a novos dados de dispositivos IoT rapidamente, juntamente com um bom poder de generalização. No entanto, a abordagem apresentou dificuldades na detecção de alguns tipos de ataques durante os experimentos com o conjunto de dados NSL-KDD. De modo geral, as redes ELM são menos precisas do que as redes tradicionais, mas são interessantes para um retreinamento em tempo real.

Além das tradicionais redes *feed-forward*, alguns trabalhos estudaram a utilização de redes neurais recorrentes, um tipo de rede neural que é projetada para modelar dados de sequência. Elas podem usar seu estado interno de memória para processar sequências de comprimento variável de entradas (Ahmad et al., 2021). Shafi et al. (2018) destacam que RNN detecta com alta precisão ataques de alta intensidade, mas não tem um bom desempenho em baixa intensidade. Almiani et al. (2020) propuseram uma abordagem que realiza a classificação dos eventos através de um método baseado em RNN profunda. No processo de treinamento do modelo neural é utilizado *Oversampling* para balancear a base de dados e evitar comportamento tendencioso para as classes com maiores exemplos, o objetivo é melhorar acurácia de ataques que possuem menos instâncias de treinamento. A abordagem apresentou boa acurácia geral mas teve dificuldades na detecção de alguns tipos de ataques como *Remote to Local* (R2L) e *User to Root* (U2R), nos experimentos realizados com o conjunto de dados NSL-KDD. As RNNs básicas possuem algumas dificuldades, normalmente são capazes de lidar com sequências de comprimento limitado e sofrem de memória de curto prazo se o comprimento da sequência for longo (Ahmad et al., 2021). Os autores Samy, Yu e Zhang (2020) propuseram uma abordagem de detecção de intrusão baseada em redes recorrentes do tipo LSTM e implantada nos dispositivos do nevoeiro para monitorar o tráfego da rede IoT. LSTM é uma variação de RNN projetada para lidar com problemas de gradiente em expansão e desaparecimento (Samy; Yu; Zhang, 2020). Ela é capaz de lembrar informações por um longo período de tempo. O tráfego da camada de borda é coletado e enviado à camada de nuvem para treinar o modelo LSTM. Em seguida, o modelo treinado é instalado no nevoeiro. A abordagem superou outros métodos de aprendizado profundo em vários conjuntos de dados, no entanto, também apresentou dificuldades na detecção de alguns tipos de ataques. Além disso, as redes LSTM também apresentam um tempo longo para realização do treinamento.

Outro tipo de rede neural abordada no estado da arte são as Redes Neurais Convolucionais, em inglês *Convolutional Neural Network* (CNN) (Abdel-Basset et al., 2021). NG e Selvakumar (2020) propuseram um algoritmo para detectar anomalias no tráfego de IoT baseado em CNN. A abordagem foi adaptada para trabalhar com um vetor e não com matriz, como a CNN tradicional. As CNN são um tipo de rede neural que usa convolução no lugar da multiplicação geral da matriz, em pelo menos uma de suas camadas (Goodfellow et al., 2016). Essa abordagem apresentou algumas dificuldades na identificação de tráfego benigno; apenas 90,5% deste tráfego foi identificado, indicando um número crescente de falsos positivos. As

redes CNN possuem um alto custo computacional, então implementá-los em um contexto com recursos limitados é um desafio (Al-Garadi et al., 2020).

Além das redes neurais baseadas em aprendizado supervisionado, alguns trabalhos apresentaram abordagens não supervisionadas. As redes *AutoEncoder* (AE) são um tipo de rede neural com múltiplas camadas cuja saída alvo é igual à entrada com alguma quantidade de erro de reconstrução (Thi-Nga et al., 2020). Espera-se que, treinando o AE para copiar a entrada para a saída, a representação latente tenha propriedades úteis. Moussa e Alazzawi (2020) propuseram um *Stacked AutoEncoder* modificado que usa um processo gradual de codificação-decodificação para detecção de ataques em aplicações automotivas IoT. A abordagem usa várias camadas ocultas nas laterais do codificador e do decodificador, simetricamente. Ao todo, são 6 camadas ocultas, com 64, 120, 32, 60, 16 e 8 unidades de neurônios. Porém, o modelo apresentou dificuldades relacionadas a falsos positivos nos experimentos com o conjunto de dados NSL-KDD. Em Labiod, Korba e Ghoualmi (2022) foi utilizada uma variação chamada Variational AE (VAE), que consiste em um modelo AE cuja distribuição de codificação é regularizada durante o treinamento para garantir que seu espaço latente tenha boas propriedades. A abordagem proposta por Kumar, Gupta e Tripathi (2021b) apresenta um mecanismo de preservação de privacidade baseado em *Sparse AutoEncoder* que transforma os dados originais em uma nova forma codificada que evita ataques de inferência. Em ataques de inferência, o invasor tenta alterar os dados obtidos do objeto inteligente adicionando medições de dados fictícios. Posteriormente, uma ANN com saída *softmax* é utilizada para a etapa de análise e classificação dos dados. No entanto, se o conjunto de dados de treinamento não for representativo, AEs podem apenas complicar o processo de aprendizagem em vez de representar as características do conjunto de dados (Al-Garadi et al., 2020).

Uma das principais dificuldades das abordagens neurais é o custo para o treinamento do modelo. De acordo com Huang, Zhu e Siew (2006), isso pode ocorrer devido a grande quantidade de neurônios e camadas ocultas, a certa lentidão os algoritmos de aprendizagem baseados em gradiente amplamente usados para treinar redes neurais e ao processo iterativo de ajuste dos parâmetros das redes.

Devido a essas desvantagens, alguns trabalhos apresentaram soluções baseadas em outros métodos. Árvores de Decisão (*Decision Tree* - DT) é um dos algoritmos mais usados em mineração de dados (Rokach, 2016; Kumar; Gupta; Tripathi, 2021b). As DTs classificam os dados usando um conjunto de decisões de recursos hierárquicos. Peng et al. (2018) investigou o algoritmo de árvore de decisão denominado CART, usando o critério de Gini (Breiman et al., 1984), para detectar intrusão no nevoeiro, a abordagem obteve resultados superiores aos modelos KNN e bayesianos em todas as classes de ataques do conjunto de dados KDDCUP99. No entanto, este conjunto de dados não possui tráfego IoT característico e é bastante antigo, portanto, a abordagem deve ser avaliada com um conjunto de dados mais atual. Além disso, os problemas de *overfitting* são um dos desafios em DT (T.K.; Annavarapu; Bablani, 2021).

Outra técnica comumente utilizada em abordagens de detecção é a *Suport Vector Machine* (SVM). Os autores Xu, Qian e Hu (2019) propuseram uma abordagem para detecção

multi-classe de ataques em infraestrutura de nevoeiro. Como método de classificação propuseram o uso de SVM baseada em *One Class Classification* (OCC). A abordagem é capaz de alcançar bons resultados, no entanto, somente tratou de ataques DoS e de sondagem. A abordagem proposta por Du et al. (2020) é baseada em SVM com parâmetros otimizados pelo algoritmo *Particle Swarm Optimization* (PSO) para obter o classificador de detecção de intrusão SVM ideal. Os experimentos com o conjunto de dados KDD CUP 99 indicaram uma precisão de 98,5% para a abordagem, no entanto, com uma baixa taxa de detecção para ataques U2R. Além disso, a abordagem *SeArch*, proposta pelos autores Nguyen et al. (2019), é composta por um arranjo hierárquico de três camadas de nodos IDS inteligentes. Estas camadas trabalham em colaboração para detectar anomalias. Na camada de computação de borda um mecanismo de detecção de intrusão que reside em um *gateway* IoT realiza análise através de SVM. Os padrões de tráfego desconhecidos são enviados para o nevoeiro, onde, são aplicados métodos mais robustos de aprendizado de máquina, neste caso redes *Self Organizing Map* (SOM). Além disso, os demais padrões de tráfego desconhecidos são enviados para a nuvem onde são realizadas análises com SAE. *SeArch* apresenta métodos para a detecção de ataques de sondagem e DoS, alcançando uma acurácia regular.

A técnica *Naïve Bayes* (NB) é um método de classificação baseado na probabilidade condicional. Ela não foi aplicada diretamente, pois, apresenta acurácia menor que outros métodos (Prabavathy; Sundarakantham; Shalinie, 2018; Souza et al., 2020). No entanto, Manimurugan (2021) propôs o *Improved Naïve Bayes* (INB), um classificador *Naïve Bayes* em conjunto com a técnica *Principal Component Analysis* (PCA), para detecção de intrusão. A abordagem apresentou resultados interessantes em experimentos com o conjunto de dados UNSW-NB15 em relação a outras técnicas, no entanto, apresentou dificuldades na detecção de alguns ataques e problemas de falsos positivos.

Conforme relatado, a classificação de um evento do tráfego de rede em uma categoria de ataque é um problema multi-classe, vários métodos de aprendizado de máquina podem ser aplicados. Para resolver problemas multi-classe é possível utilizar classificadores únicos que são capazes de realizar classificação multiclasse diretamente. No entanto, os classificadores únicos podem sofrer com instabilidades. Não há garantia que um classificador possa obter sempre o melhor desempenho em todas as situações. Desse modo, a seguir são apresentadas e discutidas abordagens *ensembles* que realizam a combinações de diversos classificadores para melhorar o desempenho de detecção.

3.1.2.3 Classificadores Ensemble em detecção de intrusão

Os classificadores baseados em *Ensemble Learning* (EL) são capazes de alcançar um desempenho de classificação melhor do que qualquer um dos classificadores únicos pode alcançar (Samat et al., 2014). A pesquisa em métodos *ensemble* é promissora. Eles podem ser propostos com o objetivo de melhorar a adaptabilidade e capacidade de generalização na classificação multiclasse (Traganitis; Pagès-Zamora; Giannakis, 2018).

No estado da arte, foram encontrados alguns trabalhos que investigaram soluções *ensemble* para detecção de intrusão. Os autores Shafi et al. (2018) propuseram um sistema de detecção e prevenção de intrusão baseado em nevoeiro definido por *software* para redes IoT. Essa arquitetura emprega uma abordagem *ensemble* de votação com três classificadores em paralelo. RNN e MLP classificam o tráfego por maioria de votos. Caso o RNN e o MLP sejam conflitantes, o resultado de uma *Alternate Decision Tree* (ADT) é utilizado como árbitro para definir a classificação final. Em Alhowaide, Alsmadi e Tang (2021) relata duas estratégias principais para combinar os resultados dos classificadores base em métodos *ensemble* por votação: votação *hard* e *soft*. Na votação *hard*, conhecida como votação por maioria, cada classificador individual vota em uma classe e a maioria vence. Na votação *soft*, a classe é prevista com base nas probabilidades previstas dos classificadores básicos. Al-Khafajiy et al. (2021) apontam que a regra de votação por maioria é o esquema de votação mais simples e eficaz neste caso. Os autores Kumar, Gupta e Tripathi (2021b) apresentam uma abordagem baseada na técnica *ensemble* chamada *stacking*. Alrashdi et al. (2019) propuseram um método *ensemble* de OS-ELM e Albdour, Manaseer e Sharieh (2020) utilizaram um *ensemble* de árvores de decisão chamado *Extra Tree*. No entanto estas abordagens trabalharam com detecção binária.

Além disso, algumas abordagens *ensemble* foram propostas com o objetivo de melhorar a detecção multiclasse. Os autores Illy et al. (2019) propuseram uma abordagem com classificador multiclasse *ensemble* baseado em votação, composto por KNN, RF, *bagging* de DTs e *boosting* de DTs. *Bagging* de DTs (Breiman, 1996) é um método que gera várias versões de DT e os usa para obter um classificador agregado. As várias versões são formadas fazendo réplicas de *bootstrap* do conjunto de aprendizagem e usando-os como novos conjuntos de aprendizagem para cada versão. *Boosting* é uma abordagem de *ensemble learning* para reduzir principalmente o viés e também a variância no aprendizado supervisionado. Neste caso, cada classificador trabalha nas limitações dos classificadores anteriores, os classificadores base posteriores são treinados ou testados apenas nos casos em que os classificadores anteriores a ele não foram capazes de atingir uma precisão satisfatória (Schapire, 1990).

Vários trabalhos recentes pesquisaram o uso de *Random Forest* em abordagens de detecção de intrusão (Illy et al., 2019; Kumar; Gupta; Tripathi, 2020; Kumar; Gupta; Tripathi, 2021b; Kumar et al., 2022). RF consiste em um método *ensemble* baseado em árvores de decisão. Kumar et al. (2020) avaliou RF no contexto de detecção de ataques DDOS e descobriu que o fato de RF ser insensível a *outliers*, valores ausentes, *overfitting* e ter a capacidade de lidar com um grande número de tráfego de entrada o torna adequado na anomalia do processo ferramenta de detecção para o ambiente *blockchain*-IoT. Kumar et al. (2022) realizou um estudo com RF para identificar classes de ataques e constatou um excelente desempenho. Hosseini e Sardo (2022) propuseram uma abordagem com classificação de RF e seleção de recursos por *Spider-Monkey Optimization* (SMO). Zhang et al. (2021) realizou diversos experimentos com variações nas quantidades de DTs utilizadas em um RF. Entre os FR com 100, 50 e 10 DTs, a abordagem com 10 DTs apresentou diferenças sutis no desempenho frente a diferentes ataques, porém, há grandes diferenças no tempo de treinamento e teste.

Seguindo a linha de classificadores *ensemble* com DT, os autores Lawal, Shaikh e Hassan (2020) apresentaram uma abordagem com *Extreme Gradient Boosting* (XGBoost) para detecção de anomalias em um *framework* IoT. O *XGBoost*, é uma método *ensemble* baseado na estratégia *Boosting*. Além dos benefícios de velocidade e desempenho do *XGBoost*, as vantagens adicionais são evitar *overfitting* e utilização total dos recursos computacionais (Lawal; Shaikh; Hassan, 2020). Nos experimentos realizados por Lawal, Shaikh e Hassan (2020) com o conjunto de dados Bot-IoT, o *XGBoost* foi capaz de alcançar desempenho superior em comparação com outros algoritmos classificadores, como DT, kNN e NB.

Alguns trabalhos propuseram abordagens híbridas *ensemble* com diversos algoritmos de classificação. Kumar, Gupta e Tripathi (2020) propuseram uma abordagem para detecção de intrusões no nevoeiro. Ela se baseia na técnica *ensemble* chamada *stacking*. Desse modo, considera classificadores simples heterogêneos, os treina em paralelo e os combina de modo a gerar um metamodelo para produzir uma previsão com base nas previsões de diferentes modelos fracos. Neste caso, os classificadores KNN, *XGBoost* e NB são treinados paralelamente e agem como classificadores básicos. A RF é responsável por aprender a combinar os resultados individuais de cada classificador fraco em um resultado geral final. Os experimentos realizados com conjunto de dados demonstraram que a abordagem obteve bom desempenho mas apresentou dificuldades na identificação de alguns ataques. Em Kumar, Tripathi e Gupta (2021) uma abordagem parecida é proposta, porém utilizando um *XGBoost* como metaclassificador.

Os autores Moustafa et al. (2021) propuseram uma abordagem baseada em *Gaussian Mixture-based Correntropy*, um novo modelo estatístico *ensemble* de aprendizado de uma classe. Os Modelos de Mistura Gaussiana (*Gaussian Mixture Models* - GMM), podem ajustar com eficiência os limites legítimos de dados em distribuições multivariadas que consideram todas as variações dos dados legítimos, em vez de usar um conjunto de observações de dados estáticos. A abordagem utiliza o PCA para selecionar os melhores atributos. Ela foi capaz de alcançar boas taxas de detecção com os conjuntos de dados NSL-KDD e UNSW-NB15, em todas as classes de ataques. No entanto, não apresentou as taxas de identificação do tráfego normal, taxas ruins nessa métrica podem indicar problemas de falsos positivos.

Zhao et al. (2022) propõe um sistema híbrido de detecção de intrusão *ensemble* com *stacking* ponderado. Os métodos RF, XGBoost e KNN são usados como classificadores básicos, e a Regressão Logística é selecionada como metaclassificador. Albulayhi et al. (2022) propõe uma abordagem com um método *ensemble* por maioria de votos de quatro classificadores básicos: KNN, árvore de decisão, modelo neural e um método *bagging*. Xu et al. (2023) propôs uma abordagem de *ensemble bagging* com árvores de decisão para realizar a classificação final. Em Lazzarini, Tianfield e Charissis (2023) é proposto um método *ensemble stacking*, o qual utiliza como classificadores base MLP, DNN, CNN e LSTM. Para realizar a combinação das classificações uma DNN é utilizada como metamodelo. Os autores destacam que como trabalhos futuros será necessário pesquisar estratégias para suportar os custos computacionais da abordagem na borda da rede.

É necessário destacar que embora os métodos *ensemble* forneçam maior robustez, eles

geralmente requerem maiores capacidades computacionais, pois trabalham com múltiplos classificadores (Lazzarini; Tianfield; Charissis, 2023).

As pesquisas em detecção multiclasse vem recebendo cada vez mais atenção. Observa-se que a quantidade de abordagens multiclasse vem crescendo ano após ano, provando ser uma área que está em foco atualmente e que possui uma grande importância no contexto de detecção de intrusão. A seguir são apresentados os mecanismos de contramedidas, propostos no estado da arte, para mitigar os ataques.

3.1.3 Contramedidas

Além de detectar as intrusões, é muito importante executar ações de contramedidas, com o objetivo de bloquear e evitar que a intrusão obtenha sucesso. Dentre as ações apresentadas por abordagens do estado da arte estão a emissão de alerta para o gerenciador da rede, descarte de pacotes e bloqueio de conexões. A emissão de apenas um alerta não configura uma ação de prevenção, pois, apenas deixa o gerenciador ciente que ocorreu uma intrusão, mas não a evita (Nguyen et al., 2019; Miranda et al., 2020).

Muitos dos trabalhos propuseram mecanismos para contextos específicos em que estão inseridos, para protocolos ou ataques específicos (Yaseen et al., 2017; Sohal et al., 2018; Maharaja; Iyer; Ye, 2019; Zhou; Guo; Deng, 2019; Potrino; Rango; Santamaria, 2019).

Alguns trabalhos propuseram abordagens que realizam o descarte de pacotes e o bloqueio de conexões quando um ataque é detectado (Pan; Pacheco; Hariri, 2017; Tu et al., 2018; Ravi; Shalinie, 2020; Lawal; Shaikh; Hassan, 2020).

Pacheco et al. (2020) apresentam como contra-medidas o reinício de conexões e sistema e até mesmo alterações de configuração. Os autores Zahra e Chishti (2020) propuseram uma abordagem para responder às intrusões detectadas com base em quão prejudiciais elas podem ser. Essa decisão é tomada por uma base de conhecimento difusa.

Por fim, alguns trabalhos encontrados apresentaram resultados promissores na pesquisa de abordagens baseadas em Redes Definidas por Software (*Software Defined Networks* - SDN) para a mitigação de intrusão em ambientes de computação em nevoeiro e IoT (Shafi et al., 2018; Priyadarshini; Barik, 2022; Nguyen et al., 2019; Miranda et al., 2020). SDN se baseia na separação de planos de controle e planos de dados. Essa separação permite que o controlador seja facilmente modificado e atualizado para alterar todo o comportamento da rede. Essas abordagens utilizam essa característica programável da SDN para implementar ações de contramedidas dinamicamente.

A maioria dos trabalhos não apresenta considerações a respeito de medidas de mitigação de intrusões. No entanto, é essencial para a segurança da rede que exista mecanismos de mitigações ativos para evitar que as intrusões obtenham sucesso.

3.2 PROBLEMAS DE PESQUISA

As abordagens do estado da arte são comparadas e discutidas nesta seção para construir uma visão clara sobre os problemas de pesquisa abordados neste trabalho. A Tabela 1 apresenta uma comparação sobre as abordagens, de acordo, com diversas características. A seguir são apresentadas as abreviações utilizadas na tabela.

- **M**: Realiza análise de detecção **M**ulticlasse;
- **BM**: O trabalho propõem estratégia combinando detecção **B**inária e **M**ulticlasse;
- **EL**: A abordagem de detecção proposta se baseia em ***E**nsemble **L**earning*;
- **BC**: A abordagem proposta utiliza técnicas de **B**alanceamento de **C**lasses para os dados de treinamento;
- **CM**: O trabalho aborda a questão de **C**ontra**M**edidas;

Algumas abordagens encontradas no estado da arte focam em detecção por assinatura, elas não são capazes de detectar novos ataques ou variações de ataques conhecidos pois utilizam assinaturas de ataques conhecidos para realizar a detecção (Sohal et al., 2018; Wang et al., 2018; Arshad et al., 2019). Por outro lado, a detecção por anomalia considera que todo comportamento anômalo é uma intrusão e desse modo consegue detectar novos ataques (Boukerche et al., 2007). Várias abordagens propuseram métodos baseados em anomalia para detectar intrusões (Diro; Chilamkurti, 2018a; Alrashdi et al., 2019; Almiani et al., 2020; Azarkasb; Kashi; Khasteh, 2021; Kumar et al., 2022). Os métodos de aprendizado de máquina são comumente aplicados nesse contexto (Buczak; Guven, 2016).

Conforme supracitado, métodos que realizam apenas a detecção de que uma intrusão ocorreu não são suficientes. É importante obter mais informações sobre o evento intrusivo, como por exemplo, a sua classificação em uma categoria. Isso permite direcionar contramedidas específicas para o determinado tipo de ameaça, de modo que o mecanismo de contramedidas seja capaz de mitigar e evitar que ela tenha sucesso (Xu; Qian; Hu, 2019). Além disso, a classificação do tipo ou categoria do ataque é importante para a tomada de decisão do responsável pela rede.

Os métodos binários apenas detectam que uma intrusão ocorreu, mas não são capazes de fornecer maiores informações. Além disso, observa-se no estado da arte que as abordagens de detecção multiclasse geralmente são mais complexas, possuem maior custo computacional e apresentam taxas de acurácia inferiores em comparação aos métodos binários, pois é um problema mais complexo (Prabavathy; Sundarakantham; Shalinie, 2018; Nguyen et al., 2019; Sarwar et al., 2022; Zhao et al., 2022).

Há uma ampla variedade de modelos básicos de aprendizado de máquina capazes de realizar a detecção multiclasse diretamente. Técnicas como DNN (Diro; Chilamkurti, 2018c;

Tabela 1 – Comparação entre os trabalhos do estado da arte.

Trabalhos	**Método**	**M**	**BM**	**EL**	**BC**	**CM**	**Observações**
Diro e Chilamkurti (2018c)	DNN	✓	-	-	-	-	Baixa detecção para alguns tipos de ataques.
Prabavathy et al. (2018)	OS-ELM	✓	-	-	-	-	Menos precisas do que ANN. Retreinamento em tempo real.
Shafi et al. (2018)	*Ensemble Voting*	-	-	✓	-	✓	Apenas detecção binária. Estratégia de mitigação baseada em SDN.
Potrino et al. (2019)	*Thresholds*	-	-	-	-	✓	Apenas detecção binária. Mitigações apenas para protocolo específico.
Arshad et al. (2019)	Assinaturas	-	-	-	-	-	Incapaz de detectar ataques sem assinatura definida.
Nguyen et al. (2019)	SVM+SOM+SAE	-	-	-	-	✓	Detecção de DDoS. Estratégia de mitigação baseada em SDN.
Samy, Yu e Zhang (2020)	LSTM	✓	-	-	-	-	Baixa detecção para alguns tipos de ataques.
Kumar et al. (2020)	*Ensemble Stacking*	✓	-	✓	-	-	Método pode sobrecarregar o dispositivo.
Almiani et al. (2020)	RNN	✓	-	-	✓	-	Suscetível a problemas em cenários de grave desbalanceamento.
Moussa e Alazzawi (2020)	SAE	✓	-	-	-	-	Apresentou dificuldades em relação a falsos positivos.
Farukee et al. (2020)	RF+CNN	-	-	-	-	-	Custo da seleção de atributos *wrapper*.
Miranda et al. (2020)	Markov Chain + SVM	-	-	-	-	✓	Sem resultados de detecção.
Lawal, Shaikh e Hassan (2021)	KNN	-	-	-	-	-	Custo computacional do algoritmo KNN pode se tornar elevado.
Moustafa et al. (2021)	PCA+GMM	✓	-	✓	-	-	Não apresentou métrica a respeito de falsos positivos.
Kumar et al. (2021)	*Ensemble Stacking*	-	-	✓	-	-	Apenas detecção binária.
Onah et al. (2021)	NB	-	-	-	-	-	Seleção de atributos *wrapper* é custosa.
Qaddoura et al. (2021b)	ANN+LSTM	✓	✓	-	✓	-	Suscetível a problemas em cenários de grave desbalanceamento.
Alhowaide et al. (2021)	*Ensemble Voting*	-	-	✓	-	-	Apenas detecção binária.
Labiod et al. (2022)	VAE+MLP	✓	✓	-	-	-	Baseado neste trabalho. Bons resultados em novos datasets.
Zhao et al. (2022)	*Ensemble Stacking*	✓	-	✓	-	-	Baixa detecção para alguns tipos de ataques.
Priyadarshini e Barik (2022)	LSTM	-	-	-	-	✓	Apenas detecção de DDoS. Estratégia de mitigação baseada em SDN.
Dat-Thinh et al. (2022)	DT	✓	✓	-	-	-	Apresentou dificuldades em relação a falsos positivos.
Xu et al. (2023)	*Ensemble Bagging*	✓	-	✓	✓	-	Suscetível a problemas em cenários de grave desbalanceamento.
Bebortta et al. (2023)	ANN	-	-	-	-	-	Apenas detecção binária.
Lazzarini et al. (2023)	Ensemble Stacking	✓	-	✓	-	-	Método *ensemble* de redes neurais possui custo significativo.
Alotaibi e Ilyas (2023)	Ensemble Stacking	✓	-	✓	-	-	Método *ensemble* possui custo significativo para a borda da rede.
Este trabalho	**DNNET+Ensemble**	✓	✓	✓	✓	✓	Apresenta apenas discussão teórica sobre contramedidas.

Almiani et al., 2020; Razaque et al., 2022), LSTM (Samy; Yu; Zhang, 2020), AE (Kumar; Tripathi, 2021; Labiod; Korba; Ghoualmi, 2022), SVM (Nguyen et al., 2019; Xu; Qian; Hu, 2019), SOM (Nguyen et al., 2019), têm sido utilizadas no estado da arte. No entanto, esses estudos apresentaram alguns problemas, como, por exemplo, dificuldades na identificação de tipos específicos de ataques (Diro; Chilamkurti, 2018c; Almiani et al., 2020; Du et al., 2020; Abdel-Basset et al., 2021). Além disso, algumas abordagens apresentam problemas relacionados à identificação normal do tráfego, gerando muitos falsos positivos (Ieracitano et al., 2020; Dat-Thinh; Xuan-Ninh; Kim-Hung, 2022). Abordagens com problemas de falsos positivos podem degradar o desempenho da rede.

As abordagens que funcionam com classificadores únicos podem sofrer instabilidades. Não há garantia de que um classificador sempre terá o melhor desempenho em todas as situações. No entanto, com *Ensemble Learning*, pode ser alcançado um melhor desempenho de classificação do que qualquer classificador único (Samat et al., 2014). Estratégias *ensemble* podem ser propostas para melhorar a adaptabilidade e generalização na classificação multiclasse (Traganitis; Pagès-Zamora; Giannakis, 2018). A pesquisa sobre métodos de *ensemble* é promissora, no estado da arte, encontramos alguns trabalhos que investigaram soluções de *ensemble* para detecção binária (Shafi et al., 2018; Alrashdi et al., 2019; Albdour; Manaseer; Sharieh, 2020; Kumar; Gupta; Tripathi, 2021b). Além disso, alguns trabalhos propuseram abordagens *ensemble* robustas para detecção multiclasses (Lawal; Shaikh; Hassan, 2020; Kumar; Gupta; Tripathi, 2020; Moustafa et al., 2021). A inserção de redundância de classificador com métodos de *ensemble* pode ser interessante também para diminuir a variação de classificadores individuais. Desse modo, combinar diferentes modelos de aprendizado de máquina para obter desempenho otimizado e detecção de ataques é outra tendência de pesquisa (Al-Garadi et al., 2020). No entanto, métodos *ensemble* possuem maior complexidade computacional e demandam mais recursos de computação, desse modo, é importante levar em consideração, no projeto de métodos de detecção, as características dos dispositivos onde a abordagem será implantada. Os dispositivos do nevoeiro, apesar de serem mais poderosos que os dispositivos IoT, não possuem recursos ilimitados como a computação em nuvem. Desse modo, um método ensemble muito complexo, com redes neurais por exemplo, pode sobrecarregar o dispositivo do nevoeiro, principalmente durante o treinamento.

As dificuldades de detecção de ataques estão frequentemente relacionadas ao desequilíbrio dos dados de treinamento existentes. Alguns trabalhos utilizaram a técnica *Synthetic Minority Oversampling Technique* (SMOTE) para balancear os dados (Qaddoura et al., 2021b; Xu et al., 2023). A sobreamostragem é utilizada para equilibrar o banco de dados e evitar comportamento discriminatório para classes com exemplos mais significativos. O objetivo é melhorar a precisão dos ataques que possuem menos instâncias de treinamento. No entanto, a aplicação da estratégia SMOTE completa em cenários extremamente desequilibrados com muitas classes criará um número muito grande de registos sintéticos, o que aumenta o custo do treino e pode degradar o desempenho do modelo de aprendizagem automática (Bowyer et al., 2011).

Outro ponto de pesquisa inclui a escolha dos atributos a serem considerados na análise. Atributos incorretos podem degradar o desempenho do detector. Alguns trabalhos buscaram filtrar as melhores características do tráfego com técnicas de seleção de atributos *wrapper* (Farukee et al., 2020; Onah et al., 2021), onde métodos de classificação são incorporados ao seletor. Comparativamente, os métodos *wrapper* obtêm conjuntos de atributos de qualidade superior para detecção do que os métodos *filter*. Entretanto, abordagens *wrapper* exigem mais processamento e geram custos computacionais mais elevados, o que pode ser proibitivo ao lidar com grandes quantidades de dados.

Portanto, em muitos casos, as técnicas utilizadas nas abordagens de detecção, seleção de atributos e balanceamento de classes podem tornar as abordagens muito onerosas para operar no ambiente Fog-IoT.

O local de implantação da solução de detecção é um problema sério que deve ser considerado ao projetar a abordagem, seja baseada em rede ou baseada em *host*. As restrições de recursos, geralmente existentes nos dispositivos inseridos no contexto de aplicações IoT, dificultam a implantação de abordagens robustas de detecção nos próprios dispositivos. Desse modo, grande parte das abordagens propuseram suas soluções para operar inteiramente na camada de computação de nevoeiro (Diro; Chilamkurti, 2018b; Shafi et al., 2018; Xu; Qian; Hu, 2019; Almiani et al., 2020). No entanto, uma análise multiclasse robusta e lenta realizada no nevoeiro pode sobrecarregar o dispositivo e causar atraso no fluxo da rede (Nguyen et al., 2019). Além disso, realizar unicamente a análise na nuvem possui a desvantagem da latência provocada pela distância da rede IoT e os *datacenters* da nuvem.

O tempo e custo de processamento de modelos robustos de aprendizado de máquina pode ser alto, especialmente quando se consideram estratégias que utilizam diversos métodos de detecção. Além disso, o emprego de técnicas como balanceamento de classes e seleção de atributos pode exigir mais recursos durante o treinamento. Isso pode ser um problema para a implementação de uma abordagem multiclasse robusta na camada de computação em nevoeiro. Portanto, uma das principais lacunas no estado da arte, é a falta de uma abordagem de detecção de anomalias multiclasse, capaz de detectar e identificar o ataque, alcançando taxas de acerto semelhantes ou melhores que a detecção binária, sem sobrecarregar o nevoeiro. Os métodos híbridos e *ensemble* vem ganhando mais destaque por conseguirem melhores resultados, no entanto, possuem um maior custo. Desse modo, é necessário aliar o desempenho de detecção com o custo computacional de abordagens mais robustas ao se projetar a solução de detecção.

Este trabalho propõem uma abordagem de detecção de duas etapas combinando a detecção binária e a detecção multiclasse em uma arquitetura hierárquica. A estratégia é empregar uma análise binária na camada de computação em nevoeiro e uma análise multiclasse na camada de nuvem. Os dispositivos da computação em nevoeiro estão mais próximos da rede IoT e conseguem fornecer respostas em um ritmo mais rápido. No entanto, não possuem recursos ilimitados como a computação em nuvem. Desse modo, apesar das dificuldades relacionadas à distância, a nuvem também pode ser uma importante aliada na tarefa de detecção, devido à grande capacidade de recursos. O objetivo é aproveitar as melhores características

de cada camada. Desse modo, utiliza-se o nevoeiro para fornecer análises e respostas rápidas, principalmente para os eventos benignos, e a nuvem para realizar uma análise mais robusta nos eventos detectados como intrusivos. A concepção dessa estratégia de arquitetura foi inicialmente publicada em Souza et al. (2020) e já foi utilizada e aprimorada em outros trabalhos (Qaddoura et al., 2021b; Labiod; Korba; Ghoualmi, 2022; Dat-Thinh; Xuan-Ninh; Kim-Hung, 2022; Kaffash; Kamel; Kheirabadi, 2023).

Além disso, conforme relatado anteriormente, a maioria das abordagens presentes no estado da arte não apresentam grandes considerações a respeito das contramedidas necessárias para mitigar os ataques detectados nos ambientes IoT. Dentre trabalhos que abordaram a mitigação dos ataques, alguns focaram apenas na mitigação de um tipo específico de ataque ou na mitigação de ataques que possuem como alvo protocolos específicos (Tu et al., 2018; Maharaja; Iyer; Ye, 2019; Pacheco et al., 2020). Apesar dos poucos estudos, alguns apresentaram estratégias promissoras baseadas em SND (Shafi et al., 2018; Nguyen et al., 2019; Miranda et al., 2020). Este trabalho busca fornecer uma breve análise a respeito do atual estado da arte sobre ações de mitigação de ataques no contexto de computação em nevoeiro e IoT. São apresentadas algumas estratégias para mitigação, e a partir disso, é realizada uma discussão sobre alguns pontos importantes ao projetar estratégias de mitigação de intrusões, bem como das próximas etapas em trabalhos futuros.

3.3 CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentada uma discussão a respeito do estado da arte levantado através de uma revisão sistemática realizada para obter uma compreensão a respeito do estado da arte em detecção e prevenção de ataques em ambientes de computação em nevoeiro e IoT. Por fim, foi realizada uma comparação a respeito das principais características envolvidas em cada trabalho e uma discussão a respeito dos problemas de pesquisa. No próximo capítulo, a proposta deste trabalho é descrita em detalhes.

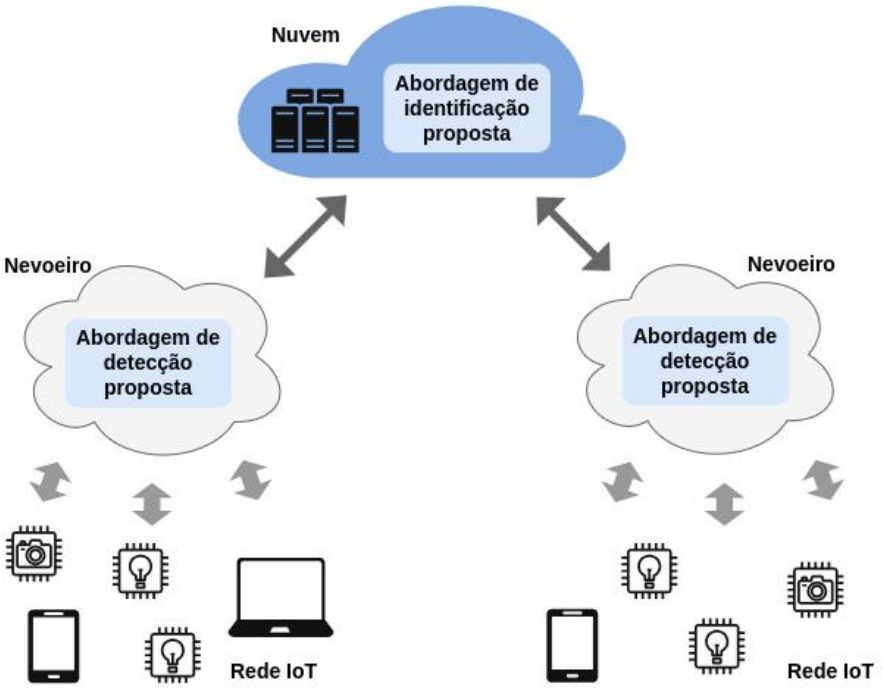
4 ABORDAGEM PROPOSTA

Considerando o estado da arte e os problemas existentes de segurança no ambiente de Internet das Coisas (*Internet of Things* - IoT) e de computação em nevoeiro (*Fog Computing*), verifica-se que as soluções atuais apresentam deficiências e desafios (Seção 3.2, Página 73). Dentro desse contexto, ressalta-se também que a capacidade de obter informações mais precisas sobre um ataque, como a identificação da classe específica à qual ele pertence, pode ser uma importante aliada para garantir um ambiente mais seguro. Isso é essencial tanto para a implementação de contramedidas mais específicas quanto para fornecer informações mais detalhadas na tomada de decisão do gerenciador da rede.

Desse modo, este trabalho propõe uma abordagem híbrida multicamada baseada em anomalia para detecção multiclasse e prevenção de ataques no nevoeiro em ambientes IoT. O objetivo da abordagem é preencher a principal lacuna presente no estado da arte (Seção 3.2, Página 73), que consiste na falta de uma abordagem de detecção e prevenção de intrusão capaz de detectar a categoria do ataque e alcançar alta taxa de precisão sem sobrecarregar os dispositivos.

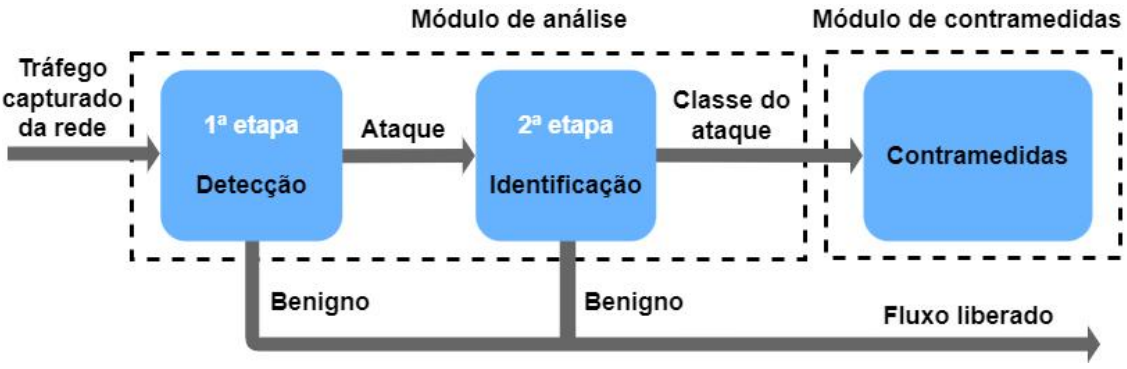
Os dispositivos IoT geralmente possuem recursos computacionais limitados (Ni et al., 2018). Essas limitações dificultam a realização de análises baseadas em técnicas complexas de anomalias diretamente nos dispositivos IoT, impedindo, consequentemente, a detecção de novos ataques (Zarpelão et al., 2017). Além disso, esse tipo de abordagem considera apenas uma visão restrita dos eventos. A computação em nuvem oferece dispositivos com maiores capacidades de computação em comparação com dispositivos IoT e a *Fog* (Ni et al., 2018), no entanto, ela encontra problemas devido à distância considerável entre a rede IoT e os *data centers* da computação em nuvem (Rebbah; Rebbah; Smail, 2017). Essa distância pode provocar problemas de latência para prover detecções de maneira rápida. Apesar de não contar com a mesma capacidade computacional, a computação em nevoeiro é capaz de fornecer processamento e armazenamento na borda da rede (Bonomi et al., 2012). Ela está mais próxima da rede IoT e, como resultado, torna possível detectar ameaças em um ritmo mais rápido. Além disso, os dispositivos de computação em nevoeiro possuem recursos computacionais mais substanciais do que os dispositivos IoT (Ni et al., 2018). Conforme mostrado na Figura 5, a abordagem proposta foi projetada para operar ao longo dessa estrutura *Fog-Cloud*, de modo a realizar no nevoeiro o primeiro passo de análise dos dados capturados do tráfego externo e da rede IoT. Portanto, a abordagem é classificada como *Network Intrusion Detection System* (NIDS).

Figura 5 – Contextualização da abordagem proposta.



A Figura 6 apresenta um esboço a respeito nos módulos necessários para um sistema de detecção e prevenção de intrusão. O primeiro módulo engloba a parte de análise do tráfego, para isso este trabalho propõe uma nova abordagem de detecção e identificação de intrusões. O segundo módulo consiste na ativação de contramedidas para mitigar os ataques detectados, e nessa etapa é realizada uma breve análise a respeito de mitigações de intrusões adequadas para este cenário.

Figura 6 – Esboço a respeito nos módulos necessários para um sistema de detecção e prevenção de intrusão.



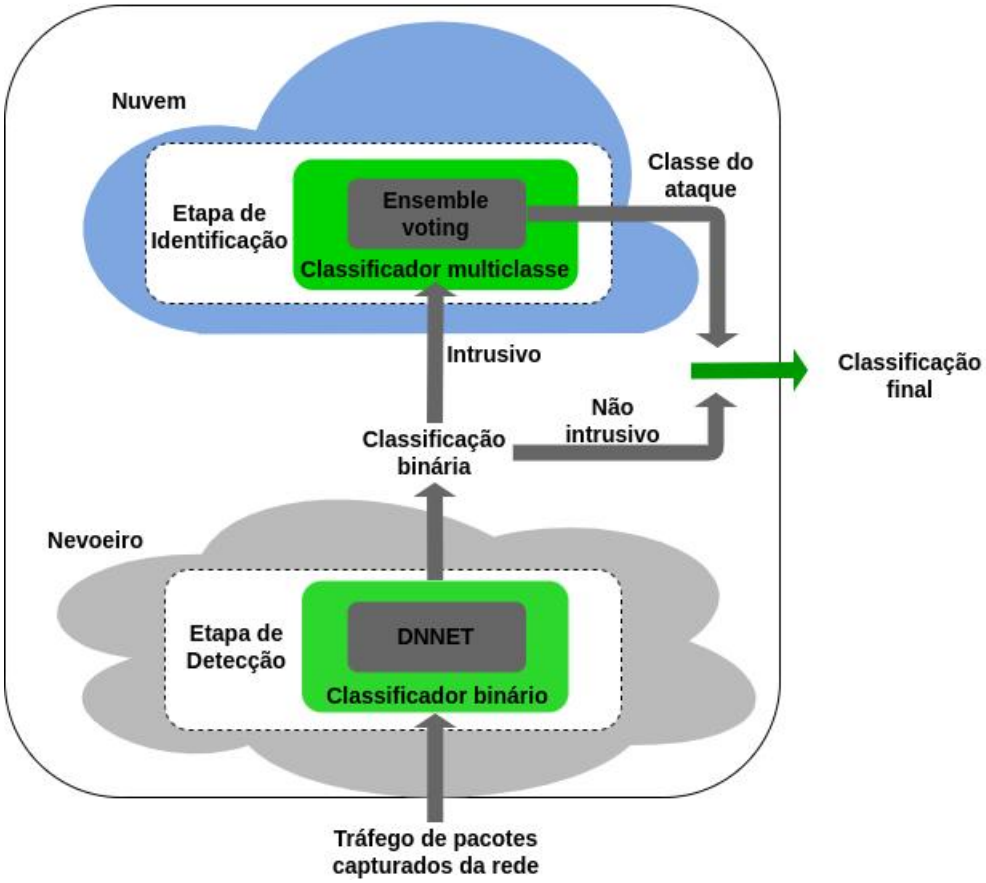
Para este primeiro módulo, é proposto um método hierárquico de detecção em duas etapas. A estratégia é empregar uma análise binária na camada de computação em nevoeiro e uma análise multiclasse na camada de nuvem. Os dispositivos da computação em nevoeiro estão mais próximos da rede IoT e conseguem fornecer respostas em um ritmo mais rápido. No entanto, não possuem recursos ilimitados como a computação em nuvem. Desse modo, apesar das dificuldades relacionadas à distância, a nuvem também pode ser uma importante aliada na tarefa de detecção, devido à grande capacidade de recursos. O objetivo é aproveitar as melhores

características de cada camada. Desse modo, utiliza-se o nevoeiro para fornecer análises e respostas rápidas, principalmente para os eventos benignos, e a nuvem para realizar uma análise mais robusta nos eventos detectados como intrusivos. A concepção dessa estratégia de arquitetura foi inicialmente publicada em Souza et al. (2020) e já foi utilizada e aprimorada em outros trabalhos (Qaddoura et al., 2021b; Labiod; Korba; Ghoualmi, 2022; Dat-Thinh; Xuan-Ninh; Kim-Hung, 2022). A seguir, são apresentados mais detalhes a respeito dessa estratégia.

4.1 MÉTODO HIERÁRQUICO DE DETECÇÃO EM DUAS ETAPAS

Com o objetivo de melhorar a precisão da detecção multiclasse, a abordagem proposta apresenta um método de detecção hierárquico de duas etapas chamado DNNET-Ensemble. Este realiza uma análise binária (Etapa 1 - Detecção) no nevoeiro e análises mais complexas (Etapa 2 - Identificação) para classificação da categoria do ataque na nuvem. A Figura 7 apresenta um esboço do método proposto, considerando as etapas de Detecção e Identificação. O tráfego de pacotes é capturado da rede, e a partir disso, as informações necessárias para a análise binária da primeira etapa de detecção são extraídas.

Figura 7 – Fluxo do método de classificação de duas etapas.



É possível analisar no Algoritmo 1 que na Etapa 1 - Detecção é realizada uma análise de detecção binária, o classificador DNNET proposto é responsável por atribuir a cada evento o rótulo intrusivo ou não intrusivo. Se for tráfego benigno, será permitido automaticamente.

Por outro lado, caso o tráfego seja classificado como intrusivo, ele é enviado para a Etapa de Identificação, na camada de nuvem, onde um classificador multiclasse realiza a identificação da categoria do ataque. Desta forma, informações mais precisas sobre o ataque podem ser obtidas.

Se o evento for detectado como não intrusivo pela Etapa 1 (Detecção), ele será automaticamente enviado para a saída do módulo, para liberar o fluxo. Quando um evento é detectado como intrusivo na primeira etapa, ele é enviado para a Etapa 2 (Identificação), onde um novo processo de análise será realizado, conforme apresentado no Algoritmo 1. Nesse novo processo de análise é empregado um classificador multiclasse Ensemble (Souza; Westphall; Machado, 2022). Este método classificará o evento em tipo de ataque ou benigno. Caso seja identificado como um tipo de ataque, essa informação será enviada ao módulo de mitigação, que será utilizado para implementar contramedidas adequadas a cada intrusão. Contudo, a abordagem de segundo nível pode classificar um evento como benigno, neste caso identificando um evento erroneamente classificado como intrusivo pelo primeiro nível. Desta forma, a abordagem permite a recuperação de falsos positivos de primeiro nível.

Algoritmo 1: Método de detecção de duas etapas.

```
Input: evento
Output: tipo_trafego
evento ← Pre-processamento (evento);
evento ← Padronização (evento);
is_intrusive ← Etapa 1 (evento);
/* Classifica em True para ataque e False para benigno.      */
if is_intrusive = False then
    tipo_trafego ← Benigno;
    Libera o tráfego;
end
else
    evento_reduzido ← Filtragem de atributos (evento);
    tipo_trafego ← Etapa 2 (evento_reduzido);
    if tipo_trafego = Benigno then
        Libera o tráfego;
    end
    else
        Invoca módulo de contramedidas (evento, tipo_trafego);
    end
end
return tipo_trafego
```

Os métodos *ensemble* fornecem maior robustez à classificação. No entanto, eles apresentam custos de processamento e treinamento mais elevados do que os modelos de classificador único. A abordagem proposta busca contornar este problema executando o método ensemble na camada de nuvem, possuindo assim maior poder computacional. Além disso, a abordagem *ensemble* só será aplicada a eventos intrusivos previamente detectados. Como os eventos benignos são detectados pela primeira etapa e são liberados, não há necessidade de resposta urgente

aos eventos enviados para esta segunda etapa de classificação, pois há grandes chances de serem intrusivos. Essa arquitetura permite que a maior parte do fluxo da rede, que é legitimo (Tavallaee et al., 2009), passe por apenas pela primeira análise e seja liberado imediatamente. O fluxo detectado como intrusivo pelo nível de nevoeiro (Etapa 1) pode, então, ser submetido a uma análise mais complexa e demorada para a classificação da categoria do ataque, sem necessitar de uma resposta urgente em tempo real. Desse modo, é possível aproveitar a principal vantagem da camada de nuvem, que é sua capacidade ilimitada de processamento, e contornar a sua limitação da latência ocasionada pela separação entre os dispositivos IoT e os *datacenters*. A seguir, são apresentados detalhes a respeito das duas etapas do método de detecção proposto.

4.1.1 Etapa 1 - Detecção

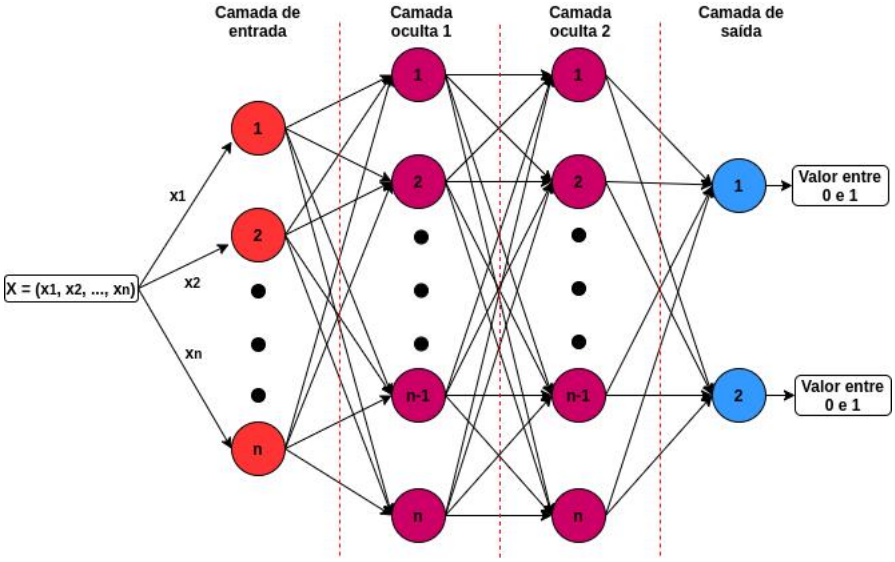
A etapa de detecção da abordagem proposta tem como objetivo examinar o tráfego monitorado buscando detectar eventos intrusivos. Esta etapa discrimina o tráfego em não intrusivo e intrusivo, constituindo, portanto, um problema de classificação binária. Os eventos intrusivos são enviados para a segunda etapa de detecção e os eventos benignos são liberados. Portanto, a maior parte dos eventos será analisada somente por este primeiro nível da abordagem e terá seus fluxos liberados, pois, serão legítimos (Tavallaee et al., 2009). Desse modo, é importante que este método de detecção binária alcance uma alta taxa de precisão e também de *recall*, mantendo a taxa de falsos negativos baixa. Ou seja, é crucial que o maior número possível de eventos intrusivos seja detectado como tal, pois eventos erroneamente detectados como benignos serão automaticamente liberados e poderão causar danos.

Há uma grande variedade de métodos de classificação capazes de realizar classificação binária e que podem ser empregados diretamente nesta etapa de detecção, como por exemplo: *Deep Neural Networks* (DNN), *k-Nearest Neighbor* (kNN), *Extra Tree* (ET) e *Random Forest (RF)*, entre outros. As redes neurais são capazes de alcançar ótimos desempenhos de classificação (Russell; Norvig, 2010). Entretanto, podem sofrer com maiores instabilidades causadas por ruídos e variança no treinamento (Seção 2.6.1.1, Página 47). O kNN, por outro lado, é um algoritmo simples e que geralmente apresenta altas taxas de acerto. No entanto, ele realiza inúmeras comparações com os dados de treinamento durante a classificação, fato este que pode acarretar em um custo elevado para detecção (Seção 2.6.1.2, Página 51). Existem problemas de aprendizado de máquina em que mesmo o melhor modelo não é preciso ou leve o suficiente. Desse modo, é interessante realizar a combinação de modelos, gerando métodos híbridos, para reduzir a instabilidade, atenuar os pontos fracos e impulsionar os pontos fortes de cada um (Govindarajan; Chandrasekaran, 2011).

Inicialmente, avaliou-se a utilização de redes neurais e do algoritmo kNN para a detecção binária desta etapa, devido aos bons desempenhos de detecção apresentados em experimentos e trabalhos anteriores. Com o objetivo de melhorar a robustez do método e reduzir o custo associado ao algoritmo kNN, em Souza et al. (2020) propusemos o método híbrido DNNKNN, uma abordagem híbrida para detecção binária baseada na combinação de DNN com o kNN.

A seguir, são apresentadas informações sobre a arquitetura DNN proposta. Trata-se de uma DNN *feed-forward* do tipo *Multilayer Perceptron*, ela foi selecionada devido a sua capacidade de resolver problemas não linearmente separáveis. A arquitetura definida para a DNN foi reutilizada a partir de experimentos prévios realizados em Souza et al. (2018) e é apresentada na Figura 8.

Figura 8 – Arquitetura da DNN proposta.



Fonte: O autor.

O número de neurônios atribuídos a cada camada oculta da rede foi definido como igual ao tamanho da entrada. Um número muito grande de camadas e/ou neurônios pode tornar o modelo muito custoso, enquanto um número muito pequeno pode comprometer o desempenho da rede (Haykin, 2001). A função tangente hiperbólica (*tanh*) foi utilizada como função de ativação dos neurônios das camadas ocultas. A função de ativação define a saída do neurônio de acordo com a entrada recebida (Russell; Norvig, 2010). Existem vários tipos de funções de ativação usadas em redes neurais. A função de ativação sigmoide possui um gráfico em forma de *S* e é uma das funções mais comumente usadas em redes neurais (Haykin, 2001). Uma importante função de ativação sigmoide é a tangente hiperbólica, que é definida a partir da razão entre seno e cosseno hiperbólico, como pode ser visto na Equação 4.1. O intervalo da função *tanh* é de $(-1$ a $1)$. Permitir que uma função de ativação assuma valores negativos fornece benefícios e vantagens analíticas durante a fase de treinamento (Haykin, 2001). Para o treinamento da rede, o algoritmo de *backpropagation* foi escolhido.

$$\varphi(v) = \frac{senh(v)}{cosh(v)} \tag{4.1}$$

De acordo com a necessidade de classificação em apenas duas categorias (intrusiva e não intrusiva), ou seja, um problema binário, a camada de saída foi fixada em 2 neurônios. Os neurônios da camada de saída utilizam a função de ativação *softmax*, que gera um valor de

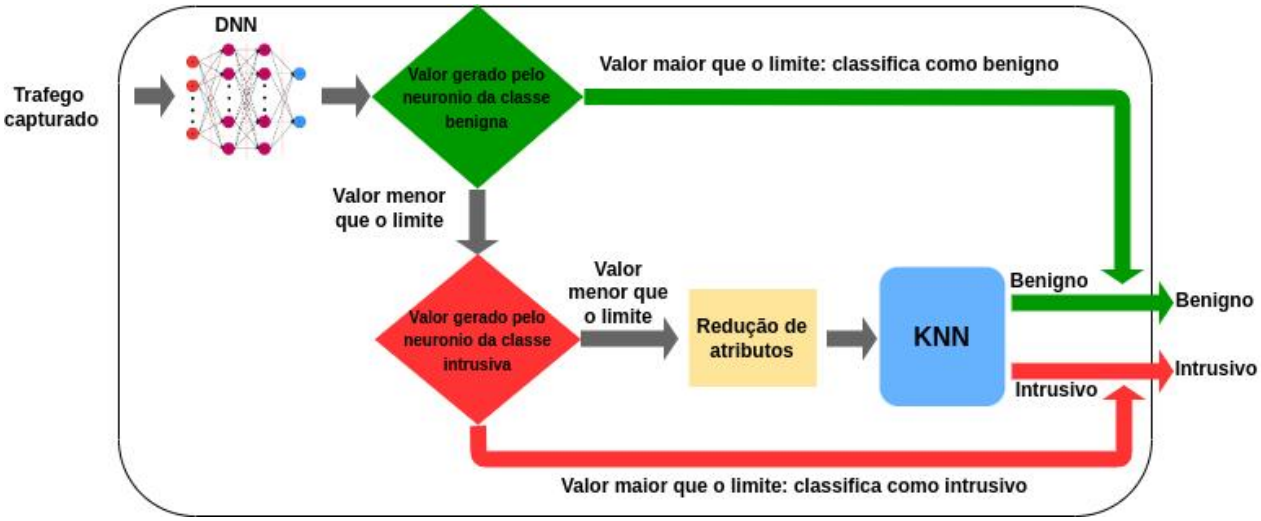
classificação entre 0 e 1 para cada neurônio. A função *softmax* transforma as saídas para cada classe em valores entre 0 e 1 e os divide pela soma (Nwankpa et al., 2018). Isso essencialmente fornece a probabilidade de a entrada pertencer a uma determinada classe.

O método proposto também é baseado no algoritmo kNN. O principal objetivo do algoritmo kNN é determinar a classe de uma nova instância. O processo calcula a similaridade dessa nova instância com os exemplos existentes no banco de dados. Este algoritmo considera os k vizinhos mais próximos, e o valor de $k = 1$ foi definido de acordo com os resultados obtidos em testes anteriores (Rosa, 2017).

Para calcular a similaridade entre as instâncias envolvidas na comparação, foi utilizada a distância euclidiana. Além disso, uma estrutura de árvore k-d é utilizada para armazenar os exemplos que compõem a base de exemplos do algoritmo kNN. As árvores k-d, abreviação de árvore k-dimensional, são estruturas de dados de particionamento de espaço para organizar pontos em um espaço k-dimensional.

O funcionamento do método DNNKNN é ilustrado na Figura 9. A sequência de ações executadas por esse método consiste em enviar o tráfego capturado para ser classificado primeiro pelo modelo neural DNN. O modelo gera uma saída em dois neurônios, um corresponde à classe intrusiva e outro à não intrusiva. Conforme supracitado, cada neurônio gera uma saída entre 0 e 1. Esse valor é a probabilidade do fluxo pertencer à classe à qual o neurônio corresponde.

Figura 9 – Ilustração do método DNNKNN.



Fonte: O autor.

A abordagem possui um limite predefinido para cada neurônio. Se o neurônio da classe normal apresentar saída maior que o seu limite predefinido, a instância de tráfego analisada será automaticamente classificada como não intrusiva. O mesmo procedimento é aplicado para a saída do neurônio da classe de ataque, conforme apresentado no Algoritmo 2. As instâncias que obtiveram valores abaixo dos limites de saída em ambos os neurônios são consideradas situações nas quais o modelo neural não obteve clareza ou precisão. Desse modo, caso nenhum

dos neurônios apresente saída maior que os seus respectivos limites, a instância de tráfego será encaminhada para análise com o método kNN.

No entanto, antes de ser analisada pelo método kNN, a instância tem seus atributos reduzidos de acordo com os escolhidos pela abordagem *InfoGain* de seleção de atributos durante o processo de treinamento. Essa etapa ocorre para reduzir a complexidade da instância e, consequentemente, do processamento do kNN. Após a redução, a instância com atributos reduzidos é submetida ao kNN.

Algoritmo 2: Método DNNET.

```
Input: evento
Output: is_intrusive
saida_neuronio_intrusivo, saida_neuronio_benigno ← DNN (evento) /* Gera
    os valores de saída dos neurônios softmax                    */
if saida_neuronio_benigno > limite_neuronio_benigno then
    is_intrusive ← False
end
else if saida_neuronio_intrusivo > limite_neuronio_intrusivo then
    is_intrusive ← True
end
else
    evento_reduzido ← Filtragem de atributos (evento)
    predicao_et ← ET (evento_reduzido) /* Classifica em 1 para
        intrusivo e 0 para benigno                               */
    if predicao_et = 1 then
        is_intrusive ← True
    end
    else
        is_intrusive ← False
    end
end
return is_intrusive
```

A ideia principal da abordagem é obter o valor de saída do modelo neural para o tráfego capturado e enviar para o método kNN as instâncias que não possuem valor maior que o limite predefinido em nenhum dos dois neurônios de saída. Para essas instâncias, a classificação gerada pelo kNN é aceita como final. Conforme relatado em Souza et al. (2020), este método proposto alcançou resultados promissores, mantendo o desempenho de classificação do kNN e obtendo uma redução de aproximadamente 89% do tempo de classificação em relação à abordagem kNN original. Além disso, foi capaz de alcançar desempenho de classificação superior a métodos clássicos de *Machine Learning* e abordagens do estado da arte.

Apesar desta melhoria alcançada através do método DNNKNN, o custo computacional ainda se apresentava elevado devido ao custo considerável do algoritmo KNN. Esse fato pode se tornar um obstáculo na aplicação da abordagem no contexto de *fog computing*, já que esses dispositivos geralmente possuem recursos mais escassos que paradigmas como a *cloud computing*. Dado que o método binário é projetado para operar na *fog* e irá atuar em um primeiro nível

de análise empregado sobre todo o tráfego da rede IoT, é necessário que ele seja leve para não causar atraso significativos no tráfego. Assim, a abordagem foi aprimorada através da substituição do KNN pelo método *Extra Tree* (ET). Portanto, o novo método, denominado DNNET, foi proposto com base na arquitetura do DNNKNN. Essa abordagem consiste na combinação das técnicas DNN e ET para classificar o tráfego como benigno ou intrusivo. O ET é um método *ensemble* de Árvores de Decisão, em inglês *Decision Tree* (DT). As DTs são métodos leves e simples que possuem bom desempenho de classificação. No entanto, são suscetíveis a problemas de sobreajuste *(overfitting)* (T.K.; Annavarapu; Bablani, 2021). A técnica ET busca, através da estratégia *ensemble*, construir um modelo de classificação mais robusto e diminuir o *overfitting*. *Extra Tree* agrega os resultados de várias árvores de decisão descorrelacionadas, acumuladas dentro de uma "floresta", para produzir os resultados da classificação. Mais informações a respeito desta técnica são apresentadas no Capítulo 2.

O procedimento de divisão ET para a construção da estrutura possui alguns parâmetros importantes. O primeiro parâmetro, c, indica o número de atributos selecionados aleatoriamente em cada nó, neste caso considera-se o valor padrão de $c = sqrt(N)$. O segundo parâmetro é m, que representa o tamanho mínimo do conjunto de treinamento para dividir um nó, neste caso definimos como 2 (m = 2). O terceiro parâmetro é a, que é o número de árvores no conjunto. A ET empregada é formada por 10 árvores de decisão, portanto, $a = 10$. O limite de profundidade (p) foi fixado em 10 níveis ($p = 10$). Este parâmetro controla o tamanho das árvores. A falha em definir essa estrutura gera árvores totalmente crescidas não podadas, que podem ser potencialmente muito grandes em alguns conjuntos de dados (Geurts; Ernst; Wehenkel, 2006). Além disso, o Índice de Gini é usado como critério, ele pode ser utilizado para medir quão bem um determinado atributo separa as classes contidas em um nó.

Conforme pode ser observado no Algoritmo 2, a abordagem DNNET possui um funcionamento similar ao DNNKNN. No entanto, nesta nova abordagem, antes da instância ser analisada pelo segundo método de detecção, ela tem seus atributos reduzidos de acordo com os escolhidos pela abordagem de seleção de atributos durante o processo de treinamento. A abordagem de seleção de atributos empregada nesta abordagem é apresentada em detalhes na Seção 4.1.3 (Página 91). Após a redução, a instância com atributos reduzidos é submetida ao método ET.

4.1.2 Etapa 2 - Identificação

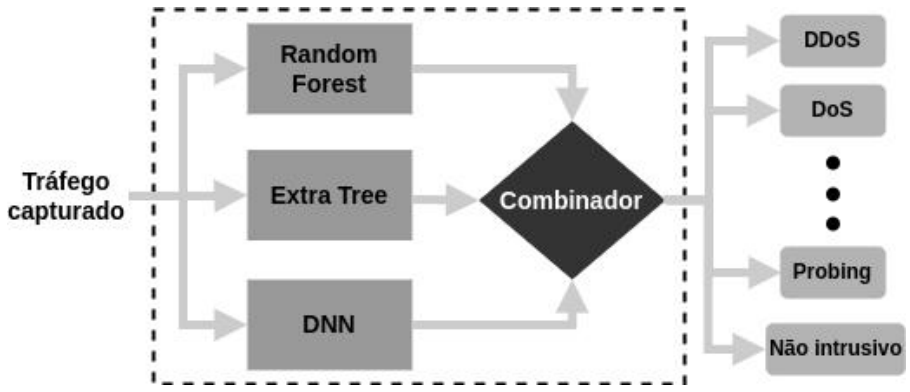
A segunda etapa da abordagem proposta é a fase de Identificação, e ela foi projetada para operar na camada de computação em nuvem. Nessa etapa, o tráfego previamente identificado como intrusivo será analisado por métodos mais robustos. O objetivo da análise é classificar o tráfego em uma categoria específica de intrusão. Portanto, esta etapa consiste em um problema multiclasse.

Conforme mencionado na Seção 3.1.2.2, para um problema multiclasse, diversos métodos de aprendizado de máquina podem ser aplicados. Alguns classificadores multiclasse di-

retos foram investigados pelo estado da arte para resolver este problema (Diro; Chilamkurti, 2018c; Nguyen et al., 2019; Almiani et al., 2020), conforme pode ser observado na Seção 3.2 (Página 73). Entretanto, estas abordagens geralmente apresentam taxas de precisão mais baixas em comparação com métodos binários (Prabavathy; Sundarakantham; Shalinie, 2018; Nguyen et al., 2019; Zhao et al., 2022). Isto é atribuído principalmente aos desafios na identificação de tipos específicos de ataques (Prabavathy; Sundarakantham; Shalinie, 2018; Diro; Chilamkurti, 2018c; Almiani et al., 2020). Os métodos únicos de classificação podem sofrer com instabilidade, não havendo garantia que um método classificador possa obter sempre o melhor desempenho em todas as situações (Samat et al., 2014).

A Figura 10 apresenta uma ilustração do método multiclasse proposto para operar na Etapa 2. Esse método consiste em um *ensemble* composto por três técnicas diferentes de aprendizado de máquina, uma *Extra Tree*, uma *Random Forest* (RF) e um modelo neural DNN. Conforme mencionado na Seção 2.6.1.6 (Página 53), os métodos *ensemble* são criados combinando vários modelos. A ideia principal dos métodos ensemble é combinar as classificações de vários classificadores de aprendizado de máquina de base conceitualmente diferentes para melhorar a generalização e a robustez em relação a classificadores únicos (Goodfellow et al., 2016). Duas das técnicas, a ET e a RF, que compõem o método, são também abordagens *ensemble* já consolidadas. A inserção de redundância de classificadores com métodos *ensemble* mostra-se interessante para diminuir variação dos classificadores únicos. Com *Ensemble Learning*, um desempenho de classificação melhor do que qualquer classificador único pode ser alcançado (Samat et al., 2014).

Figura 10 – Esboço da estrutura do método *ensemble* da Etapa 2.



Fonte: O autor.

Apesar da robustez fornecida pelos métodos *ensemble* em relação a classificação, eles possuem custo de tempo de processamento e de treinamento maior em relação a modelos de classificadores únicos. A abordagem proposta contorna esta limitação através da execução do método *ensemble* na camada de nuvem, dispondo, portanto, de maior poder computacional. Além disso, a abordagem baseada em *ensemble* será aplicada somente nos eventos previamente detectados como intrusivos, para classificá-los em uma categoria específica de ataque. Como

os eventos normais já foram detectados pela primeira etapa e já foram liberados, não há necessidade de uma resposta urgente aos eventos enviados para esta segunda etapa de classificação, pois há grande chance de serem intrusivos. Desse modo, o atraso no fluxo de tráfego legítimo será mínimo.

A primeira técnica que compõe a abordagem *ensemble* é a *Random Forest*. As RFs são métodos *ensemble* que agregam os resultados de várias DTs descorrelacionadas acumuladas dentro de uma "floresta" para diminuir o sobreajuste *(Overfitting)* (Breiman, 2001). Elas são baseadas em *Bagging* mas fornecem uma camada adicional de aleatoriedade. Além de construir cada DT usando uma amostra de *bootstrap* diferente dos dados, em RFs, cada nó é dividido usando o melhor entre um subconjunto de preditores escolhidos aleatoriamente naquele nó. As fontes de aleatoriedade visam diminuir a variância do estimador de floresta (Breiman, 2001; Liaw; Wiener et al., 2002).

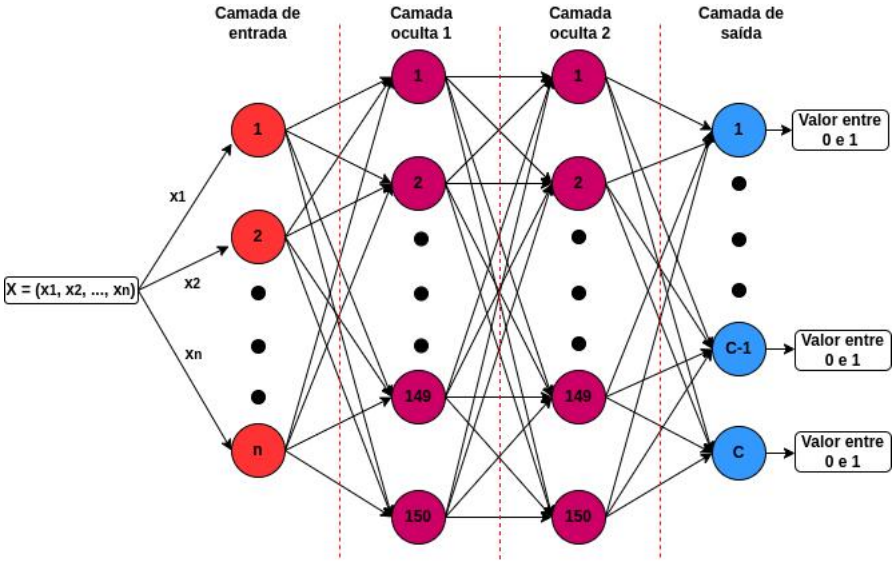
O principal parâmetro da RF é a quantidade de DTs para compor a RF, ou seja, o número de árvores na floresta. Devido a maior capacidade de recursos da nuvem, não há uma necessidade de diminuir recursos da estrutura para economizar recursos, desse modo, a quantidade de DTs na RF foi definida como 100 ($a = 100$), sendo o valor padrão da biblioteca. O índice de Gini é utilizado como critério e o número de atributos candidatos selecionados aleatoriamente para melhor divisão foi definido como a raiz do número de atributos existentes ($c = sqrt(N)$), sendo N o numero de atributos do *dataset*. Além disso, o tamanho mínimo do conjunto de treinamento para dividir um nó foi definido como 2 ($m = 2$) e o limite de profundidade, parâmetro p, foi fixado em 100 níveis para cada árvore ($p = 100$). Isso para evitar o crescimento excessivo de árvores, que podem ser tornar muito grandes em alguns conjuntos de dados (Geurts; Ernst; Wehenkel, 2006). Em contraste com a implementação original de RF, a implementação usada neste trabalho combina classificadores pela média de sua previsão probabilística em vez de permitir que cada classificador vote em uma única classe.

A segunda técnica que compõem o método *ensemble* é a ET. Devido ao bom desempenho dessa técnica no método binário DNNET da primeira etapa e em experimentos prévios, buscou-se utilizar para este método as mesmas configurações e estruturas da ET idealizada no método DNNET. No entanto, nesta etapa, este método compõe uma abordagem *ensemble* juntamente com os outros métodos de classificação. Devido à maior capacidade de recursos disponíveis na Etapa 2 e à possibilidade de trabalhar com um *delay* no tempo de resposta, essa estrutura da ET pode ser aprimorada através da utilização de um número maior de arvores de decisão. Desse modo, a quantidade de árvores para compor a ET foi fixada em ($a = 100$). Além disso, a estrutura ET definida para o método binário DNNET possuía limite de profundidade igual a 10 ($p = 10$). Desse modo, as arvores não poderiam crescer excessivamente e, consequentemente, seriam menos custosas. Como o método multiclasse vai operar em um contexto com maiores recursos, o limite de profundidade foi relaxado em 10 vezes, portanto, fixou-se um limite de 100 níveis para cada árvore ($p = 100$). A estratégia utilizada para combinação das predições das DTs foi a média da previsão probabilística.

Por fim, a última técnica que compõe o método *ensemble* é uma DNN. A DNN utili-

zada no método binário DNNET da Etapa 1 possui uma arquitetura com duas camadas ocultas, com quantidade de neurônios igual à quantidade de atributos existentes. Nesta Etapa 2, a maior capacidade de recursos permite trabalhar com uma DNN com arquitetura mais complexa, buscando melhores resultados de classificação. Desse modo, a arquitetura da DNN presente no método *ensemble* foi definida com duas camadas ocultas, cada uma contendo 150 neurônios, conforme apresentado na Figura 11.

Figura 11 – Arquitetura da DNN proposta para compor o método *ensemble* multiclasse.



Fonte: O autor.

Os neurônios dessas camadas utilizam a função de ativação ReLU, que retorna o valor recebido pelo somador se for maior que zero. Se os valores forem negativos, retorna zero. Esta função tornou-se amplamente utilizada em diversas redes neurais por ser mais fácil de treinar e geralmente alcançar bons resultados. Em relação à camada de saída, ao contrário da DNN binária da primeira etapa, que possuía apenas 2 neurônios, esta DNN deve possuir uma quantidade C de neurônios, correspondente à quantidade de classes utilizadas para treinar o modelo. Os neurônios da camada de saída utilizam a função de ativação *softmax*, gerando um valor de classificação entre 0 e 1 para cada neurônio, o qual corresponde à probabilidade da entrada pertencer à classe daquele neurônio. Desse modo, a instância será classificada de acordo com neurônio que possuir o maior valor de saída.

Além das técnicas de classificação, outro componente extremamente importante em métodos *ensemble* é a estratégia de combinação das classificações geradas por cada uma das técnicas individuais. Uma das estratégias é conhecida como combinação *hard*, onde cada técnica de classificação individual vota em uma classe, e define-se como classificação final a classe que obteve mais votos. A outra estratégia é conhecida como combinação *soft* e foi a utilizada no método *ensemble* proposto. Esta estratégia consiste em um combinador baseado na máxima soma das probabilidades das previsões. De modo simplificado, cada técnica de classificação

individual fornece um valor de probabilidade de que a instância pertença a uma determinada classe. As previsões são então somadas, e em seguida, a classe com a maior soma de probabilidades é definida como a classificação final.

A seguir são apresentados detalhes a respeito da estratégia de seleção de atributos proposta.

4.1.3 Estratégia de seleção de atributos

Os métodos de seleção de atributos são amplamente utilizados em diversas aplicações relacionadas ao aprendizado de máquina. Essa técnica é uma grande aliada, pois busca diminuir a dimensionalidade das bases de dados ao selecionar os atributos que mais contribuem para o processo de aprendizado (Hoque; Singh; Bhattacharyya, 2018).

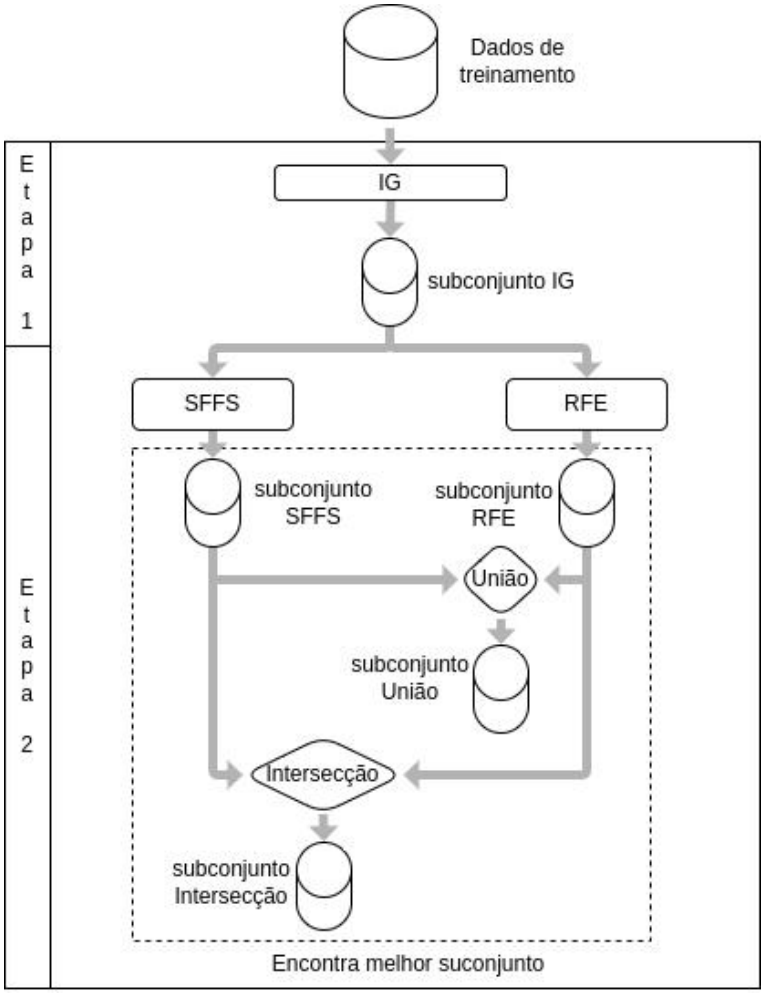
Neste contexto, a seleção de atributos é realizada com as informações extraídas do tráfego de rede capturado, visando encontrar as melhores características a serem utilizadas no processo de análise. Somente os recursos mais relevantes para o processo de detecção são extraídos e empregados no treinamento dos modelos de detecção. A atividade de seleção de atributos visa obter uma melhora no desempenho de detecção e/ou reduzir o uso de recursos computacionais. Os atributos podem ser selecionados de acordo com sua relevância e qualidade para a tarefa de classificação.

Conforme apresentado no Capitulo 2, existem vários tipos de métodos para a seleção de atributos, cada um com suas vantagens e desvantagens. Juntamente com Valencio (2022), propomos uma estratégia *ensemble* de seleção de atributos através da combinação de métodos baseados em diferentes técnicas, de modo a minimizar as desvantagens e maximizar seus pontos fortes.

A estratégia *ensemble* de seleção de atributos proposta possui duas etapas principais. Na Etapa 1, é aplicado um algoritmo de seleção de atributos do tipo filtro, o *Information Gain* (IG). O conjunto de atributos selecionado pela Etapa 1 é então submetido para a Etapa 2, conforme apresentado na Figura 12. A Etapa 2 é composta por uma combinação de algoritmos do tipo *wrapper*. Por fim, o conjunto de melhores atributos gerado pela segunda etapa é tido como resultado da abordagem.

A utilização de um método do tipo *filter* na Etapa 1 possui grande importância, pois, é independente de um algoritmo de classificação. Além disso, a filtragem inicial com o *Information Gain* permite submeter à Etapa 2 um conjunto de dados já parcialmente reduzido, agilizando o processo. Abordagens *wrapper* demandam maior processamento e geram maiores custos computacionais, fato este que pode ser proibitivo ao se lidar com grandes quantidades de dados. No entanto, a utilização de métodos *wrapper* não deve ser descartada, pois eles tendem a obter conjuntos de atributos com maior qualidade para o processo de detecção. Desse modo, a estratégia proposta combina métodos *filter* com métodos *wrappers* para obter os melhores atributos e reduzir o custo computacional dos métodos *wrapper*.

Figura 12 – Arquitetura da estratégia de seleção de atributos.



A seguir são apresentadas as principais características de cada uma das etapas da estratégia proposta.

- **Etapa 1:** Nesta etapa, é aplicado inicialmente o algoritmo *Information Gain*, do tipo *filter*. O objetivo desta etapa é realizar uma redução inicial de dimensionalidade dos dados em um primeiro contato. Os métodos *filter* são computacionalmente mais leves em relação às abordagens *wrapper*. Diversos métodos do tipo *filter* são utilizados na literatura, e entre estes, o IG é amplamente utilizado entre os pesquisadores para analisar quais os atributos mais relevantes e significativos (Stiawan et al., 2020). O IG é um algoritmo que gera como saída um ranqueamento de acordo com o valor de ganho de informação de cada atributo. Ele trabalha com um ponto de corte, ou *threshold*, que precisa ser estabelecido para descartar os atributos menos relevantes. Com o propósito de definir o ponto de corte, diversas abordagens foram avaliadas no estado da arte. Belanche e González (2011) optaram por descartar os atributos cujo peso variavam da média mais que duas vezes a variância. Bolón-Canedo, Sánchez-Maroño e Alonso-Betanzos (2015) fixaram uma porcentagem de atributos, de acordo com a quantidade total de atributos presentes na base de eventos, por exemplo, descartar 25% dos atributos quando $N < 10$. A abor-

dagem considerada neste trabalho foi a apresentada por Mejía-Lavalle, Sucar e Arroyo (2006) e é definida pela maior diferença entre dois atributos consecutivos no ranking, ou seja, busca-se descartar todos os atributos abaixo da maior redução de pontuação. Os atributos acima desse ponto de redução são então enviados para a Etapa 2 da estratégia proposta.

- **Etapa 2:** Esta etapa é responsável por realizar a seleção de atributos com métodos *wrapper*. A utilização combinada de métodos *filter* com métodos *wrappers* potencializa as chances de ser encontrado um melhor subconjunto de atributos e também reduz o custo computacional de uma abordagem *wrapper* tradicional, visto que os algoritmos de menor contribuição na avaliação já foram removidos na Etapa 1. Os métodos *wrapper* tradicionais geralmente possuem um custo elevado devido a realização de N iterações, onde em cada iteração busca-se encontrar os melhores K atributos, sendo N a quantidade de atributos existentes no conjunto de dados e K a quantidade de atributos selecionados. Em cada iteração, é considerado um número de K diferente, de modo, que todas as quantidades possíveis de atributos sejam avaliadas com os seus melhores atributos. Após todo esse processo, é possível obter os melhores atributos na quantidade ideal em relação a todas as combinações possíveis. Devido ao alto custo dessa abordagem tradicional, propomos uma nova estratégia onde, em vez de executar um algoritmo *wrapper* completo, executamos dois diferentes algoritmos *wrapper* de maneira parcial e buscamos combinar os seus resultados. Neste caso, em vez de realizar as N iterações, a abordagem realiza apenas 3 para cada algoritmo, totalizando 6 iterações. Para cada algoritmo são considerados valores de K igual a 75%, 50% e 25% da quantidade de atributos existentes, para cada uma das iterações respectivamente. Por fim, o melhor conjunto de cada algoritmo dentre as 3 iterações é selecionado. De modo, a contornar o risco de selecionar um conjunto de atributos que não corresponde a quantidade ótima, buscou-se a utilização de duas estratégias *wrapper* distintas, cujos subconjuntos gerados são então combinados para obter um subconjunto de atributos otimizado. Para se combinar esses conjuntos de maneira efetiva, é aplicado um algoritmo combinador que busca uma relação entre os atributos de cada subconjunto produzido pelos métodos *wrapper*, de modo a gerar um conjunto final de atributos melhores e mais relevantes.

  Os métodos *wrappers* que compõem a Etapa 2 são:

  - *Sequential Forward Feature Selection* (SFFS): algoritmo *wrapper* baseado na estratégia de busca *forward*.
  - *Recursive Feature Elimination* (RFE): algoritmo *wrapper* baseado em selecionar recursos considerando recursivamente conjuntos cada vez menores.

  Os métodos *wrappers* de seleção de atributos geralmente são capazes de obter melhores conjuntos de atributos do que os métodos *filter*, pois buscam a melhor combinação

de atributos em determinado conjunto com a utilização de um motor de inferência para avaliar o subconjunto de cada iteração (Guerra-Manzanares; Bahsi; Nõmm, 2019). Nesta abordagem, os algoritmos são utilizados com o método *Extra Tree* como motor de indução. Dentre as várias formas possíveis de combinar os subconjuntos de atributos gerados pelos métodos *wrappers*, utilizamos a união e a intersecção entre estes subconjuntos, pois são técnicas consideradas interessantes por Bolón-Canedo e Alonso-Betanzos (2019). A união entre os conjuntos consiste em combinar todos os atributos selecionados pelo menos uma vez por algum dos algoritmos. Esta abordagem tende a gerar uma menor redução no número de atributos. A combinação por intersecção consiste em selecionar apenas os atributos selecionados por ambos os algoritmos. No entanto, esta abordagem pode levar a um conjunto de atributos muito restritivo, produzindo maus resultados (Álvarez-Estévez et al., 2011).

O Algoritmo 3 apresenta o pseudocódigo da estratégia proposta. Inicialmente, na Etapa 1, é realizada a seleção com IG, e o subconjunto gerado é submetido para a Etapa 2.

Algoritmo 3: Estratégia de seleção de atributos.

```
Input: dados
Output: dados_selecionados
/* Etapa 1 - Filtro.                                          */
dados_IG ← InformationGain(dados);
/* Etapa 2 - Wrappers.                                        */
dados_SFFS ← SequentialFeatureSelector(dados_IG);
dados_RFE ← RecursiveFeatureElimination(dados_IG);
dados_selecionados ← dados_SFFS;
if Avaliação(dados_SFFS) < Avaliação(dados_RFE) then
    dados_selecionados ← dados_RFE;
end
dados_Uniao ← Uniao(dados_SFFS, dados_RFE);
if Avaliação(dados_selecionados) < Avaliação(dados_Uniao) then
    dados_selecionados ← dados_Uniao;
end
dados_Interseccao ← Interseccao(dados_SFFS, dados_RFE);
if Avaliação(dados_selecionados) < Avaliação(dados_Interseccao) then
    dados_selecionados ← dados_Interseccao;
end
return dados_selecionados;
```

Na Etapa 2, inicialmente, são executados os dois algoritmos *wrapper*, o SFFS e o RFE. A partir disso, os subconjuntos gerados são avaliados e comparados para encontrar o melhor subconjunto. O melhor subconjunto dentre SFFS e RFE é então comparado com um subconjunto formado através da união dos subconjuntos SFFS e RFE. Posteriormente, o novo

melhor subconjunto é comparado com um subconjunto formado através da intersecção dos subconjuntos SFFS e RFE. Por fim, o algoritmo retorna o subconjunto que obteve a melhor avaliação dentre todas as configurações.

A seguir são apresentados detalhes a respeito do processo de treinamento completo da abordagem proposta.

4.1.4 Treinamento

O processo de treinamento é responsável por gerar os modelos da abordagem proposta e torná-la capaz de analisar o tráfego da rede para identificar intrusões. Este processo pode ser tornar oneroso, pois envolve também o treinamento do modelo neural do método de detecção binária DNNET, onde os pesos dos neurônios são ajustados.

O Algoritmo 4 descreve o processo de treinamento da abordagem proposta. Inicialmente, o conjunto de dados de treinamento passa por uma etapa de pré-processamento, na qual são realizadas diversas tarefas. Uma delas é a padronização, para obter o conjunto normalizado de informações.

Algoritmo 4: Algoritmo de treinamento da abordagem proposta.

Input: *dados*, *rotulos*, *taxa_FP_aceitavel*, *taxa_FN_aceitavel*
dados ← Pre_processamento (*dados*);
dados, *rotulos* ← BalancearClasses (*dados*, *rotulos*);
dados_reduzidos ← EstrategiaSelecaoAtributos (*dados*, *rotulos*);
et ← TreinarMetodoMulticlasse (*dados_reduzidos*, *rotulos*);
rotulos_binarios ← BinarizarDados (*rotulos*);
dnnet ← TreinarMetodoBinárioDNNET (*dados*, *rotulos_binarios*, *taxa_FP_aceitavel*, *taxa_FN_aceitavel*);

A padronização do conjunto de dados ajuda a melhorar o desempenho de alguns classificadores baseados em aprendizado de máquina que precisam que seus recursos sejam distribuídos normalmente. O método utilizado para normalização dos dados foi o escalonamento padrão, dado pela Equação 4.2, sendo x a amostra, u a média e s o desvio padrão. A média e o desvio padrão são obtidos com base nas estatísticas de cada atributo no conjunto de dados.

$$z = \frac{x - u}{s} \tag{4.2}$$

Além disso, nesse processo de pré-processamento, também são excluídos os registros com valores inválidos e nulos. O conjunto de dados pré-processado é então submetido ao processo de balanceamento de classes, conforme ilustrado na Figura 13.

A etapa de balanceamento de classes visa contornar a disparidade na proporção de tráfego entre os tipos de ataques, pois isso pode causar problemas durante o treinamento. O objetivo é proporcionar dados suficientes para que os modelos possam entender os padrões de ataques que têm pouca ocorrência. Uma das estratégias para trabalhar com conjuntos de dados

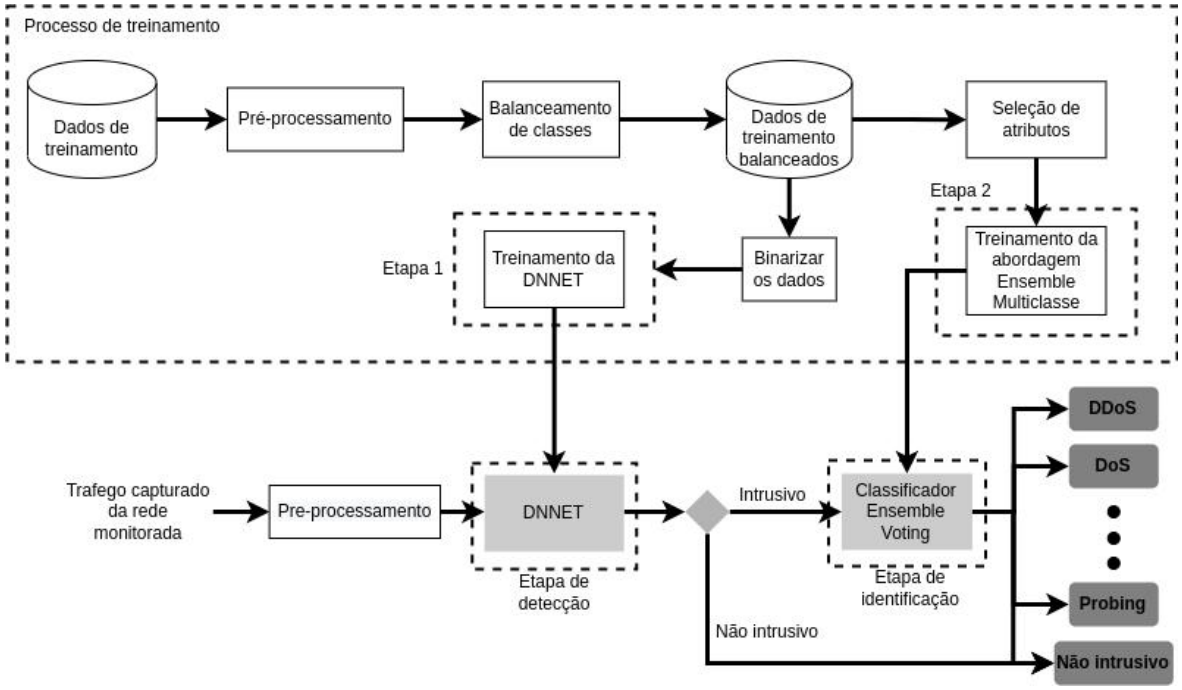


Figura 13 – Processo de treinamento da abordagem proposta.

desbalanceados é criar novas instâncias para classes minoritárias. Novas instâncias de dados podem ser geradas a partir de exemplos existentes.

O método *Synthetic Minority Oversampling Technique* (SMOTE) (Bowyer et al., 2011) se baseia na estratégia de criar novas amostras sintéticas e é amplamente utilizado para o balanceamento de classes. Ele seleciona amostras próximas no espaço de recursos, desenha uma linha entre elas e cria uma nova instância em um ponto ao longo dessa linha. No entanto, aplicar este método em cenários onde há um grande desbalanceamento entre as classes e um grande número de classes, pode gerar problemas. A criação de um número muito grande de registros sintéticos pode prejudicar o desempenho do modelo de aprendizado de máquina, além de tornar mais custoso o processo de treinamento.

Desse modo, propomos a utilização de uma estratégia menos agressiva de balanceamento, onde definimos uma porcentagem adequada de registros para cada classe, baseada na quantidade total de registros. O objetivo é criar novos registros para as classes minoritárias até que elas atinjam uma quantidade mínima que possa contribuir para o processo de aprendizado do método de detecção. A estratégia não tem como objetivo igualar a quantidade de registros de todas as classes com a quantidade de registros da classe majoritária, pois, em cenários extremamente desbalanceados, isso acarretaria em uma enorme quantidade de novos registros sintéticos. O pseudo-código da estratégia de balanceamento de classes proposta é apresentado no Algoritmo 5. Inicialmente, identifica-se a quantidade de classes existentes no conjunto de dados (*quantidade_classes*), e a partir disso, calcula-se a porcentagem adequada de registros para cada classe (*porcentagem*). Essa porcentagem é utilizada para calcular a quantidade mínima de registros para cada classe (*quantidade_minima*).

Algoritmo 5: Estratégia de balanceamento de classes proposta.

Input: $dados$, $rotulos$
Output: $dados\_balanceados, rotulos\_balanceados$
$quantidade\_classes \leftarrow contarQuantidadeClasses(rotulos)$;
$porcentagem \leftarrow 100/quantidade\_classes$;
$quantidade\_minima \leftarrow (contarRegistros(dados)/100) * porcentagem$;
$quantidade\_registros\_por\_classe \leftarrow contarRegistrosPorClasse(dados, rotulos)$;
$i \leftarrow 1$;
while $i <= quantidade\_classes$ **do**
  if $quantidade\_registros\_por\_classe[i] < quantidade\_minima$ **then**
    $quantidade\_final\_registros\_por\_classe[i] \leftarrow quantidade\_minima$;
  end
  else
    $quantidade\_final\_registros\_por\_classe[i] \leftarrow quantidade\_por\_classe[i]$;
  end
  $i \leftarrow i + 1$;
end
$dados\_balanceados, rotulos\_balanceados \leftarrow$ SMOTE
 $(dados, rotulos, quantidade\_final\_registros\_por\_classe)$;

Posteriormente, para cada uma das classes, é realizada uma verificação. Caso a quantidade existente atualmente seja menor que a mínima ($quantidade\_minima$), então define-se como nova quantidade de registros para a classe ($quantidade\_final\_registros\_por\_classe$) a quantidade mínima definida anteriormente. Esses valores são utilizados, então, para indicar ao SMOTE a quantidade pretendida de registros para cada uma das classes.

Após balancear os dados de treinamento originais, um novo conjunto de dados com classes balanceadas é gerado. Esse conjunto é, então, submetido em dois fluxos diferentes: um para o treinamento do método DNNET binário e outro para o treinamento do método multiclasse, conforme pode ser visto na Figura 13.

4.1.4.1 Treinamento do modelo de detecção da Etapa 1

Um dos fluxos, conforme ilustrado na Figura 13, corresponde ao treinamento do método DNNET. No entanto, por se tratar de uma classificador para a etapa de detecção binária, é necessário fornecer dados com rótulos binários para o treinamento. Desse modo, o conjunto de dados balanceado passa por uma etapa de binarização para converter todos os fluxos intrusivos para o rótulo 1 e os benignos para o rótulo 0. O conjunto de dados binário é usado para treinar a abordagem DNNET, conforme mostrado no Algoritmo 4.

O processo de treinamento da abordagem inclui o treinamento do modelo neural da DNNET, a seleção de atributos no contexto binário, o treinamento da ET binária e o processo de ajuste dos limites dos neurônios de saída do modelo neural, como pode ser visto no Algoritmo 6.

Para uma melhor compreensão, é importante enfatizar que a DNN gera dois valores

de saída entre 0 e 1 para cada instância do tráfego. Basicamente, a função *softmax* transforma os valores de cada neurônio da camada de saída em valores entre 0 e 1 e divide pela soma das saídas. Isso essencialmente dá a probabilidade de que a entrada pertença a classe representada por aquele neurônio. A abordagem DNNET possui limites para cada um dos neurônios de saída da DNN, os quais são utilizados para determinar se a classificação de uma determinada entrada será aceita com base apenas na classificação do modelo neural ou se será necessário submetê-la à classificação da ET binária. As entradas que obtiverem um neurônio com valor acima do limite são classificadas com a classe correspondente ao respectivo neurônio. Entradas que obtém ambas as saídas abaixo dos limites são enviados para serem classificados com a ET.

O Algoritmo 6 é usado para treinar e definir os limites dos neurônios de saída do modelo neural. Esse algoritmo possui dois parâmetros principais, a taxa aceitável de falsos positivos ($taxa\_FP\_aceitavel$) e a taxa aceitável de falsos negativos ($taxa\_FN\_aceitavel$). O processo de definição dos limites é realizado de maneira iterativa até que as taxas de falso positivo ($taxa\_fp$) e falso negativo ($taxa\_fn$) obtidas sejam menores que às taxas aceitáveis ou até que os limites sejam iguais a 1. Neste último caso, trata-se de um cenário no qual o treinamento do modelo neural apresentou instabilidades, resultando na decisão de enviar todas as instâncias de tráfego para serem classificadas pela ET.

Algoritmo 6: Treinamento da abordagem DNNET.

```
Input: dados, rotulos, taxa_FP_aceitavel, taxa_FN_aceitavel
modelo_neural ← Treinamento Modelo DNN (dados, rotulos);
dados_reduzidos ← EstrategiaSelecaoAtributos (dados, rotulos)
et ← Treinamento ET Binária(dados_reduzidos, rotulos);
limite_neuronio_benigno ← 0.5;
limite_neuronio_intrusivo ← 0.5;
i ← 1;
while i <= 10 do
    predicoes, saida_neuronios ← DNNET (dados)
    taxa_fp, taxa_fn ← Calcular Métricas (predicoes, labels);
    if (taxa_fp > taxa_FP_aceitavel) then
        limite_neuronio_intrusivo ← percentile(saida_neuronios.intrusivo, i*10);
    end
    if (taxa_fn > taxa_FN_aceitavel) then
        limite_neuronio_benigno ← percentile(saida_neuronios.benigno, i*10);
    end
    if (taxa_fp <= taxa_FP_aceitavel) and (taxa_fn <= taxa_FN_aceitavel)
     then
        break;
    end
    i ← i + 1;
end
```

Inicialmente, é realizado o treinamento do modelo DNN. Esse processo envolve várias iterações nas quais dados são submetidos ao modelo, e a resposta obtida é comparada com a

resposta desejada. Em cada iteração, os pesos dos neurônios são ajustados com base no erro calculado. Dessa maneira, o modelo aprende com os dados e converge para um modelo final capaz de classificar novos eventos. No entanto, esse processo pode se tornar oneroso ao se utilizar uma grande quantidade de dados de treinamento.

Treinar o modelo neural DNN é um processo de alto custo. A realização deste treinamento de maneira tradicional em nós locais da computação em nevoeiro pode sobrecarregar os dispositivos. Para gerar um modelo neural, é necessária uma quantidade substancial de dados rotulados para treinamento, tornando o processo caro. Além disso, a transmissão de dados locais para um dispositivo central com maior capacidade, como a nuvem, pode prejudicar a comunicação e levantar preocupações sobre a privacidade dos dados (Rey et al., 2022; Souza et al., 2022).

Para enfrentar esses desafios, foi proposta uma abordagem de treinamento do modelo neural baseada em aprendizagem federada. Nesse cenário, os dispositivos do nevoeiro colaboram para treinar um modelo compartilhado. Cada dispositivo retém seus dados localmente, eliminando a necessidade de transmitir os dados para um servidor de nuvem externo e, assim, preservando a privacidade dos dados (Rey et al., 2022; Souza et al., 2022).

Nesta abordagem, existe um servidor central localizado na nuvem que coordena o processo de treinamento envolvendo diversos clientes inseridos nos dispositivos do nevoeiro. No entanto, os dados locais não são enviados para o servidor central, esse servidor é responsável pela agregação dos parâmetros e definição de hiper-parâmetros. Inicialmente, o servidor central inicia o processo de treinamento com um modelo global, o qual é transmitido para os clientes. Nesta estratégia, os clientes atualizam seus modelos de acordo com o global e realizam treinamento localmente utilizando seus dados locais para gerar novos modelos locais atualizados. Em seguida, os clientes enviam o conjunto de pesos sinápticos dos neurônios dos modelos locais atualizados para o servidor central em nuvem, onde seus resultados são agregados para construir o modelo global, conforme ilustrado na Figura 14. Na solução proposta o servidor central realiza a agregação dos pesos sinápticos através do algoritmo *FedAvg* (Rey et al., 2022), o qual executa a média dos modelos. Após a agregação do modelo global, os pesos sinápticos globais atualizados são enviados para todos os dispositivos envolvidos na estratégia de treinamento na borda da rede. Esse ciclo se repete a cada rodada de treinamento.

Considerando a questão da privacidade, o envio de dados locais para servidores em nuvem pode ser considerado um problema. Conforme ilustrado na Figura 14, a solução proposta visa treinar a abordagem de detecção preservando os princípios de privacidade. Portanto, a solução evita o envio de dados para servidores em nuvem. O processo de treinamento ocorre localmente, resultando na geração de modelos de detecção locais. Portanto, apenas os pesos do modelo neural são transmitidos aos servidores em nuvem para serem adicionados ao modelo global. Após a agregação, os parâmetros atualizados do modelo global são enviados aos nós locais. Desse modo, preserva-se a privacidade dos dados e torna-se possível realizar o treinamento neural de maneira distribuída.

Após esta etapa, é realizada uma seleção de atributos nesse contexto binário, através

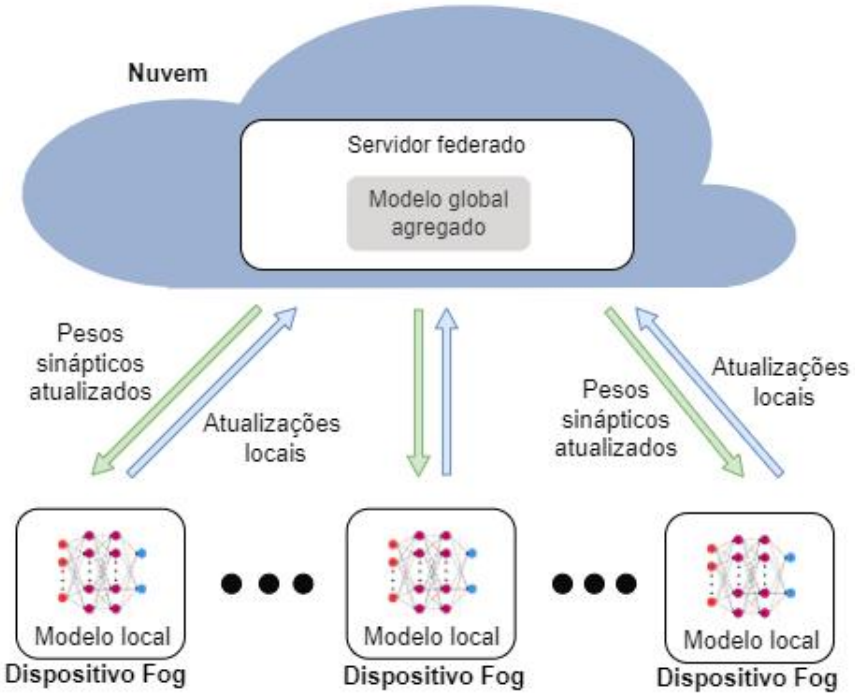


Figura 14 – Estratégia de treinamento federado.

da aplicação da estratégia de seleção de atributos apresentada na Seção 4.1.3 (Página 91). O método utilizado é mesmo aplicado para a etapa multiclasse, no entanto, neste caso, ele é aplicado ao conjunto de dados binário. Este conjunto de dados com os atributos selecionados é então usado para construir a estrutura ET binária.

O processo de treinamento da ET binária consiste na criação das 10 árvores de decisão que compõem a ET binária. O algoritmo de treinamento de árvores de decisão constrói de maneira recursiva uma estrutura em forma de árvore de dados, buscando em cada iteração da construção o atributo que melhor divide o conjunto de dados. A ET introduz aspectos de aleatoriedade no processo de construção das árvores que à compõem, de modo a criar árvores descorrelacionada. Mais informações a respeito desse processo são apresentadas no Capítulo 2.

Posteriormente, inicia-se o processo de definição dos limites dos neurônios utilizados pela DNNET. Inicialmente, os limites do neurônio benigno e do neurônio intrusivo são definidos em $0,5$. Nesse caso, todas as instâncias classificadas pela DNN são aceitas e nenhuma é enviada à ET. A partir disso, inicia-se um processo iterativo de até 10 etapas. Em cada iteração, o método DNNET é aplicado aos dados de treinamento para gerar classificações e dois conjuntos de valores. Esses valores correspondem à saída gerada pelos neurônios para cada um dos dados de treinamento. Um conjunto de valores de saída do neurônio correspondente à classe benigna e um conjunto de valores de saída do neurônio correspondente à classe intrusiva.

A partir das classificações geradas, são calculadas as métricas taxa de Falso Positivo (FP) e taxa de Falso Negativo (FN), representadas por $taxa_fp$ e $taxa_fn$, respectivamente. A métrica $taxa\_fp$, correspondente à taxa de falsos positivos, é comparada com o parâmetro

$taxa\_FP\_aceitavel$. Se for maior que o aceitável, um novo limiar é definido para o neurônio de ataque. Esse novo limite será maior para que um número menor de instâncias seja classificado apenas pelo DNN, e um maior seja enviado à ET. O mesmo procedimento ocorre com a taxa de falso-negativo ($taxa\_fn$) e o parâmetro $taxa\_FN\_aceitavel$.

A definição do novo limiar é baseada nos conjuntos de valores de saída do neurônio benigno e do neurônio de ataque obtidos durante a classificação da DNNET com os dados de treinamento. Este novo limiar para cada neurônio é atribuído de acordo com o valor correspondente ao percentil $(i * 10)$ do conjunto de valores gerados por cada neurônio de saída, onde i corresponde à iteração atual. O processo de definição de limite continua para a próxima iteração, onde é repetido com os novos limites. Assim, os limites são incrementados em 10 percentis a cada iteração, até atingir um limite que atinja a taxa aceitável de FP e FN. Se as porcentagens de FP e FN forem menores que as aceitáveis a qualquer momento, os limites ideais são encontrados e o treinamento é finalizado.

4.1.4.2 Treinamento do modelo de identificação da Etapa 2

No outro fluxo do treinamento geral, o conjunto de dados balanceado é submetido à etapa de seleção de atributos. Nesta fase, são encontrados atributos mais úteis para o processo de detecção multiclasse. Assim, para encontrar os melhores atributos e remover atributos irrelevantes do conjunto de dados, a proposta contempla o uso da estratégia de seleção de atributos apresentada anteriormente na Seção 4.1.3 (Página 91). Em seguida, o conjunto de dados com apenas os atributos selecionados é usado para treinar o classificador *ensemble* multiclasse.

O treinamento do método *ensemble* consiste no treinamento de cada uma das técnicas que o compõem. As técnicas RF e ET na abordagem proposta são compostas por 100 árvores de decisão, desse modo, nesta etapa cada técnica realiza o processo de construção das suas 100 árvores. O processo de treinamento dessas técnicas tende a ser mais custoso que o treinamento da ET binária da primeira etapa, pois a ET binária possui apenas 10 árvores.

O processo de treinamento do modelo DNN é o mesmo mencionado para a DNN da etapa binária. No entanto, neste caso, trata-se de uma arquitetura neural mais complexa, com mais neurônios, o que acarreta em um mair custo de treinamento. Esse processo pode se tornar oneroso ao se utilizar uma grande quantidade de dados de treinamento. No entanto, neste caso não foi necessário utilizar estratégias para reduzir o custo pois esta etapa da abordagem está inserida em um contexto de maior capacidade computacional.

Após o treinamento, é obtida uma estrutura *ensemble* capaz de realizar classificação multiclasse de novos dados, ela estará pronta para operar, recebendo dados capturados da rede para serem analisados e classificados. Durante a execução real, os dados são capturados, pré-processados para obter apenas atributos selecionados, padronizados e submetidos à abordagem treinada, que então detecta e identifica a categoria de tráfego.

4.2 ETAPA DE CONTRAMEDIDAS

A etapa de contramedidas será acionada quando o tráfego for detectado como intrusivo pelo módulo de análise. Esta seção fornece uma breve análise a respeito das estratégias de contramedidas empregadas nesse contexto de computação em nevoeiro e IoT.

Primeiramente, é importante destacar que além de detectar as intrusões, é muito importante executar ações de contramedidas para bloquear e evitar que a intrusão obtenha sucesso. Dentre as ações apresentadas por abordagens do estado da arte estão a emissão de alerta para o gerenciador da rede, descarte de pacotes e bloqueio de conexões. A emissão de apenas um alerta não configura uma ação de prevenção, pois, apenas deixa o gerenciador ciente que ocorreu uma intrusão, mas não a evita (Nguyen et al., 2019; Miranda et al., 2020).

Muitos dos trabalhos propuseram mecanismos de contramedidas para contextos específicos em que estão inseridos, para protocolos ou ataques específicos. Em Potrino, Rango e Santamaria (2019), foram propostas abordagens para mitigar ataques específicos contra o protocolo MQTT. São aplicadas um conjunto de restrições preventivas e políticas de descarte de pacotes de acordo com as especificações definidas pelos autores. Alguns trabalhos abordaram o uso de criptografia para mitigar certos tipos de ataques. Aliyu, Sheltami e Shakshuki (2018) utilizaram o *Advanced Encryption System* (AES), técnica de criptografia simétrica, enquanto a chave de criptografia é trocada usando a troca de chaves *Diffie-Hellman*, para evitar que as mensagens sejam modificadas por um ataque *Man-in-the-Middle* (MITM). No entanto, não são fornecidos detalhes sobre o processo de bloqueio do nó detectado como malicioso. Além disso, Potrino, Rango e Fazio (2019) propuseram criptografar os *payloads* MQTT usando *Elliptic Curve Cryptography* (ECC) para mitigar data *tampering* e *eavesdropping*. Além disso, Yaseen et al. (2018) propuseram mecanismos para ponderar as leituras de dispositivos sensores e mitigar ataques de conluio. Alguns trabalhos também utilizaram *firewalls* para bloquear e filtrar pacotes e acessos. Na abordagem FOCUS (Maharaja; Iyer; Ye, 2019), o *firewall* é usado para bloquear o acesso ao servidor de Rede Privada Virtual (*Virtual Private Network* - VPN) que protege a comunicação entre o nevoeiro e os dispositivos IoT de ataques DDoS. Zhou, Guo e Deng (2019) propôs uma abordagem para mitigação de DDoS onde os *firewalls* empregados executam as funções de filtragem de pacotes com base em regras, enquanto os servidores locais e em nuvem conduzem a funcionalidade de detecção de anomalias. Já a abordagem proposta pelos autores Sandhu, Sohal e Sood (2017) utiliza *Honeypot*. Ela consiste em uma armadilha criada pelo administrador do sistema, uma réplica do ambiente real para atrair intrusos ao sistema.

Uma das principais estratégias encontradas no estado da arte para a mitigação de ataques é o descarte de pacotes após uma detecção de intrusão. Na solução proposta por Ravi e Shalinie (2020), quando um ataque é detectado o módulo de mitigação é acionado e uma regra de descarte é definida no nevoeiro para o par de endereços MAC-IP do dispositivo que enviou os pacotes. Além disso, um temporizador aleatório é definido. Depois que o cronômetro expira, a regra de descarte é revogada. O cronômetro deve ser aleatório para que o invasor não possa

deduzir a estratégia de tempo de mitigação (Ravi; Shalinie, 2020). Quando não há uma classificação clara sobre o tráfego, a regra de descarte é definida para um tempo aleatório relativamente pequeno. No entanto, não são fornecidos maiores detalhes de como o filtro é implementado. A abordagem proposta por Tu et al. (2018) rejeita pacotes de um determinado canal de comunicação entre o dispositivo e o nevoeiro quando detecta um ataque de personificação. Em Pan, Pacheco e Hariri (2017) é proposto uma abordagem com medidas de mitigação de ataques no contexto de automação de prédios e computação em nevoeiro. As ações de proteção incluem descartar pacotes, suspender a conexão usada pelo fluxo de tráfego e descartar o tráfego. A fase final da operação do módulo é traduzir o alerta, bem como a decisão de ação tomada na fase anterior, em uma mensagem de alarme compreensível. Os autores Lawal, Shaikh e Hassan (2020) também apresentam um *framework* que conta com um módulo de bloqueio e descarte de pacotes identificados como intrusivos, além do envio de mensagem de alerta. No entanto, não são apresentados detalhes de como as ações dessas abordagens são implementadas.

Pacheco et al. (2020) apresentam como contra-medidas o reinício de conexões e sistema e até mesmo alterações de configuração. A abordagem depende de níveis de ação. Inicialmente, a ação de reinicialização da conexão é usada. Se o problema persistir sempre que a conexão for renovada, outras ações serão necessárias. O manipulador de ações emprega um arquivo de *log* para manter um registro de cada erro, incluindo seu carimbo de data/hora. Antes de aplicar qualquer política de proteção, o arquivo de *log* é revisado, procurando a frequência de um determinado erro. Se uma alta frequência for encontrada, ações são realizadas para alterar a configuração do sistema e solicitar autenticação. Os autores Zahra e Chishti (2020) propuseram uma abordagem para responder às intrusões detectadas com base em quão prejudiciais elas podem ser. Essa decisão é tomada por uma base de conhecimento difusa. Dependendo das prioridades de uma organização, ela atribuirá peso às regras e dará uma decisão sobre a resposta do sistema. Nesse contexto, foi proposto como regra para a mitigação de ataques de conluio e DDoS, a inserção do nó detectado como intrusivo na *blacklist* mantida na nuvem e a geração de alertas à toda a rede para bloqueio de comunicação com o nó detectado.

Por fim, alguns trabalhos encontrados apresentaram estratégias promissoras baseadas em Redes Definidas por Software (*Software Defined Networks* - SDN) para a mitigação de intrusão em ambientes de computação em nevoeiro e IoT (Shafi et al., 2018; Nguyen et al., 2019; Miranda et al., 2020; Priyadarshini; Barik, 2022).

As SDNs são abordagens de redes programáveis capazes de separar os planos de dados e de controle através de *interfaces* padronizadas (Haleplidis et al., 2015). Em redes tradicionais, esses planos se encontram juntos e não há uma clara diferenciação dos dois. Na SDN, um dispositivo de rede, como um *switch*, é dividido em dois planos: o plano de controle e o de dados. O plano de controle é responsável pela inteligência, é ele quem executa os códigos dos protocolos de roteamento, por exemplo. Já o plano de dados é responsável por encaminhar os pacotes entrantes no dispositivo, ele simplesmente os encaminha para o destino determinado pelo plano de controle. Essa separação permite que o controlador seja facilmente modificado e atualizado para alterar todo o comportamento da rede. O controlador é programável e elimina a

necessidade de configuração prévia do dispositivo físico da rede, tornando-o reutilizável (McKeown et al., 2008). Os dispositivos de rede responsáveis pelo plano de dados possuem uma tabela de fluxos. Ela é atualizada pelo controlador. Quando chega um pacote que possui fluxo já definido na tabela, ele é comutado pelo dispositivo de acordo com o determinado fluxo. Por outro lado, quando um pacote chega e não possui fluxo definido na tabela, o controlador é acionado para definir qual será o próximo destino desse pacote, de acordo com as aplicações que foram programadas. O controlador então adiciona esse novo fluxo na tabela do dispositivo de rede (McKeown et al., 2008). O controlador (plano de controle) e os dispositivos de rede (plano de dados) que realizam o encaminhamento dos pacotes, se comunicam através de um protocolo, o *OpenFlow* é um dos principais. O protocolo *OpenFlow* é a base para a criação de uma SDN aberta e baseada em padrões. A separação existente entre os planos permite que o controlador possa ser modificado e atualizado facilmente para efetuar alterações no comportamento da rede. Cada aplicação pode definir como o gerenciamento se dará e não fica limitada a configuração prévia de um equipamento de rede (McKeown et al., 2008). De acordo com (Krishnan; Duttagupta; Achuthan, 2020), a SDN permite moldar o tráfego em uma rede para definir caminhos programaticamente, isolar dispositivos de rede não seguros, controlar a trajetória dos pacotes para passar por um dispositivo ou uma cadeia de serviços, e assim por diante. Os estudos em relação a mitigação de ataques com SDN utilizam essa característica programável para implementar ações de contramedidas dinamicamente.

Nas abordagens encontradas no estado da arte, geralmente, quando o controlador SDN recebe um pacote da rede IoT, ele o envia para o módulo de análise. Se o pacote for identificado como não intrusivo, o controlador SDN libera o tráfego. Se o pacote for identificado como intrusivo, o controlador SDN implementa contramedidas por meio de regras na tabela de fluxo (Shafi et al., 2018; Nguyen et al., 2019; Priyadarshini; Barik, 2022; Miranda et al., 2020).

Os autores Shafi et al. (2018) propuseram uma abordagens baseada em SND onde o controlador é inserido na camada de computação em nevoeiro. A abordagem conta com um método de aprendizado de máquina para realizar a detecção de ataques. Quando um ataque é detectado, o controlador SDN é invocado para instalar dinamicamente medidas preventivas. O controlador SDN instala regras de fluxo apropriadas nos *switches* da borda da rede para controlar a interrupção do tráfego de ataque. De acordo com os autores, a rede definida por software forneceu meios e formas de racionalizar a mitigação de ameaças. O sistema proposto propõe-se a atender os requisitos de escalabilidade, é possível adicionar mais recursos computacionais no caso do grande aumento no número de dispositivos de IoT.

Na abordagem proposta por Priyadarshini e Barik (2022), a camada do nevoeiro é responsável por inspecionar todos os pacotes da rede IoT. O controlador SDN foi inserido na camada de nevoeiro para gerenciar toda a rede. Quando um pacote é detectado como intrusivo todas as suas informações são enviadas e o controlador executa ações para impedir que esse pacote entre na rede, como bloquear o endereço IP na tabela de fluxo. O trabalho foca apenas na mitigação de DDoS e apresenta um arquitetura suscetível a problemas de ponto único de falha.

O autores Li et al. (2018) propuseram uma abordagem baseada em SDN que extrai recursos de todos os pacotes transmitidos ao *switch OpenFlow* e constrói uma matriz de recursos, a qual é utilizada por um modulo de aprendizado profundo para detectar ataques DDoS. Para os pacotes detectados como intrusivos, o módulo *Flow Table Generator* determina regras de fluxo no *switch OpenFlow*. O *switch OpenFlow* executa a tabela de fluxo e descarta os pacotes de acordo com as regras definidas.

A abordagem *SeArch* (Nguyen et al., 2019) apresenta uma arquitetura composta por um arranjo hierárquico de três camadas. Os controladores SDN, responsáveis pelo gerenciamento da rede, são implantados na camada de nevoeiro. Estas camadas trabalham em colaboração para detectar anomalias e formular políticas nos dispositivos de *gateway* IoT baseados em SDN, para interromper o tráfego malicioso o mais rápido possível. Se um ataque for detectado, são instaladas regras de fluxo no *gateway* IoT baseado em SDN afetado, para uma reação rápida aos ataques. No caso de ataques de negação de serviço, a política padrão é a implementação de uma regra e a definição de um valor de *hard_timeout* para descartar todos os pacotes da fonte de ataque. Além disso, a arquitetura apresentada busca fornecer solução baseada em balanceamento de carga para suportar o grande volume de trafego gerado por ataques.

Rathore, Wook Kwon e Park (2019) propuseram uma abordagem de detecção com aprendizagem profunda em redes IoT-Fog baseadas em SDN para mitigar ataques. Os autores implementam algoritmos de aprendizado de máquina no controlador SDN para analisar o tráfego. Este trabalho trata vários controladores SDN como dispositivos de nevoeiro, e um servidor central em nuvem é usado como controlador SDN raiz para atuar como gerenciador. Ao detectar uma intrusão o controlador implementa regras na tabela de fluxos para descartar o tráfego. Essa arquitetura, assim como as demais baseadas em SDN, se mostra interessante para ser utilizada como base em trabalhos futuros para a implantação da abordagem de detecção e identificação de intrusão proposta neste trabalho. O módulo de detecção do primeiro nível poderia ser implantado no controlador SDN do nevoeiro e o módulo de identificação do segundo nível poderia ser implantado no controlador SDN da nuvem. Desse modo, ao receber um pacote vindo da rede IoT o controlador SDN o enviaria para o módulo de análise para que fosse executada a primeira etapa de detecção. Caso o pacote fosse detectado como não intrusivo o controlador SDN o liberaria pelo fluxo normal. Caso o pacote fosse identificado como intrusivo ele seria enviado para a etapa de identificação no controlador SDN na nuvem, onde seria realizada uma análise multiclasse para identificar o tipo do ataque. A partir da análise multiclasse, o método de contramedidas, através do controlador SDN poderia executar as ações necessárias para mitigar o determinado tipo de ataque por meio da inserção de regras na tabela de fluxos do *switch* SDN.

Apesar das arquitetura proposta em Rathore, Wook Kwon e Park (2019) permitir a implementação de ações de contramedidas dinamicamente e se mostrar promissora para a implantação da abordagem de detecção e identificação proposta nesse trabalho, é importante refletir sobre alguns pontos de atenção importantes ao se projetar estratégias de contramedidas.

Primeiramente, é importante considerar os problemas das abordagens de detecção ba-

seadas em anomalias relacionados a falsos positivos, pois nem todo comportamento anômalo é uma intrusão. É evidente no estado da arte que a as abordagens de detecção de intrusão baseadas em anomalia apresentam problemas relacionados a falsos positivos (Ieracitano et al., 2020; Dat-Thinh; Xuan-Ninh; Kim-Hung, 2022). Portanto, um ponto de atenção importante nas abordagens de mitigação de ataques é que o descarte de pacotes e bloqueio de conexões de todo o tráfego detectado como intrusivo podem prejudicar o funcionamento da rede em cenários nos quais a abordagem de detecção gera uma alta taxa de falsos positivos. Pois, quando uma abordagem de detecção baseada em anomalia identifica erroneamente um tráfego benigno como intrusivo, as mitigações de ataques baseadas em bloqueio e descarte de pacotes irão bloquear imediatamente esse tráfego legítimo. Nesse cenário, uma abordagem de detecção com problemas de falsos positivos pode resultar no bloqueio de uma parte significativa da rede.

As abordagens que realizam apenas a detecção binária direcionarão todo o tráfego detectado como intrusivo para um único conjunto de ações de contramedidas, não possibilitando a separação do tratamento de mitigação em fluxos específicos para o tipo de ataque detetado. Assim, geralmente a ação realizada é o descarte do pacote e o bloqueio de conexão (Ravi; Shalinie, 2020; Lawal; Shaikh; Hassan, 2020). Nesse cenário, os problemas de falsos positivos poderão prejudicar o funcionamento da rede pois muitos pacotes serão bloqueados erroneamente.

Nesse ponto, destaca-se a importância da identificação dos tipos de ataques, pois a partir dessa identificação torna-se possível executar ações específicas para cada tipo de ataque detectado. Nesse contexto, a detecção multiclasse, aliada a uma arquitetura SDN, permite a implementação dinâmica de ações de contramedidas. Desse modo, em trabalho futuros, poderia considerar que quando o módulo de análise detectar um ataque de negação de serviço, o controlador poderia implementar regras de bloqueio do tráfego da origem de ataque por um período de tempo, seguindo uma estratégia similar à proposta em Ravi e Shalinie (2020). Por outro lado, alguns tipos de ataques podem ter uma ação de contramedida menos drástica, por exemplo através de direcionamento para mecanismos como *Honeypot* (Sandhu; Sohal; Sood, 2017) ou ativação de módulos de detecção especializados. Por exemplo, uma ameaça de sondagem geralmente é executada antes de ataques mais complexo e poderosos, como DDoS, ataques de acesso remoto, etc (Nguyen et al., 2019). Assim, ao detectar uma ameaça da categoria de sondagem, pode ser interessante executar mecanismos de detecção adicionais para reforçar a segurança. Nesse cenário, a arquitetura baseada em SDN seria muito interessante para, por exemplo, através do controlador SDN implementar regras, válidas por um período de tempo predefinido, na tabela de fluxo do *switch* SDN, as quais direcionariam o tráfego proveniente da fonte de ataque para um módulo de detecção especializado.

Outro aspecto importante de uma estratégia de contramedidas é a tradução da detecção, bem como a decisão de ação tomada para mitigação, em uma mensagem de alarme compreensível para alertar o gerenciador da rede (Pan; Pacheco; Hariri, 2017). Portanto, essa etapa também requer informações mais detalhadas sobre o tráfego detectado para gerar um alerta com informações mais precisas sobre a intrusão. A simples detecção binária do tráfego como intrusivo, por ser anômalo em relação ao comportamento esperado, não é suficiente, pois não fornece

maiores informações para o gerenciador da rede. Essa identificação do ataque e emissão de um alerta compreensível são importantes para o gerenciador da rede tomar decisões. A partir da identificação da categoria de um determinado ataque que ocorre com uma frequência específica, o responsável pela rede pode decidir implementar ações para corrigir a vulnerabilidade explorada pela intrusão.

Apesar de fornecer uma maneira promissora de realizar a implementação de contramedidas no contexto de computação em nevoeiro e IoT, a SDN possui desafios. Segundo os autores Javanmardi et al. (2023), os principais estão relacionados ao consumo de recursos do nevoeiro e a segurança do controlador. O consumo de recursos foi abordado por (Nguyen et al., 2019) que apresentou solução baseada em balanceamento de carga para que o controlador possa suportar o grande volume de tráfego gerado por ataques. Por outro lado, em arquitetura baseada em SDN o controlador pode se tornar um alvo para usuários mal-intencionados, pois possui um papel crucial nesse contexto. Uma das principais ameaças contra os controladores são os ataques DDoS. Pois todo o sistema irá funcionar mal ou será corrompido se o controlador SND falhar ou for comprometido por ataques DDoS. Outro ponto que pode ser alvo de ataques é a comunicação entre o plano de controle e o plano de dados. Como cada comando do controlador é retransmitido por meio desse canal para o plano de dados, uma vez violado, o invasor terá controle total sobre a rede. Portanto, a infraestrutura SDN das redes IoT-Fog deve ser protegida contra ameaças como ataques de homem do meio (Javanmardi et al., 2023). Portanto, a segurança do controlador SDN também é um aspecto que precisa ser trabalhado em pesquisas futuras.

5 AVALIAÇÃO

Neste capítulo é apresentada a metodologia definida para a avaliação da proposta. A abordagem proposta é avaliada em relação às técnicas clássicas de aprendizado de máquina e aos trabalhos do estado da arte, através de experimentos com conjuntos de dados públicos de detecção de intrusão. Por fim, é apresentada uma análise da complexidade de tempo envolvida na solução proposta. A seguir, são apresentados as métricas de avaliação consideradas nos experimentos.

5.1 MÉTRICAS DE AVALIAÇÃO

A avaliação dos métodos de detecção é imprescindível para examinar a viabilidade de aplicá-los em ambiente real. A partir dos experimentos realizados é possível categorizar as classificações dos eventos da base de dados nos termos apresentados a seguir.

- **Falso Negativo (FN):** Eventos classificados como normais pelo método de detecção e que são intrusões;
- **Falso Positivo (FP):** Eventos não intrusivos e classificados como intrusivos pela técnica de detecção de intrusão;
- **Verdadeiro Negativo (VN):** Esta categoria abrange eventos não intrusivos, que foram corretamente classificados pelo método de detecção;
- **Verdadeiro Positivo (VP):** Esta classe inclui eventos relatados corretamente como intrusos pela técnica de detecção de intrusão.

A partir desses termos é possível construir matrizes de confusão. Elas consistem em tabelas que apresentam um resumo da classificação realizada pelo método, apontando o número de eventos classificados em relação a classe predita e a verdadeira classe do elemento. A matriz de confusão por si não é uma métrica de avaliação de métodos de classificação, no entanto, diversas análises e métricas podem ser feitas a partir de tais informações. Na Tabela 2 é possível observar um exemplo de matriz de confusão.

Classe	**Classe Predita**	
	Normal (+)	Ataque (-)
Normal (+)	VN	FP
Ataque (-)	FN	VP

Tabela 2 – Exemplo de uma matriz de confusão.

Com base nos termos supracitados é possível realizar o cálculo de diferentes métricas que auxiliam na avaliação dos métodos de detecção construídos por algoritmos de aprendizado

de máquina. A seguir são apresentados as métricas utilizadas para avaliar os métodos neste trabalho (Liu; Lang, 2019a).

1. **Acurácia (ACC):** esta métrica corresponde a proporção de instâncias classificadas corretamente em relação ao total de instâncias existentes. Pode não ser um bom aspecto a ser considerado em casos de grande desbalanceamento de classes. A acurácia é calculada a partir da Equação 5.1, apresentada a seguir:

$$ACC = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

2. **Precisão (PRE):** outra métrica amplamente usada para avaliar métodos de aprendizado de máquina. Essa taxa indica a proporção de instâncias corretamente detectadas como intrusivas dentre todas as detectadas como intrusivas, como pode ser visto na Equação 5.2.

$$PRE = \frac{VP}{VP + FP} \tag{5.2}$$

3. ***Recall* ou *True Positive Rate* (TPR):** também conhecida como sensibilidade, consiste no número de instâncias corretamente classificadas como intrusivas dentre todos as instâncias intrusivas, é calculado de acordo com a Equação 5.3.

$$Recall = \frac{VP}{VP + FN} \tag{5.3}$$

4. ***True Negative Rate (TNR)*:** também chamada de especificidade, é o número de instâncias classificadas como não intrusivas, entre todas as instâncias não intrusivas. É determinado de acordo com a Equação 5.4.

$$TRN = \frac{VN}{VN + FP} \tag{5.4}$$

5. **F1-Score:** A pontuação F1 (também pontuação F ou medida F) é uma medida da precisão de um teste. A pontuação F1 é a média harmônica de precisão e recuperação, onde uma pontuação F1 atinge seu melhor valor em 1 (precisão e recuperação perfeitas) e pior em 0. A fórmula para a pontuação F1 é apresentada na Equação 5.5.

$$F1 - Score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \tag{5.5}$$

6. ***Matthews Correlation Coefficient* (MCC):** MCC é usado no aprendizado de máquina como uma medida da qualidade das classificações binárias. O MCC tem um intervalo de -1 a 1, onde -1 indica um classificador binário completamente errado, enquanto 1 indica um classificador binário completamente correto. Ele é apresentado na Equação 5.5.

$$MCC = \frac{VP * VN - FP * FN}{\sqrt{(VP+FP) * (FN+VN) * (FP+VN) * (VP+FN)}} \tag{5.6}$$

7. **Balanced Accuracy (BACC):** A Acurácia Balanceada é uma métrica interessante para avaliar o desempenho da detecção em conjuntos de dados não balanceados. É definido como a média do *recall* obtida em cada classe, conforme pode ser observado na Equação 5.7. Onde R_i corresponde a ao *recall* (R) obtido considerando a i-ésima (i) classe presente no conjunto de dados e n é a quantidade de classes existentes.

$$Acurácia \quad Balanceada = \frac{\sum_{i=1}^{n} R_i}{n} \tag{5.7}$$

Além das métricas de classificação, também são consideradas neste trabalho, medidas a respeito do tempo necessário para realizar determinadas tarefas, como:

1. **Tempo de treino:** corresponde ao tempo, em segundos, que o algoritmo precisou para realizar o treinamento com o conjunto de treinamento correspondente.

2. **Tempo de teste:** corresponde ao tempo, em segundos, que o algoritmo precisou para realizar a predição das instâncias do conjunto de teste.

Para realizar a avaliação, também são aplicados métodos estatísticos com o objetivo de verificar a existência de diferenças estatísticas significativas entre os resultados dos experimentos. O objetivo da análise é comparar grupos de 5 resultados por abordagem, para verificar se existem diferenças significativas entre eles. Primeiramente, para definir qual método estatístico de significância utilizar, é necessário verificar a normalidade dos dados. Para isso, é utilizado o método *Shapiro-Wilk* (Shapiro; Wilk, 1965). De acordo com o método, utiliza-se a seguinte formulação de hipótese:

- H_0: A amostra possui uma distribuição normal.

- H_1: A amostra não possui uma distribuição normal.

Estes testes presumem inicialmente que a amostra faz parte de uma distribuição normal (hipótese nula ou H_0). O nível de significância, também denotado como alfa ou α, é a probabilidade de rejeição da hipótese nula quando ela é verdadeira. Um nível de significância de $0,05$ indica um risco de 5% de concluir que não possui distribuição normal quando na realidade possui. Desse modo, é definido um nível de significância (α) de 5% ($0,05$), com intervalo de confiança de 95%. Caso $p - value <= \alpha$, H_0 é rejeitada, caso $p - value > \alpha$, não há evidência significativa para rejeitar H_0. O teste de normalidade aplicado aos dados no processo de avaliação foi o *Shapiro-Wilk*.

Após a verificação da normalidade dos dados, utiliza-se testes de significância para verificar se existe diferenças estatísticas significativas entre as amostras. No caso de dados que

seguem uma distribuição normal, aplicam-se testes paramétricos, caso não haja conhecimento prévio do conjunto de dados, deve ser utilizados testes não-paramétricos. Dentre os testes observados, considerando a natureza não pareada dos dados e a existência de mais de dois conjuntos de amostras para comparação, as opções de testes considerados foram:

- Análise de Variância (ANOVA): Teste paramétrico que verifica se a média entre 2 ou mais grupos é significativamente diferente, utilizado para amostras independentes (Fisher, 1992).

- Teste de *Kruskal-Wallis*: Teste não-paramétrico, aplicado para comparar 2 ou mais amostras independentes (Kruskal; Wallis, 1952).

Assim como no testes estatísticos de normalidade, os testes ANOVA e Kruskal-Wallis são testes de hipótese. Portanto, considerou-se as seguintes hipóteses.

- H_0: Não há diferença entre os grupos ou médias.

- H_1: Existe pelo menos uma das médias que é diferente.

Nesta analise, também é utilizado o nível de significância de $0,05$ ($\alpha = 0,05$). Para ambos os casos, considera-se que se $p - value < \alpha$ rejeita-se H_0 e conclui-se que há diferença estatística significativa entre os grupos. Por outro lado, com um $p - value > \alpha$, não há evidência significativa para rejeitar H_0, conclui-se que não é possível dizer se há diferença estatística significativa entre os grupos.

A aplicação de um teste de significância como o ANOVA ou *Kruskal-Wallis* identifica se existem diferenças estatísticas significativas entre 2 ou mais amostras. No entanto, para verificar quais das amostras diferem, é necessário aplicar uma técnica de pós-teste. Para realizar esta comparação podem ser utilizados métodos como o pós-teste de Dunn (Dunn, 1961) e o de Tukey (Tukey, 1949).

A seguir são apresentadas os conjuntos de dados de detecção de intrusão utilizados nos experimentos deste trabalho.

5.2 BASE DE DADOS

Nesta seção, são apresentados detalhes a respeito dos três conjuntos de dados utilizados para avaliar a abordagem proposta. O objetivo ao utilizar múltiplos conjuntos de dados é validar se a abordagem de detecção é capaz de trabalhar em diferentes cenários. Os conjuntos de dados utilizados foram: NSLKDD, IoTID20 e NF-UQ-NIDSv2. A seguir, são apresentadas as informações a respeito do conjunto de dados NSLKDD.

5.2.1 NSL-KDD

As bases de dados KDD CUP 99[1] e a DARPA[2] são bem conceituadas no contexto de detecção de intrusão. Elas possuem tanto tráfego de ataques quanto atividades normais capturadas de redes de computadores. Estudos realizados nestas bases observaram que elas possuem algumas deficiências, como grande amostras redundantes e atributos excessivos (McHugh, 2000). Com o intuito de resolver tais problemas, os pesquisadores Tavallaee et al. (2009) propuseram uma nova base denominada NSL-KDD[3]. Ela foi construída com registros selecionados do conjunto completo de dados da KDD CUP 99, buscando sanar os problemas mencionados anteriormente. A base NSL-KDD é constituída de 125973 registros.

Cada exemplo da base NSL-KDD possui 42 atributos, sendo 41 deles relacionados às características do comportamento na rede e 1 relacionado ao rótulo desse exemplo, indicando se se trata de algum tipo de ataque ou de comportamento normal. Os rótulos presentes no conjunto de dados estão detalhados na Tabela 3, que também apresenta a quantidade de instâncias por rótulo.

Tabela 3 – Distribuição dos dados por classe na NSLKDD.

Categorias	**Quantidade**
Benigno	67343
Denial of Service (DoS)	45927
Probing (Probe)	11656
Remote to Local (R2L)	995
User to Root (U2R)	52

Os atributos de características do tráfego podem ser divididos em 4 categorias: atributos básicos, atributos de conteúdo, atributos de tráfego baseados em tempo e atributos de tráfego baseados em conexão. A Tabela 4 apresenta os atributos que podem ser extraídos de uma conexão TCP/IP sem considerar a carga útil do pacote, esses atributos são chamados de atributos básicos.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] https://ll.mit.edu/ideval/data/
[3] http://www.unb.ca/cic/datasets/nsl.html

Tabela 4 – Atributos básicos capturados de conexão TCP.

#	Atributo	Descrição	Tipo
1	duration	Duração da conexão	Contínuo
2	protocol_type	Tipo de protocolo (tcp, udp, etc)	Discreto
3	service	Tipo de Serviço de Destino (HTTP, Telnet)	Discreto
4	flag	Estado da Conexão	Discreto
5	src_bytes	Números de bytes da origem ao destino	Contínuo
6	dst_bytes	Número de bytes do destino à origem	Contínuo
7	land	1 se o Host e a porta da origem e destino são os mesmos, 0 caso contrário	Discreto
8	wrong_fragment	Número de fragmentos errados	Contínuo
9	urgent	Número de Pacotes Urgentes	Contínuo

A Tabela 5 apresenta os atributos extraídos da carga útil do pacote TCP para observar comportamentos suspeitos dentro do segmento de carga útil, esses atributos são denominados como atributos de conteúdo.

Tabela 5 – Atributos de conteúdo.

#	Atributo	Descrição	Tipo
10	hot	Número de ações "hot"em uma conexão, como: inserir um diretório do sistema, criar e executar programas	Contínuo
11	num_failed_logins	Número de tentativas de login com falha	Contínuo
12	logged_in	1 se o login obteve sucesso e 0 Caso contrário	Discreto
13	num_compromised	Número de condições comprometidas	Contínuo
14	root_shell	1 se o shell root é obtido 0 caso contrário	Discreto
15	su_attempted	1 se houver tentativa de conseguir “su root” 0 caso contrário	Discreto
16	num_root	Número de acessos como root	Contínuo
17	num_file_creations	Número de operações de criação de arquivos	Contínuo
18	num_shells	Número de shell prompts abertos	Contínuo
19	num_access_files	Número de operações a arquivos de controle de acesso	Contínuo
20	num_outbound_cmds	Números de comandos externos (sessão FTP)	Contínuo
21	is_hot_login	1 se o login pertence à lista “hot”, 0 caso contrário	Discreto
22	is_guest_login	1 se o login é do tipo “guest”, 0 caso contrário	Discreto

O restante dos atributos da base corresponde a estatísticas relacionadas ao tráfego anterior à conexão atual e são denominados como atributos de tráfego. Esses atributos podem ser divididos em dois grupos: atributos de tráfego baseados no tempo, que levam em consideração as conexões mais recentes dentro de um intervalo de tempo, e atributos de tráfego baseados em conexões, que consideram um número específico de conexões mais recentes. Os atributos de tráfego, de maneira geral, representam estatísticas sobre o tráfego, porém levam em consideração dois pontos principais:

- Atributos de mesmo *host*: são levadas em consideração apenas as conexões que possuem o mesmo *host* que a conexão atual.

- Atributos de mesmo serviço: são levadas em consideração apenas as conexões que possuem o mesmo serviço que a conexão atual.

Os atributos presentes na Tabela 6 pertencem ao grupo de atributos de tráfego baseados em tempo, levando em consideração as conexões do intervalo dos últimos 2 segundos.

Tabela 6 – Atributos de tráfego baseados em tempo.

#	Atributo	Descrição	Tipo
23	Count	Número de conexões para o mesmo host da conexão atual nos últimos 2 segundos	Contínuo
24	srv_count	Número de conexões ao mesmo serviço da conexão atual nos últimos 2 segundos	Contínuo
25	serror_rate	% de conexões que tiveram erros do tipo “SYN”, dentre as conexões consideradas em count (23)	Contínuo
26	srv_serror_rate	% de conexões que tiveram erros “SYN”, dentre as conexões consideradas em srv_count (24)	Contínuo
27	rerror_rate	% de conexões que tiveram erros do tipo “REJ”, dentre as conexões consideradas em count (23)	Contínuo
28	srv_rerror_rate	% de conexões que tiveram erros “REJ”, dentre as conexões consideradas em srv_count (24)	Contínuo
29	same_srv_rate	% de conexões ao mesmo serviço, dentre as conexões consideradas em count (23)	Contínuo
30	diff_srv_rate	% de conexões a diferentes serviços, dentre as conexões consideradas em count (23)	Contínuo
31	srv_diff_host_rate	% de conexões a diferentes hosts, dentre as conexões consideradas em srv_count (24)	Contínuo

No entanto, existem vários ataques de sondagem lenta que exploram os *hosts* (ou portas) utilizando um intervalo de tempo muito maior do que 2 segundos, por exemplo, um a cada minuto. Como consequência, esses ataques não geram padrões de intrusão com uma janela de tempo de 2 segundos. Para solucionar essa problemática, os atributos "mesmo *host*"e "mesmo serviço"são recalculados, porém com base em uma janela de conexão de 100 conexões, em vez de uma janela de tempo de 2 segundos. Esses atributos estão detalhados na Tabela 7 e fazem parte do grupo de atributos de tráfego baseados em conexão, considerando as últimas 100 conexões.

Tabela 7 – Atributos de tráfego baseados em conexão.

#	Atributo	Descrição	Tipo
32	dst_host_count	Número de conexões para o mesmo host da conexão atual dentre as últimas 100 conexões	Contínuo
33	dst_host_srv_count	Número de conexões ao mesmo serviço da conexão atual dentre as últimas 100 conexões	Contínuo
34	dst_host_same_srv_rate	% de conexões ao mesmo serviço, dentre as conexões consideradas em count (32)	Contínuo
35	dst_host_diff_srv_rate	% de conexões a diferentes serviços, dentre as conexões consideradas em count (32)	Contínuo
36	dst_host_same_src_port_rate	% de conexões a mesma porta de origem, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo
37	dst_host_srv_diff_host_rate	% de conexões a diferentes hosts, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo
38	dst_host_serror_rate	% de conexões que tiveram erros do tipo “SYN”, dentre as conexões consideradas em dst_host_count (32)	Contínuo
39	dst_host_srv_serror_rate	% de conexões que tiveram erros “SYN”, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo
40	dst_host_rerror_rate	% de conexões que tiveram erros do tipo “REJ”, dentre as conexões consideradas em dst_host_count (32)	Contínuo
41	dst_host_srv_rerror_rate	% de conexões que tiveram erros do tipo “REJ”, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo

A Tabela 8 apresenta os valores que os atributos categóricos podem possuir. Os atributos como *land*, *logged_in*, *root_shell*, *su_attempted*, *is_host_login*, *is_guest_login* podem possuir dois valores, 0 ou 1. Outros recursos, como o tipo de protocolo, o serviço e a *flag*, possuem mais de dois valores diferentes. O atributo *protocol_type* possui 3 valores distintos, o atributo *flag* possui 11 valores distintos e o atributo *service* possui 66 valores distintos.

Tabela 8 – Valores dos atributos categóricos.

Atributos categóricos	**Valores**	**#. de valores**
protocol_type	tcp, udp, icmp	3
service	smtp, ntp_u, shell, kshell, imap4, urh_i, netbios_ssn, ftp_u, mtp, uucp, nnsp, echo, tim_i, ssh, iso_tsap, time, netbios_ns, systat, hostnames, login, efs, supdup, http_8001, courier, ctf, finger, nntp, ftp_data, red_i, ldap, http, ftp, pm_dump, exec, klogin, auth, netbios_dgm, ...	66
flag	RSTR, S3, SF, RSTO, SH, OTH, S2, RSTOS0, S1, S0, REJ	11
land	0, 1	2
logged_in	0, 1	2
root_shell	0, 1	2
su_attempted	0, 1	2
is_host_login	0, 1	2
is_guest_login	0, 1	2

As instâncias da base estão classificadas em comportamento normal, ou em ataque. A classificação de ataque pode ser de 22 tipos diferentes, listados na Tabela 9. Estes tipos de ataque podem ser agrupados em 4 categorias principais: *Denial of service* (DoS), *User to root* (U2R), *Remote to local* (R2L) e *Probing* (PROBE).

Tabela 9 – Tipos de ataques.

Denial of service (DoS)	**User to root (U2R)**	**Remote to local (R2L)**	**Probing (PROBE)**
Back	Perl	FTP write	IP sweep
Ping of death	Buffer overflow	Guess password	NMAP
Neptune	Load module	IMAP	Port sweep
Smurf	Rootkit	Multi HOP	Satan
Land		Phf	
Teardrop		SPY	
		Wareclient	
		Warezmaster	

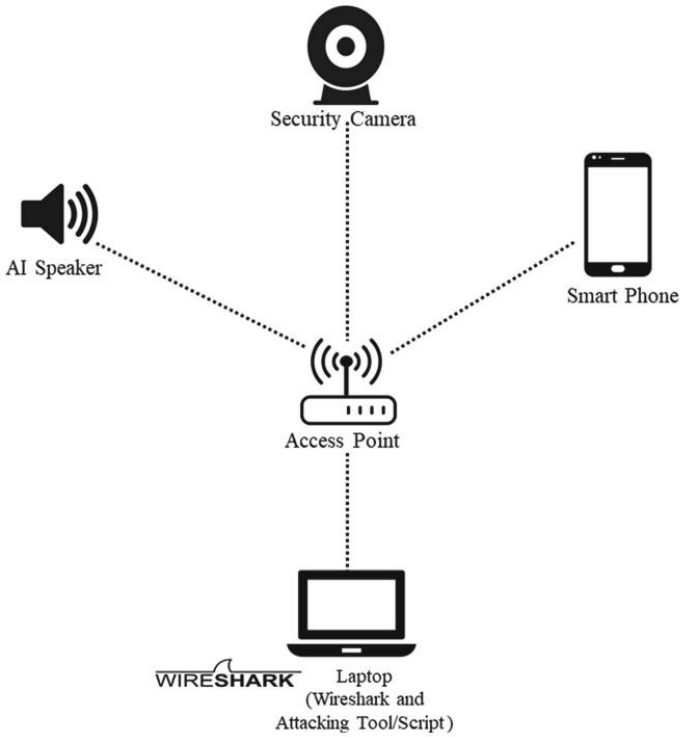
5.2.2 IoTID20

A IoTID20 é uma das mais recentes bases de detecção de intrusão focadas em dispositivos IoT. Ela foi gerada por meio da combinação de dispositivos IoT e estruturas de interconexão, simulando um ambiente típico de casa inteligente (*Smart Home*). A base inclui dois dispositivos IoT principais: um alto-falante inteligente SKT NGU e uma câmera Wi-Fi EZVIZ (Ullah; Mahmoud, 2020).

Esses dois dispositivos IoT foram conectados a um roteador Wi-Fi doméstico, que, por sua vez, estabelece a interconexão com outros dispositivos presentes na *Smart Home*, como

laptops, *tablets* e *smartphones*. Os dispositivos IoT SKT NGU e EZVIZ desempenham o papel de dispositivos vítimas, enquanto os demais são considerados como dispositivos maliciosos. A Figura 15 ilustra uma versão simplificada da arquitetura do ambiente de ensaio.

Figura 15 – Arquitetura do ambiente monitorado para criação do conjunto de dados IoTID20.



Fonte: (Ullah; Mahmoud, 2020)

O conjunto de dados IoTID20 possui 80 atributos de características de rede e três atributos correspondentes a rótulos. O primeiro rótulo é binário, o segundo refere-se a categorias de ataques e o último a subcategorias de ataques. A Tabela 10 exibe o número de instâncias de tráfego presente em cada tipo de rótulo.

Tabela 10 – Instâncias de tráfego normal, tráfego anômalo e sua classe e subclasse.

Binário	**Instâncias**	**Categoria**	**Instâncias**	**Subcategoria**	**Instâncias**
Benigno	40073	Benigno	40073	Benigno	40073
Anômalo	585710	DoS	59391	DoS	59391
		Mirai	415677	*Ack Flooding*	55124
				Brute force	121181
				HTTP Flooding	55818
				UDP Flooding	183554
		MITM	35377	MITM	35377
		Scan	75265	*Host Port*	22192
				Port OS	53073

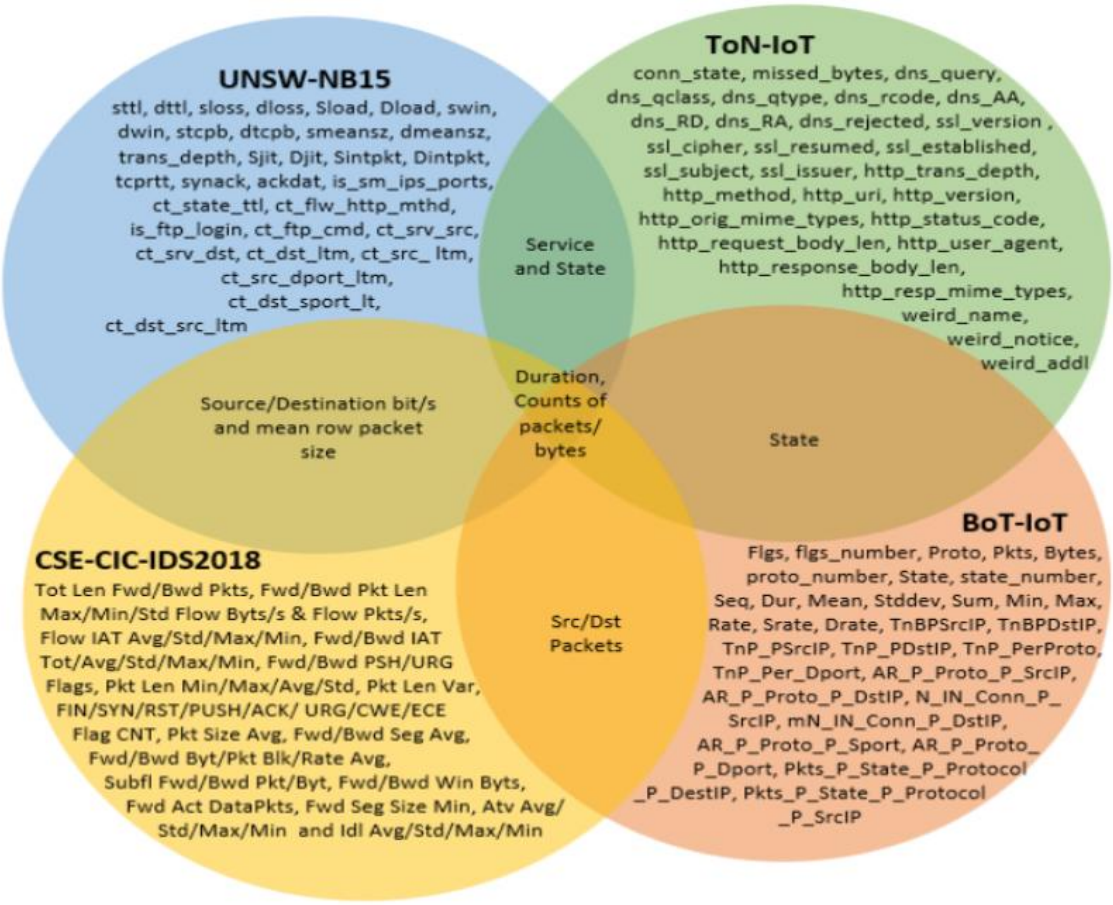
Nesse trabalho foi utilizado o rótulo “Categoria”, isto é, o tráfego é classificado em benigno ou em alguma categoria de ataque, ou seja, DoS, Mirai, MITM ou Scan. Os outros dois rótulos são retirados do conjunto de dados.

5.2.3 NF-UQ-NIDS

O *Netflow University of Queensland Network Intrusion Detection System* (NF-UQ-NIDS) é um *dataset* focado em detecção de intrusão que foi desenvolvido por pesquisadores da Universidade de Queensland na Austrália. Este conjuntos de dados (Sarhan et al., 2021) foi construído para preencher a lacuna da falta de padronização em relação aos *datasets* e a variação das informações representadas.

A Figura 16 demonstra uma comparação entre quatro importantes conjuntos de dados do contexto de detecção de intrusão.

Figura 16 – Diagrama de Venn mostrando a heterogeneidade de cada dataset.



Fonte: Adaptado de (Sarhan et al., 2020).

Observa-se que esses conjuntos possuem poucos atributos em comum. Diante disso, os pesquisadores propuseram a utilização de um conjunto estendido de atributos baseados no NetFlow v9. Para isso, eles realizaram uma combinação com os principais conjuntos de dados

empregados na pesquisa em detecção de intrusões para Internet das Coisas (IoT). Os conjuntos de dados UNSW-BN15, BoT-IoT, ToN-IoT e CSE-CIC-IDS2018 tiveram seus arquivos de captura de pacotes traduzidos para atributos extraídos através do NetFlow. Desse modo, foi gerado o conjunto de dados NF-UQ-NIDS, uma fusão dos demais *datasets*. Esta fusão propicia que haja diversos fluxos de diferentes configurações de rede, contendo características de ataques distintas. Isso pode ser usado para comparar os mesmos cenários de ataque conduzidos em redes de teste diferentes.

Algumas categorias de ataque foram agrupadas para uma categoria mais ampla. Ataques como DoS-Hulk, DoS-SlowHTTPTest, DoS-GoldenEye e DoS-Slowloris foram acopladas como DoS. Ataques DDoS, como DDoS-LOIC-UDP, DDoS-HOIC e DDoS-LOIC-HTTP foram acoplados em DDoS. Ataques de Força Bruta *(Brute Force*) como FTP-BruteForce, SSH-BruteForce, BruteForce-Web e BruteForce-XSS foram agrupados como *BruteForce*. Por fim, os ataques de injeção SQL foram agrupados em *SQL Injection*.

O conjunto de dados NF-UQ-NIDSv2 contêm um total de 75,987,976 registros, sendo 25,165,295 (33,12%) fluxos benignos e 50,822,681 (66,88%) são fluxos intrusivos. A Tabela 11 lista a distribuição das classes de ataques.

Tabela 11 – Distribuição das classes do conjunto de dados NF-UQ-NIDSv2.

Classe	Quantidade	Descrição
Benigno	25165295	Fluxos não maliciosos.
DDoS	21748351	DoS com várias fontes distribuídas diferentes.
Reconnaissance	2633778	Técnica para coletar informações sobre um host de rede e também é conhecida como sonda.
Injection	684897	Uma variedade de ataques que fornecem entradas não confiáveis que visam alterar o curso da execução, sendo as injeções de SQL e de código dois dos principais.
DoS	17875585	Sobrecarrega os recursos de um host com o objetivo de impedir o acesso ou a disponibilidade de seus dados.
Brute Force	123982	Técnica que visa obter credenciais de usuário e senha acessando uma lista de possibilidades predefinidas.
Password	1153323	Uma variedade de ataques destinados a recuperar senhas por força bruta ou sniffing.
XSS	2455020	Tipo de injeção em que um invasor usa aplicativos da Web para enviar scripts maliciosos aos usuários.
Infiltration	116361	Ataque interno que envia um arquivo malicioso para explorar um aplicativo e é seguido por um backdoor que verifica a rede em busca de outras vulnerabilidades.
Exploits	31551	São sequências de comandos que controlam o comportamento de um host por meio de uma vulnerabilidade.
Theft	2431	Um grupo de ataques que visa obter dados confidenciais, como roubo de dados e keylogging.
Scanning	3781419	O atacante procura por vulnerabilidades na rede, geralmente com o uso de ferramentas.
Fuzzers	22310	Ataque no qual o invasor envia grandes quantidades de dados aleatórios que causam a falha de um sistema e também visam descobrir vulnerabilidades de segurança.
Backdoor	18978	Técnica que visa atacar computadores de acesso remoto respondendo a aplicativos clientes.
Bot	143097	Ataque que permite que um invasor controle remotamente vários computadores sequestrados para realizar atividades maliciosas.
Generic	165560	Ataque que serve como uma etapa dentro de um processo de infecção por *Ransomware*.
Analysis	2299	Um grupo de ameaças que visam aplicativos da Web por meio de portas e scripts.
Shellcode	1427	Um malware que penetra em um código para controlar o host de uma vítima.
MITM	7723	Ataque que busca um invasor tenta interceptar o tráfego e a comunicação entre uma vítima e o host com o qual a vítima está tentando se comunicar.
Worms	164	Ataques que se replicam para outros computadores.
Ransomware	3425	Um ataque que criptografa os arquivos armazenados em um host e pede compensação em troca da técnica/chave de descriptografia.

Na Tabela 5.2.3 são listados os 43 atributos presentes neste *dataset*.

Tabela 12 – Conjunto de atributos presentes no conjunto de dados NF-UQ-NIDSv2.

Feature	**Descrição**
IPV4_SRC_ADDR	Endereço IPv4 de Origem
L4_SRC_PORT	Porta de Origem
IPV4_DST_ADDR	Endereço IPv4 de Destino
L4_DST_PORT	Porta de Destino
PROTOCOL	Identificador do Protocolo de Rede
L7_PROTO	Protocolo de aplicação
IN_BYTES	Número de bytes de Entrada
IN_PKTS	Número de Pacotes de Entrada
OUT_BYTES	Número de bytes de Saída
OUT_PKTS	Número de Pacotes de Saída
TCP_FLAGS	Cumulativo de todos os sinalizadores TCP
CLIENT_TCP_FLAGS	Cumulativo de todos os sinalizadores TCP Clientes
SERVER_TCP_FLAGS	Cumulativo de todos os sinalizadores TCP Servidor
FLOW_DURATION_MILLISECONDS	Duração do Fluxo em Milisegundos
DURATION_IN	Duração do fluxo do cliente para o servidor (Milisegundos)
DURATION_OUT	Duração do fluxo do cliente para o servidor (Milisegundos)
MIN_TTL	TTL Mínimo em Pacotes de Entrada do Fluxo
MAX_TTL	TTL Máximo em Pacotes de Entrada do Fluxo
LONGEST_FLOW_PKT	Maior Pacote em bytes do Fluxo
SHORTEST_FLOW_PKT	Menor Pacote em bytes do Fluxo
MIN_IP_PKT_LEN	Menor tamanho de Pacote Dentro do Fluxo
MAX_IP_PKT_LEN	Maior tamanho de Pacote Dentro do Fluxo
SRC_TO_DST_SECOND_BYTES	Quantidade de Bytes/seg (Origem e Destino)
DST_TO_SRC_SECOND_BYTES	Quantidade de Bytes/seg (Destino >Origem)
RETRANSMITTED_IN_BYTES	Quantidade de bytes de Fluxo TCP retransmitidos (Origem >Destino)
RETRANSMITTED_IN_PKTS	Quantidade de Pacotes de Fluxo TCP retransmitidos (Origem >Destino)

RETRANSMITTED_OUT_BYTES	Quantidade de bytes de Fluxo TCP retransmitidos (Destino >Origem)
RETRANSMITTED_OUT_PKTS	Quantidade de Pacotes de Fluxo TCP retransmitidos (Destino >Origem)
SRC_TO_DST_AVG_THROUGHPUT	Taxa de Transferência média de bytes/seg entre Origem >Destino
DST_TO_SRC_AVG_THROUGHPUT	Taxa de Transferência média de bytes/seg pelo Destino >Origem
NUM_PKTS_UP_TO_128_BYTES	Quantidade de Pacotes com Tamanho <= 128 bytes
NUM_PKTS_128_TO_256_BYTES	Quantidade de Pacotes com Tamanho >128 <= 256 bytes
NUM_PKTS_256_TO_512_BYTES	Quantidade de Pacotes com Tamanho >256 <= 512 bytes
NUM_PKTS_512_TO_1024_BYTES	Quantidade de Pacotes com Tamanho >512 <= 1024 bytes
NUM_PKTS_1024_TO_1514_BYTES	Quantidade de Pacotes com Tamanho >1024 <= 1514 bytes
TCP_WIN_MAX_IN	Maior Janela TCP do Fluxo (Origem >Destino)
TCP_WIN_MAX_OUT	Maior Janela TCP do Fluxo (Destino >Origem)
ICMP_TYPE	Tipo de Mensagem ICMP * 256 bits + Código ICMP
ICMP_IPV4_TYPE	Tipo de Mensagem ICMP
DNS_QUERY_ID	Identificação de Consulta DNS
DNS_QUERY_TYPE	Tipo de Consulta DNS
DNS_TTL_ANSWER	Tempo de Cache de Resposta DNS
FTP_COMMAND_RET_CODE	Código de Retorno de Comandos FTP

A próxima seção apresenta detalhes a respeito das configurações de experimentos realizados para avaliar a abordagem proposta.

5.3 DESCRIÇÃO DOS EXPERIMENTOS

Nesta seção, são apresentados os principais aspectos da metodologia experimental definida para a realização dos experimentos, os quais têm como objetivo validar e avaliar as abordagens e arquiteturas propostas neste trabalho.

5.3.1 Experimento 01 - Avaliação dos métodos binários propostos

Primeiramente, para avaliar as abordagens binárias propostas, DNNKNN e DNNET, foram conduzidos experimentos utilizando o conjunto de dados NSLKDD. A qual consiste em uma base de dados pública utilizada em muitos trabalhos (Diro; Chilamkurti, 2018c; Rathore; Park, 2018; Yahyaoui et al., 2021; Gopalakrishnan; Purusothaman, 2022) para avaliar métodos de detecção de intrusão.

As bases de dados de detecção de intrusão apresentadas na Seção 5.2 são conjuntos de dados extensos que podem ser utilizados para validação e avaliação de modelos de detecção. Esses dados podem ser empregados tanto para o treinamento quanto para os testes dos modelos. No entanto, idealmente, os modelos devem ser avaliados com amostras que não foram usadas para construir, treinar ou ajustar o modelo. Isso busca garantir uma avaliação imparcial da eficácia do modelo.

Os experimentos realizados neste trabalho empregaram a técnica *Hold-Out 70-30* para dividir o conjunto de dados e realizar a avaliação. Nessa estratégia, o conjunto de dados é particionado em 70% dos dados para treinamento dos modelos e 30% dos dados para o teste dos modelos gerados. Essa divisão é realizada com o intuito de evitar que dados utilizados para treinar o modelo sejam também usados para testá-lo. Além disso, é adotada uma estratégia estratificada para dividir estes dados, de modo, que o conjunto de treino e teste mantenham proporções iguais de instâncias por classe.

Conforme ilustrado na Figura 17, o conjunto de dados NSL-KDD passa inicialmente por uma etapa de pré-processamento, onde ocorre a normalização de dados e a remoção de valores ou símbolos inválidos. Em seguida, o conjunto de dados é submetido a um processo de binarização, onde o tráfego intrusivo recebe rótulo 1 e o não intrusivo recebe rótulo 0. Por fim, o conjunto binarizado é utilizado para avaliar as abordagens propostas.

Para realizar a avaliação, o método *Hold-Out 70-30* é executado 5 vezes para cada experimento. Desse modo, cada técnica é submetida 5 vezes aos processos de treino e teste, gerando 5 resultados. Estes 5 resultados permitem considerar uma visão mais ampla do desempenho das técnicas e verificar se elas conseguem manter um padrão de desempenho.

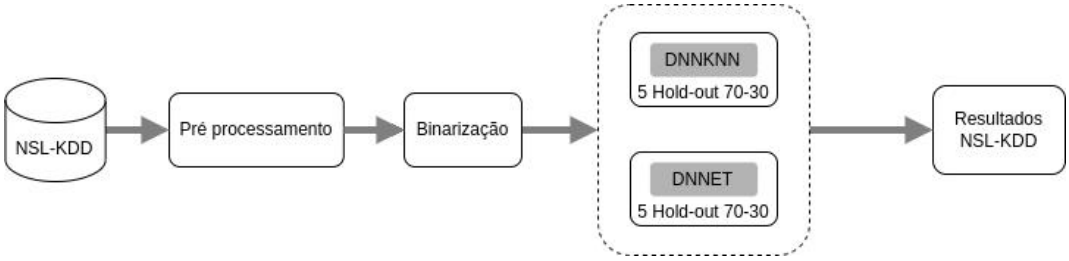


Figura 17 – Ilustração do processo experimental definido para avaliação das abordagens binárias.

Nestes experimentos, a DNN utilizada em ambos os métodos teve a mesma arquitetura apresentada no Capítulo 4. Para o treinamento, o *Adam* foi utilizado como algoritmo de

otimização e a *Sparse Categorical Crossentropy* como função de perda. Foi utilizado também o mecanismo de parada antecipada quando uma métrica monitorada passa mais de 20 iterações sem melhorar. Os valores para os demais parâmetros do treinamento do modelo foram os padrão da biblioteca *Keras*. Do mesmo modo, os parâmetros $taxa\_FP\_aceitavel$ e $taxa\_FN\_aceitavel$ foram definidos como $0,05$ em ambos os métodos.

Na abordagem DNNKNN, para o algoritmo KNN foram considerados os $k = 1$ vizinhos mais próximos e como métrica de similaridade a distância Euclidiana. No caso da ET do método DNNET, foi utilizada uma estrutura de 10 árvores de decisão e limite de profundidade de 10 níveis ($p = 10$). Os demais parâmetros foram utilizados com os valores padrão da biblioteca *Scikit Learn*.

Os resultados de detecção obtidos pelas abordagens propostas são comparados com os obtidos por abordagens do estado da arte em experimentos com o conjunto de dados NSL-KDD. Além disso, são analisados os tempos de treino e teste das abordagens propostas.

5.3.2 Experimento 02 - Avaliação da abordagem de detecção e identificação

Nesta seção, é apresentada a metodologia experimental definida para a validação e avaliação da abordagem de detecção e identificação proposta. A fim de avaliar a viabilidade e o desempenho, foram conduzidos experimentos e efetuadas comparações com técnicas clássicas de aprendizado de máquina e abordagens do estado da arte.

A técnica *Hold-Out 70-30* também é empregada para particionar o conjunto de dados e conduzir as avaliações nos experimentos realizados nesta etapa. Da mesma forma, visando obter uma avaliação mais robusta, o método *Hold-Out 70-30* é executado em 5 repetições para cada experimento. Assim, cada técnica é submetida a 5 iterações de treinamento e teste, gerando um total de 5 conjuntos de resultados.

A Figura 18 ilustra o processo empregado para a avaliação da abordagem proposta em comparação com abordagens clássicas do estado da arte. Os conjuntos de dados NSL-KDD, IoTID20 e NF-UQ-NIDSv2, apresentados na Seção 5.2, são utilizados para essa avaliação. A versão utilizada da NF-UQ-NIDSv2 neste estudo corresponde a 10% da base de dados original, reduzida de maneira estratificada para preservar as proporções originais de instâncias por classe.

Inicialmente, cada conjunto de dados passa por uma etapa de pré-processamento, que envolve a normalização dos dados e a remoção de valores ou símbolos inválidos. Após o pré-processamento, os conjuntos de dados são utilizados para a avaliação tanto das técnicas clássicas quanto da abordagem proposta.

Nestes experimentos, a abordagem proposta contou com o método DNNET e a abordagem *ensemble* de votação. O método DNNET teve um modelo neural com a mesma arquitetura apresentada no Capítulo 4. Para o treinamento, o *Adam* foi utilizado como algoritmo de otimização e a *Sparse Categorical Crossentropy* como função de perda. Foi utilizado o mecanismo de parada antecipada quando uma métrica monitorada passa mais de 20 iterações sem melhorar. Os valores para os demais parâmetros do treinamento do modelo foram os padrão da biblioteca

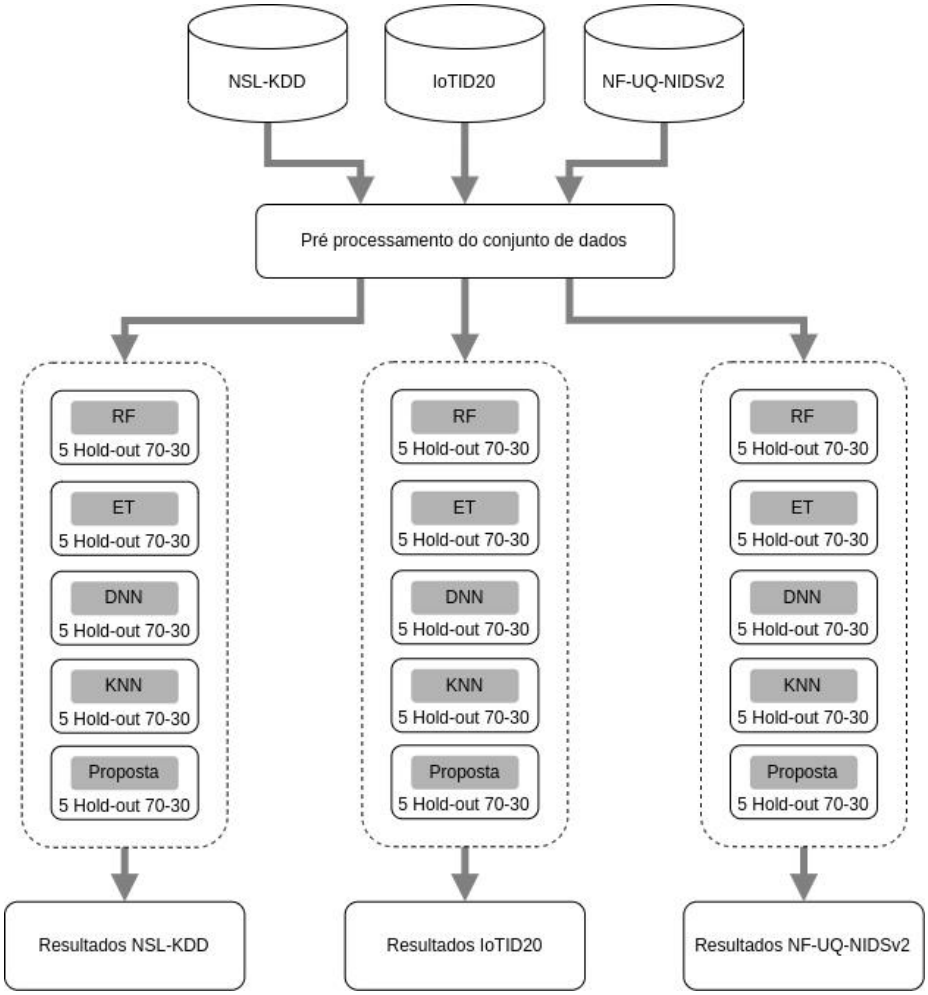


Figura 18 – Ilustração do processo experimental definido para avaliação da abordagem proposta.

Keras. Os parâmetros $taxa\_FP\_aceitavel$ e $taxa\_FN\_aceitavel$ foram definidos como $0,05$. A ET teve uma estrutura de 10 árvores de decisão e limite de profundidade de 10 níveis ($p = 10$). Os demais parâmetros foram utilizados com os valores padrão da biblioteca *Scikit Learn*. O método *ensemble* teve uma RF e uma ET com 100 árvores de decisão e limite de 100 níveis de profundidade para cada árvore ($p = 100$). Os demais valores padrão da biblioteca *Scikit Learn*. A arquitetura da DNN presente no método *ensemble* foi definida com duas camadas ocultas, cada uma contendo 150 neurônios com função de ativação *ReLU*. Os neurônios da camada de saída utilizam a função de ativação *softmax*. Para o treinamento, o *Adam* foi utilizado como algoritmo de otimização e foi utilizado o mecanismo de parada antecipada quando uma métrica monitorada passa mais de 20 iterações sem melhorar. A estratégia de combinação das classificações geradas por cada uma das técnicas individuais no método *ensemble* foi a combinação *soft*.

Conforme apresentado na Figura 18, as técnicas clássicas utilizadas na comparação foram: *Deep Neural Network* (DNN), *K-Nearest Neighbor* (KNN), *Extra Tree* (ET) e *Random Forest* (RF). A DNN padrão utilizada nos experimentos contou com duas camadas ocultas com

o mesmo número de neurônios que a quantidade de atributos presente na base de dados. Os neurônios das camadas ocultas operam com função de ativação tangente hiperbólica. A camada de saída da arquitetura possui função de ativação *softmax* em cada neurônio.

Também foi utilizado nos experimento a técnica *Random Forest*. Seu parâmetro principal é denotado por a, que representa o número de árvores na floresta. Nestes experimentos o valor de a foi definido como 100, correspondendo à configuração padrão da biblioteca. O parâmetro c, que indica o número de atributos candidatos selecionados aleatoriamente em cada nó, foi definido como $sqrt(N)$, onde N é o numero de atributos do conjunto de dados. A seguir, diversos parâmetros relacionados à estrutura das árvores de decisão que compõem a estrutura da *Random Forest* são descritos. O critério de Gini foi escolhido como função para medir a qualidade de uma divisão. Os critérios suportados são Gini para a impureza Gini e entropia para o ganho de informação. Foi escolhido o critério de Gini por ser o padrão da biblioteca. O parâmetro p, que indica a profundidade máxima permitida da árvore, foi definido como *None*. Neste caso, os nós são expandidos até que todas as folhas sejam puras ou até que todas as folhas contenham menos de m amostras. O valor de m foi definido como 2, assegurando que seja necessário um mínimo de duas amostras para efetuar a divisão de um nó interno. Além disso, o número mínimo de amostras necessárias para formar um nó folha (f) foi definido com 1, garantindo que um ponto de divisão em qualquer profundidade seja considerado somente se pelo menos uma amostra for alocada em ambos os ramos esquerdo e direito. Esses três parâmetros também foram definidos de acordo com a configuração padrão da biblioteca, com o objetivo de criar uma árvore completa sem restrições de tamanho.

Outra técnica empregada para comparação foi a *Extra Tree*. Ela é bastante semelhante em operação à RF, variando principalmente na maneira como as árvores são construídas dentro da floresta.Assim como na RF, um subconjunto aleatório de c atributos candidatos é utilizado. No entanto, neste caso, em vez de procurar os pontos de corte mais discriminativos, os pontos de corte são sorteados aleatoriamente para cada atributo candidato. Para a *Extra Tree* os parâmetros a e c também foram definidos como 100 e $sqrt(N)$, respectivamente. Do mesmo modo, utilizou-se, para as 100 árvores da ET, as mesmas configurações relacionadas as árvores de decisão. O critério utilizado foi o de Gini, p foi definido como *none*, m como 2 e f como 1.

Por fim, a última técnica utilizada na avaliação foi o algoritmo KNN. Essa técnica se fundamenta no cálculo da similaridade das instâncias com os exemplos existentes na base de dados. Este algoritmo considera os k= vizinhos mais próximos para, e o valor de $k = 1$ foi definido de acordo com os resultados obtidos em estudos anteriores (Souza et al., 2018). Para calcular a similaridade, a distância euclidiana é utilizada como métrica. Além disso, uma estrutura de árvore k-dimensional é utilizada para armazenar os exemplos que compõem a base de exemplos do algoritmo kNN. É uma estrutura de dados de particionamento de espaço para organizar pontos em um espaço k-dimensional e permite otimizar o custo computacional.

Cada técnica mencionada anteriormente é avaliada por meio de 5 execuções do método *Hold-out 70-30* para cada conjunto de dados. A média dos resultados dessas 5 execuções do *Hold-out* é empregada para apresentar os resultados. Além disso, os 5 conjuntos de resultados

são utilizados para realizar uma análise estatística, a fim de verificar a presença de diferença estatisticamente significativa entre os grupos.

5.3.3 Experimento 03 - Avaliação da abordagem de treinamento federado

O terceiro experimento consiste em uma avaliação sobre a estratégia de treino federado apresentada. O objetivo é avaliar se a abordagem federada pode ser aplicada para reduzir os custos de treinamento local e manter a privacidade dos dados locais sem comprometer o desempenho de detecção da abordagem em relação à abordagem centralizada. Para esta avaliação, também foi utilizada a técnica *Hold-Out 70-30*, sendo realizados os experimentos apresentados na Figura 19.

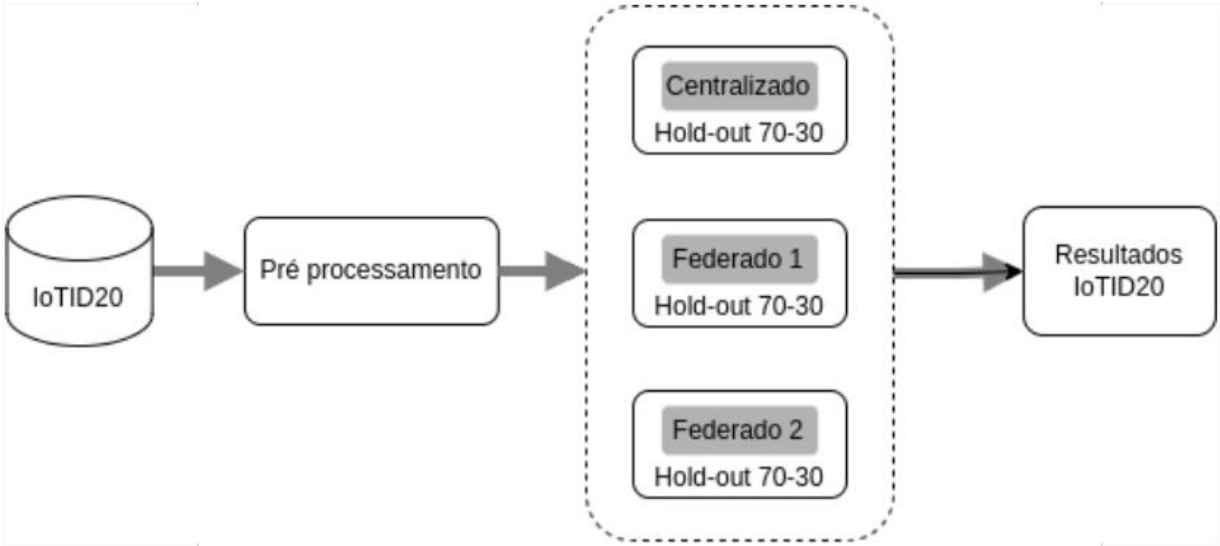


Figura 19 – Ilustração do processo experimental definido para avaliação da abordagem de treinamento federado.

A seguir são descritos cada um dos experimentos:

- **Centralizado:** treinamento centralizado do modelo neural.
- **Federado 1:** treinamento federado com dados de treinamento distribuídos de forma estratificada em 10 partes, uma para cada cliente, buscando manter as mesmas proporções de instâncias por classe em cada cliente.
- **Federado 2:** treinamento federado com dados de treinamento distribuídos em 10 partes, aleatoriamente entre os clientes.

Para esses experimentos, foram considerados 10 clientes. Consequentemente, existem 10 nós de neblina com modelos de detecção neural. Esses clientes utilizam seu conjunto específico de dados locais para treinar seu modelo local e enviam apenas os pesos sinápticos para agregação ao modelo global na nuvem. A mesma arquitetura de modelo neural é empregada para todos os clientes e também na abordagem centralizada. O modelo consiste na DNN

Multilayer Perceptron Feed-Forward com duas camadas ocultas proposta no método DNNET. Os parâmetros utilizados nesse experimento são os mesmos utilizados para a DNN do método DNNET do experimento binário.

5.4 MATERIAIS APLICADOS

Os experimentos realizados neste trabalho foram feitos através de simulação com conjunto de dados renomados de detecção de intrusão.

Para implementar a abordagem proposta e também executar os experimentos foi utilizada a linguagem Python [4]. Além disso, também foram utilizadas algumas bibliotecas Python.

A biblioteca Pandas foi utilizada no carregamento e manipulação das bases de dados. A biblioteca *NumPy*, que é utilizada para manipulação de *arrays* e matrizes, também foi utilizada para manipular o conjunto de dados.

Para a implementação de alguns métodos de aprendizado de máquina foi utilizada a biblioteca *Scikit-Learn*. Através dela também foi possível utilizar o método de *Hold-Out*, realizar os cálculos das métricas utilizadas na avaliação, entre outras tarefas.

A implementação dos modelos neurais foi realizada através da biblioteca *Keras*[5], que permite a manipulação e configuração de algumas características das redes neurais, como por exemplo, o número de camadas. Esta biblioteca se trata de uma interface de alto nível para a biblioteca *TensorFlow*[6], que é a responsável pela criação e execução do modelo neural. Além disso, para a realização do treinamento federado foi utilizada a biblioteca *TensorFlow Federated*[7].

Por fim, para a execução dos testes estatísticos realizados foi utilizada a linguagem *R*[8].

[4] https://www.python.org/
[5] https://keras.io/
[6] https://www.tensorflow.org/
[7] https://www.tensorflow.org/federated
[8] https://www.r-project.org/

6 RESULTADOS E DISCUSSÕES

Neste capítulo, são apresentados os resultados decorrentes dos experimentos conduzidos para avaliar as abordagens propostas neste trabalho. Inicialmente, são apresentados os resultados de experimentos binários para avaliação dos métodos propostos para o primeiro nível de detecção. Posteriormente, é apresentada uma extensa avaliação da abordagem completa de detecção e identificação de intrusão com os conjuntos de dados NSLKDD, IoTID20 e NF-UQ-NIDSv2. Além dos experimentos de simulação, também é apresentada uma análise a respeito da complexidade de tempo envolvida na solução proposta. Por fim, apresentamos uma discussão a respeito dos principais resultados obtidos na avaliação da abordagem proposta.

6.1 EXPERIMENTO 01 - AVALIAÇÃO BINÁRIA

Nesta seção, são apresentados os resultados obtidos por meio do experimento destinado a avaliar as abordagens propostas, DNNKNN e DNNET, em comparação com abordagens binárias do estado da arte.

A Tabela 13 apresenta a comparação dos resultados alcançados pelas abordagens binárias propostas e por trabalhos do estado da arte, no que se refere às métricas de detecção calculadas a partir dos experimentos com o conjunto de dados NSL-KDD. A partir desses resultados, é possível observar que as abordagens DNNKNN e DNNET obtiveram as melhores taxas de precisão e detecção, seguidas pelas abordagens propostas por Diro e Chilamkurti (2018c), Omar, Goyal e Varadarajan (2021) e Nasir et al. (2022).

Observa-se que ambas as abordagens propostas apresentaram taxas de *F1-Score* superiores às das demais abordagens, indicando que tanto a precisão quanto o *recall* apresentado pelas abordagens foram bons. O *F1-Score* apresentado pela DNNKNN foi de 99,51% e o da DNNET foi de 99,78%. A abordagem proposta por Nasir et al. (2022) apresentou resultados muito similares, alcançando um *F1-Score* de 99,27%. O método proposto por Omar, Goyal e Varadarajan (2021) obteve alta taxa de recall, no entanto, a taxa de precisão ficou abaixo dos 99%, o que prejudicou a métrica *F1-Score*. De modo geral, conforme apresentado na Tabela 13, as abordagens DNNKNN e DNNET foram capazes de obter resultados similares ou superiores às abordagens do estado da arte.

Alguns outros trabalhos exibiram desempenho de precisão inferior (Sadaf; Sultana, 2020; Sahar; Mishra; Kalam, 2021; Gopalakrishnan; Purusothaman, 2022) quando comparados àqueles que alcançaram taxas de precisão superiores a 99%. O trabalho de Niu et al. (2023) se destacou no estado da arte, alcançando uma taxa de precisão de 99,02%. No entanto, as taxas de precisão obtidas pelas abordagens propostas foram ainda maiores. A DNNKNN detectou corretamente 99,59% dos eventos, e a DNNET foi ainda mais longe, alcançando uma notável taxa de precisão de 99,88%.

Além disso, ambos os métodos alcançaram taxas de detecção de ataques superiores a 99%. A taxa obtida pela DNNKNN foi de 99,43% e a da DNNET foi de 99,67%. O resultado

Tabela 13 – Resultados de detecção binária de diferentes abordagens recentes com o conjunto de dados NSL-KDD.

Abordagens	**ACC**	**Precisão**	**Recall**	**TNR**	***F1-Score***	**MCC**
D'Angelo et al. (2015)	94,10	93,6	89,3	96,7	91,4	87,0
Raman et al. (2017)	97,14	-	-	-	-	-
Tang et al. (2018)	82,2	-	-	-	-	-
Li et al. (2018)	94,92	98,72	-	-	95,39	-
Pamukov et al. (2018)	95,88	73,54	93,37	96,16	80,66	73,51
Prabavathy et al. (2018)	97,36	-	97,7	-	-	-
Diro et al. (2018b)	99,20	-	99,27	-	-	-
Diro et al. (2018c)	99,20	-	99,27	-	-	-
Florencio et al. (2018)	97,14	98,6	-	-	96,9	-
Vinayakumar et al. (2019)	80,1	69,2	96,9	-	80,7	-
Pajouh et al. (2019)	84,86	-	-	-	-	-
Alrashdi et al. (2019)	98,2	-	97,1	-	-	-
Wang et al. (2020)	80,6	82,8	80,6	-	80,7	-
Sadaf et al. (2020)	95,4	94,8	97,2	-	96,0	-
Omar et al. (2021)	99,4	98,7	99,4	-	99,0	-
Sahar et al. (2021)	95,4	96,2	95,4	-	90,2	-
Yahyaoui et al. (2021)	99	-	-	-	-	-
Gopalakrishnan et al. (2022)	95,6	98,3	92,2	98,6	95,2	91,3
Nasir et al. (2022)	99,23	99,30	99,24	-	99,27	-
Bebortta et al. (2023)	93,47	93,89	92,48	-	93,71	-
Niu et al. (2023)	-	99,02	99,03	-	-	-
Jullian et al. (2023)	98,67	98,30	99,22	-	-	-
KNN	99,70	99,71	99,66	99,74	99,68	99,35
DNNKNN	**99,54**	**99,59**	**99,43**	**99,64**	**99,51**	**99,35**
DNNET	**99,79**	**99,88**	**99,67**	**99,90**	**99,78**	**99,34**

foi muito próximo do obtido pelo algoritmo KNN, que alacançou taxa de detecção de ataque de 99,66%. Ao analisar a métrica MCC, é possível notar uma similaridade entre os resultados das abordagens KNN, DNNKNN e DNNET. O KNN obteve 99,35%, a DNNKNN 99,35% e a DNNET 99,34%. Está métrica indica a qualidade da detecção binária, sendo 100% uma classificação totalmente correta. Nota-se que nos experimentos realizados, a abordagem DNNKNN e a DNNET foi capaz de alcançar uma qualidade de detecção binária similar ou superior à do KNN.

Além disso, o objetivo de reduzir o custo computacional também foi atingido, conforme evidenciado na Figura 20. O método DNNKNN proposto em Souza et al. (2020) teve como objetivo otimizar o algoritmo KNN em relação ao custo computacional na detecção de intrusões. Conforme pode ser observado, o tempo de detecção do KNN foi de 113,57 segundos e o da abordagem DNNKNN foi de 3,11 segundos, uma redução de aproximadamente 97%. É importante notar que o tempo relatado corresponde à duração necessária para classificar todo o conjunto de dados de teste, não apenas uma única instância. Apesar da redução alcançada, o método DNNET foi concebido para otimizar ainda mais o tempo de predição, ao mesmo tempo em que se esforça para manter a qualidade da detecção binária apresentada pelo DNNKNN.

O tempo de detecção apresentado pela DNNET foi de 1,96 segundos, uma redução de aproximadamente 36% em relação a abordagem DNNKNN. Essa diferença tende a se tornar maior em cenários onde um número maior do trafego necessita ser enviado para o segundo método de detecção, pois a ET é um método mais leve que o algoritmo KNN em relação ao tempo de detecção.

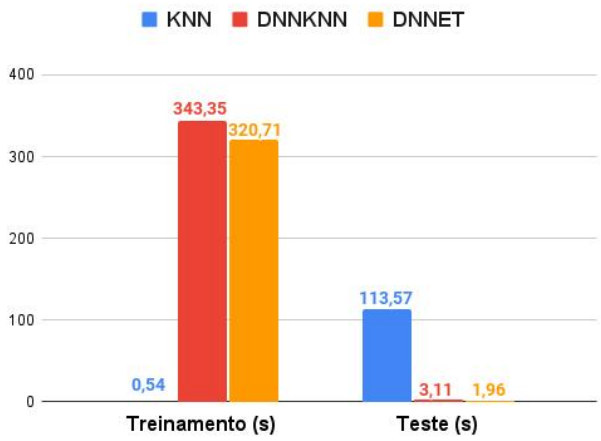


Figura 20 – Comparação de tempo entre as abordagens KNN, DNNKNN e DNNET em experimentos com o conjunto de dados NSLKDD.

Em relação ao tempo de treinamento, observa-se que as abordagens propostas apresentaram um tempo muito maior que o KNN. Isso ocorre pois o KNN não gera modelo, o custo está todo na etapa de detecção, o que é ruim onde busca-se uma detecção em tempo real. O tempo elevado em relação ao treinamento das abordagens propostas pode ser atenuado com a execução paralela do treinamento federado do modelo neural, estratégia que não foi considerada nesse experimento.

A seguir, são apresentados os resultados dos experimentos conduzidos para avaliar a abordagem completa de detecção e identificação de intrusões.

6.2 EXPERIMENTO 02 - AVALIAÇÃO DA ABORDAGEM DE DETECÇÃO E IDENTIFICAÇÃO

Nesta seção, são apresentados os resultados obtidos nos experimentos para avaliar a abordagem completa de detecção e identificação de intrusões. Estão contemplados nesta avaliação experimentos com as bases de dados NSLKDD, IoTID20 e NF-UQ-NIDSv2. A seguir, são apresentados os resultados do experimento com a base NSLKDD.

6.2.1 NSLKDD

Os resultados dos experimentos com o conjunto de dados NSL-KDD são apresentados nesta seção. Mesmo que esse conjunto de dados não contenha registros de tráfego da Internet

das Coisas (*Internet of Things* - IoT), ele foi escolhido devido à sua relevância na área de detecção de intrusões. Nesta seção a abordagem proposta DNNET-ensemble é comparada com abordagens clássicas de Aprendizado de Máquina, dentre elas estão a rede neural *Multi-Layer Perceptron* (MLP), *k-Nearest Neighbor* (kNN), *Extra Tree* (ET) e *Random Forest* (RF). Os resultados exibidos foram calculados de acordo com as métricas apresentadas na Seção 5.1.

A Tabela 14 exibe os resultados obtidos pelo método RF na classificação dos dados da base NSL-KDD. A RF conseguiu atingir taxas de detecção, precisão e F1-Score superiores a 99% na identificação do tráfego benigno, bem como de ataques DoS e Probe. Além disso, identificou aproximadamente 95% dos ataques R2L e somente 37.5% dos ataques U2R.

Tabela 14 – Resultados dos experimentos da abordagem RF com conjunto de dados NSL-KDD.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,80%	99,94%	99,87%
DoS		99,97%	99,98%	99,97%
Probe		99,86%	99,55%	99,71%
R2L		98,69%	95,64%	97,13%
U2R		92,29%	37,50%	52,36%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
8,2465	0,4651	99,86%	86,52%	99,86%

A ET é um método baseado em árvores de decisão, assim como a RF. Ambas demonstraram desempenho similar nos experimentos com o conjunto de dados NSL-KDD. A ET também foi capaz de alcançar taxas de detecção superiores a 99% na identificação de tráfego benigno e de ataques DoS e Probe, conforme evidenciado na Tabela 15. dicionalmente, conseguiu identificar 94% dos ataques R2L, um pouco menos que a RF. No entanto, ela superou a RF na identificação dos ataques U2R, ao identificar 45%, enquanto a RF identificou aproximadamente 37,5%.

Tabela 15 – Resultados dos experimentos da abordagem ET com conjunto de dados NSL-KDD.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,76%	99,92%	99,84%
DoS		99,97%	99,96%	99,97%
Probe		99,88%	99,49%	99,68%
R2L		97,84%	94,16%	95,96%
U2R		79,33%	45,00%	56,83%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
8,1259	0,5640	99,83%	87,71%	99,82%

A DNN avaliada com o conjunto de dados NSL-KDD também apresentou bons resultados na identificação de tráfego benigno e de ataques DoS e *Probe*, alcançando taxas de detecção, precisão e F1-Score superiores a 99%, conforme apresentado na Tabela 16. Além disso, essa técnica manteve a taxa de detecção de ataques U2R próxima a 42%, porém teve um desempenho inferior em relação às demais técnicas na detecção de ataques R2L, identificando aproximadamente 89,2% desses ataques.

Tabela 16 – Resultados dos experimentos da abordagem DNN com conjunto de dados NSL-KDD.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,59%	99,69%	99,64%
DoS		99,90%	99,87%	99,88%
Probe		99,14%	99,15%	99,15%
R2L		92,53%	89,26%	92,74%
U2R		64,26%	41,25%	49,68%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
421,6841	1,3721	99,60%	85,84%	99,59%

A Tabela 17 exibe os resultados obtidos pelo algoritmo KNN nos experimentos realizados com o conjunto de dados NSL-KDD. Semelhantemente às outras técnicas, o KNN também demonstrou taxas de detecção, precisão e F1-Score superiores a 99% na identificação de tráfego benigno e ataques DoS e Probe. Adicionalmente, registrou uma taxa de detecção de aproximadamente 94,4% na identificação de ataques R2L e de 52,1% em ataques U2R.

Tabela 17 – Resultados dos experimentos da abordagem KNN com conjunto de dados NSL-KDD.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,74%	99,71%	99,73%
DoS		99,89%	99,92%	99,90%
Probe		99,31%	99,34%	99,33%
R2L		93,93%	95,03%	94,46%
U2R		58,06%	47,50%	52,13%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
0,6069	182,2003	99,69%	88,30%	99,69%

A Tabela 18 ilustra os resultados obtidos pela abordagem proposta durante a avaliação com o conjunto de dados NSL-KDD. A abordagem demonstrou capacidade para alcançar taxas de detecção, precisão e F1-Score superiores a 99% na identificação de tráfego normal e ataques como DoS e *Probe*. Em relação aos ataques do tipo R2L, a abordagem proposta foi capaz de detectar aproximadamente 95,2% dos fluxos, com uma precisão de 95,7%. Por fim, também identificou corretamente cerca de 68,7% dos ataques do tipo U2R, com uma precisão aproximada de 69%. Portanto, podemos observar que em relação as demais abordagens avaliadas, a abordagem proposta não apenas manteve um bom desempenho na identificação do tráfego benigno e de ataques DoS e *Probe*, mas também foi capaz de melhorar o desempenho de identificação em relação a ataques como R2L e U2R.

Tabela 18 – Resultados dos experimentos da abordagem proposta com conjunto de dados NSL-KDD.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,75 %	99,89 %	99,82 %
DoS		99,98 %	99,95 %	99,96 %
Probe		99,87 %	99,19 %	99,53 %
R2L		95,70 %	95,23 %	95,46 %
U2R		69,04 %	68,75 %	68,43 %
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
535,7167	3,0114	99,80%	92,60%	99,80%

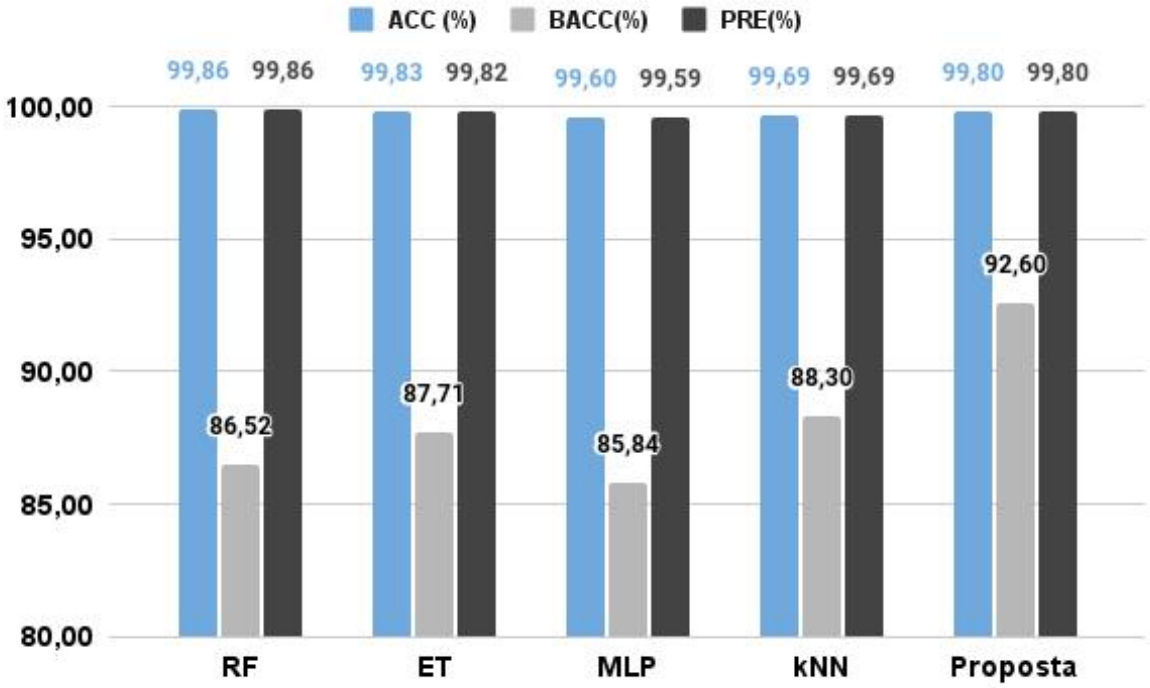
Os resultados gerais dos experimentos apresentados nas tabelas anteriores são analisados e discutidos a seguir. A Tabela 19 oferece uma nova perspectiva comparativa dos resultados obtidos nos experimentos, abrangendo métricas como tempo de treinamento (s), tempo de teste (s), acurácia, acurácia balanceada e precisão entre as abordagens propostas.

Tabela 19 – Resultados gerais obtidos pelas técnicas clássicas e a abordagem proposta em experimentos com o conjunto de dados NSL-KDD.

Abordagem	**ACC (%)**	**BACC (%)**	**PRE (%)**	**Treino (s)**	**Teste (s)**
RF	99,86%	86,52%	99,86%	8,2465	0,4651
ET	99,83%	87,71%	99,82%	8,1259	0,5640
DNN	99,60%	85,84%	99,59%	421,6841	1,3721
kNN	99,69%	88,30%	99,69%	0,6069	182,2003
Abordagem proposta	99,80%	92,60%	99,80%	435,7167	3,0114

Todas as técnicas avaliadas apresentaram taxas elevadas de acurácia e precisão, acima de 99%, conforme pode ser visualizado também no gráfico apresentado na Figura 21. Esses valores foram alcançados principalmente porque estas métricas são fortemente influenciadas pelo resultados das classes que possuem muitas instâncias, enquanto que são menos afetadas pelas classes com poucas instâncias. Todas as técnicas alcançaram elevadas taxas de detecção para classes de tráfego benigno e de ataques de negação de serviço, as quais possuem muitas instâncias no conjunto de dados NSL-KDD. Consequentemente, a acurácia e precisão global foram extremamente altas, mesmo existindo classes que as técnicas não conseguiram alcançar boas taxas de detecção.

Figura 21 – Gráfico dos resultados gerais obtidos pelas técnicas clássicas e a abordagem proposta em experimentos com o conjunto de dados NSL-KDD.

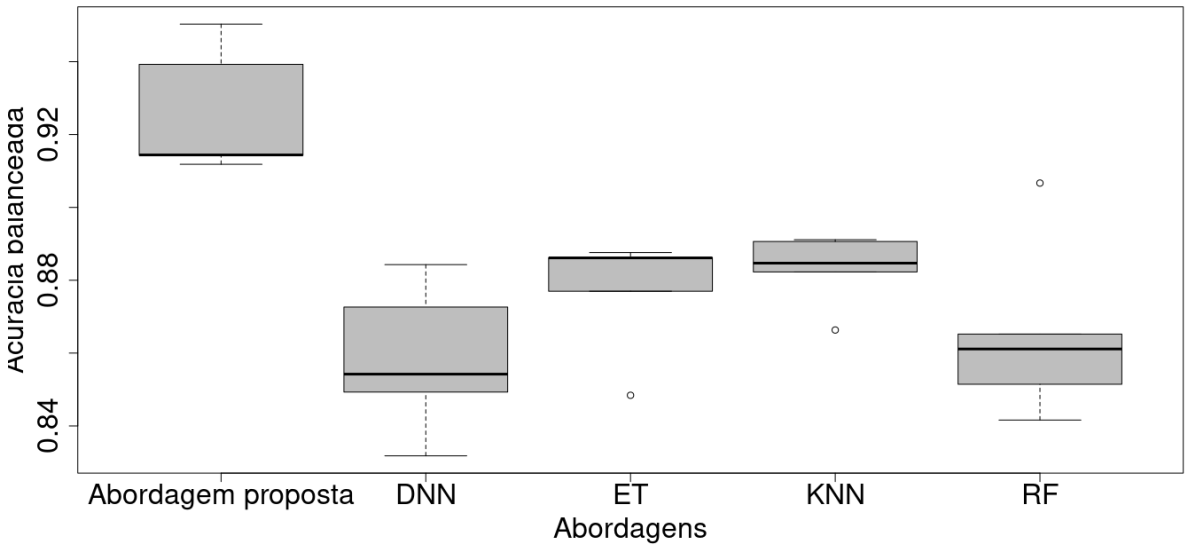


A métrica de Acurácia Balanceada (BACC) se revela mais relevante e realista neste contexto, uma vez que calcula a média das taxas de detecção das classes, tornando-se assim mais apropriada para a avaliação de resultados de detecção em conjuntos de dados desbalanceados. Conforme apresentado na Tabela 19, as técnicas RF e DNN apresentaram as menores taxas de acurácia balanceada, aproximadamente 86,5% e 85,8%, respectivamente. A técnica ET apresentou uma BACC superior, cerca de 87,7%, seguida pelo algoritmo KNN que obteve 88,3%.

Observa-se na Figura 21 que a abordagem proposta conseguiu superar o desempenho das demais técnicas em relação às taxas de Acurácia Balanceada, apresentando o maior valor, aproximadamente 92,6%. Esse desempenho superior é principalmente atribuído à sua taxa de detecção superior para ataques do tipo U2R, em comparação com as demais técnicas.

A Figura 22 apresenta um gráfico *boxplot* que representa a Acurácia Balanceada obtida pelas abordagens considerando as 5 execuções do experimento. Observa-se que a abordagem proposta apresentou os melhores resultados, já que o limite inferior apresentado por ela é mais elevado do que os limites superiores das demais técnicas.É importante ressaltar que a abordagem proposta não apresentou valores discrepantes (*outliers*). Foram aplicados testes estatísticos visando verificar se as técnicas realmente possuem taxa de acurácia balanceada com diferenças estatisticamente significativas entre si, bem como para confirmar que a abordagem proposta apresenta resultados superiores.

Figura 22 – Gráfico *boxplot* da acurácia balanceada obtida pelas técnicas clássicas e pela abordagem proposta em experimentos com o conjunto de dados NSL-KDD.



Na análise estatística realizada, foram levados em consideração os valores de Acurácia Balanceada gerados a partir das 5 execuções de cada técnica. Os detalhes dessa análise são apresentados na Tabela 20. Inicialmente, para determinar qual método estatístico deveria ser aplicado, foi necessário avaliar a normalidade dos dados utilizando o teste de *Shapiro-Wilk*.O resultado do teste de normalidade apresentou um valor de p de $0,5509$, que é superior a $0,05$. Isso sugere que não existem evidências significativas para rejeitar a hipótese de que os dados seguem uma distribuição normal. Devido a essa constatação, foi utilizado o método ANOVA para a comparação das médias entre as técnicas, com um intervalo de confiança de 95% ($\alpha = 0,05$). O resultado do teste ANOVA gerou um valor de p de $0,000119$, que é menor do que $0,05$. Portanto, a hipótese de que todas as abordagens são iguais foi rejeitada, levando à conclusão de que pelo menos uma abordagem difere significativamente das demais.

Tabela 20 – Testes estatísticos realizados com os resultados de Acurácia balanceada dos experimentos realizados no conjunto de dados NSL-KDD.

Shapiro Wilk (Normalidade)		**ANOVA (Significância)**	
W	0,96619	F	10,14
$p-value$	0,5509	$p-value$	0,000119

Tukey HSD (Pós-Teste)					
	RF	ET	DNN	KNN	Proposta
RF	1,0000000	0,8485857	0,9771392	0,5653500	**0,0003986**
ET	0,8485857	1,0000000	0,5237085	0,9857629	**0,0038781**
DNN	0,9771392	0,5237085	1,0000000	0,2624362	**0,0001111**
KNN	0,5653500	0,9857629	0,2624362	1,0000000	**0,0120206**
Proposta	**0,0003986**	**0,0038781**	**0,0001111**	**0,0120206**	1,0000000

Para identificar quais grupos específicos apresentavam diferenças estatisticamente significativas, aplicou-se o teste pós-hoc de Tukey HSD com um nível de confiança de 95% ($\alpha = 0,05$). Conforme ilustrado na Tabela 20, a aplicação desse teste revelou diferenças estatisticamente significativas nos resultados da Acurácia Balanceada da abordagem proposta em relação a todas as outras técnicas. Dado que a abordagem proposta obteve o maior valor de Acurácia Balanceada (92,60%), podemos concluir que ela é capaz de alcançar resultados estatisticamente superiores em termos de Acurácia Balanceada em comparação com as demais técnicas.

No processo de detecção de intrusão, o tempo de treinamento possui uma grande relevância. A fase de treinamento é onde os modelos de detecção são construídos para, posteriormente, serem utilizados na classificação. Importante notar que o processo de treinamento ocorre de forma independente do processo de análise da rede e identificação de eventos. Dada a natureza mutável das redes, torna-se necessário atualizar periodicamente o modelo, de forma a sempre manter o modelo atualizado. A dinamicidade e elasticidade presente em redes IoT contempla a inserção e remoção de diversos nós, ou dispositivos. Consequentemente, o comportamento da rede monitorada pode mudar e, desse modo, a abordagem precisa acompanhar tais variações. Torna-se, portanto, essencial determinar o momento em que o modelo de detecção precisa ser atualizado, o que implica em realizar um novo treinamento e gerar um novo modelo. Consequentemente, quanto menor for o tempo necessário para realizar o treinamento, menor será o tempo de espera por um novo modelo ou o uso de um modelo desatualizado.

Por outro lado, o tempo de teste denota o período requerido para a análise e identificação do tráfego, sendo igualmente relevante. Uma abordagem que envolva um processo de análise excessivamente custoso pode resultar em latência e, assim, tornar-se um ponto de gargalo na comunicação.

A Tabela 19 apresenta os tempos de treino e teste, em segundos, obtidos pelas diversas técnicas. É importante notar que o tempo relatado corresponde à duração necessária para classificar todo o conjunto de dados de teste, não apenas uma única instância. Conforme pode ser observado, a técnica KNN apresentou o menor tempo de treino, sendo a única a realizar esta etapa em menos de 1 segundo. Isso ocorre devido ao algoritmo KNN ser uma técnica que não gera modelo. Por outro lado, ele teve o maior tempo de teste, muito acima das demais técnicas, cerca de 182 segundos. Isso se deve principalmente às inúmeras comparações com os dados do conjunto de treinamento que são realizados durante o processo de análise e identificação.

As técnicas RF e ET são semelhantes, ambas baseadas em árvores de decisão, e registraram tempos de treinamento de aproximadamente 8 segundos. Suas estruturas e operações justificam esses curtos períodos de treinamento. Além do treinamento rápido, essas abordagens conseguiram obter os menores tempos de teste, ambos abaixo de 1 segundo.

A DNN registrou um tempo de treinamento consideravelmente longo, aproximadamente 422 segundos, em comparação com as demais abordagens clássicas. A abordagem proposta também apresentou um elevado tempo de treinamento, isso se deve ao fato de ser composta pela técnica DNN, a qual possui um complexo processo de treinamento. Existem alternati-

vas promissoras para contornar estes problemas de treinamento apresentados pelas abordagens DNN. As quais incluem a realização de treinamentos em dispositivos com maior capacidade computacional e a implementação de treinamentos distribuídos por meio dos dispositivos de computação em nevoeiro (*fog computing*), usando a técnica de aprendizado federado (*Federated Learning*). Na abordagem proposta inserimos o treinamento baseado em aprendizado federado para preservar a privacidade dos dados locais, no entanto, os experimentos de simulação não consideraram a distribuição das tarefas de computação de maneira paralela, por esse motivo o tempo de treinamento se manteve alto mesmo com o aprendizado federado. São necessários mais experimentos para avaliar o treinamento federado realizado de maneira paralela entre os dispositivos envolvidos.

O tempo de teste da DNN foi de 1,3 segundos, consideravelmente inferior ao tempo do KNN. Por outro lado, a abordagem proposta registrou um tempo de teste de aproximadamente 3 segundos, superior ao da DNN e das técnicas baseadas em árvores de decisão. No entanto, é fundamental ressaltar que a arquitetura da abordagem proposta incorpora um primeiro nível de detecção binária, e apenas os eventos detectados como intrusivos são enviados para o segundo nível onde são analisados por um método mais robusto. Isso significa que eventos benignos são analisados somente pelo primeiro nível. Dado que a maior parte do tráfego real costuma ser benigno, essa abordagem tende a se aproximar do tempo de detecção obtido pela DNN, uma vez que o primeiro nível da abordagem consiste em uma DNN simples. Entretanto, vale observar que essa característica não se manifesta plenamente no momento deste experimento, uma vez que o conjunto de dados utilizado possui uma quantidade significativa de exemplos intrusivos, representando quase metade das instâncias. Como resultado, ambas as etapas da abordagem são aplicadas a um grande número de instâncias, o que eleva o tempo de detecção, conforme observado neste caso.

A Tabela 21 apresenta os resultados obtidos por abordagens do estado da arte em experimentos com a base de dados NSL-KDD. O trabalho de Gp e D'Souza (2017) apresentou apenas resultados em termos de acurácia, uma métrica que pode ser influenciada pelo desequilíbrio de classes presente neste conjunto de dados e, portanto, não é a métrica ideal para comparação. Os demais trabalhos apresentaram índices de *recall*, indicando o quanto foi efetivamente detectado para cada classe. Como pode ser observado, a abordagem proposta obteve as maiores taxas de detecção para fluxos benignos e ataques DoS. Além disso, obteve taxas de detecção superiores a 99% na identificação de ataques Probe. Em relação aos ataques R2L, a abordagem proposta alcançou aproximadamente 95,2% de detecção, a segunda maior taxa, atrás da abordagem de Blanco et al. (2018) que obteve 99,18%. No entanto, em Blanco et al. (2018), não foi possível detectar ataques U2R.

Tabela 21 – Resultados obtidos por trabalhos do estado da arte em experimentos com o conjunto de dados NSL-KDD.

Abordagens	**Métricas**	**Benigno**	**DoS**	**Pro**	**R2L**	**U2R**
Prabavathy et al., (2018)	Recall	98,63	84,20	96,61	71,87	53,81
Almiani et al. (2020)	Recall	-	98,27	97,35	64,93	77,25
Diro e Chilamkurti (2018c)	Recall	97,43	99,50	99,00	91,00	-
Vinayakumar et al., (2017)	Recall	99,60	78,10	84,90	44,70	52,90
Hagos et al. (2017)	Recall	98,00	97,70	90,40	28,00	50,00
Gp e D'Souza (2017)	ACC	95,50	91,60	93,80	75,30	45,30
Ieracitano et al. (2020)	Recall	96,19	98,18	94,03	39,78	-
Blanco et al. (2018)	Recall	95,45	97,96	98,89	99,18	-
Moussa e Alazzawi (2020)	Recall	19,6	96,4	99,8	96,6	95,5
Onah et al. (2021)	Recall	97,5	96,9	93,4	77,1	73,5
Ponnusamy e Sharma (2021)	Recall	98,0	71,0	60,0	0,0	0,0
Diwan et al. (2021)	Recall	-	71,0	60,0	0,0	0,0
Zhao et al. (2022)	Recall	97,34	99,67	99,99	6,12	4,35
Liang et al. (2022)	Recall	99,45	99,38	99,38	86,33	67,23
Hou et al. (2022)	Recall	81,79	90,53	62,83	53,34	23,50
Kale et al. (2022)	Recall	-	99,33	99,91	94,63	62,16
Khan e Mailewa (2023)	Recall	92,0	73,0	76,0	24,0	48,0
Kasongo (2023)	Recall	99,27	99,90	99,40	92,13	0,0
Vishwakarma e Kesswani (2023)	Recall	94,65	93,05	87,69	58,40	40,29
Wang, Xu e Liu (2023)	Recall	96,19	96,62	95,01	79,84	87,05
Liu, Zhang e Wang (2023)	Recall	96,98	76,23	81,60	0,0	12,16
Proposta	**Recall**	**99,89**	**99,95**	**99,19**	**95,23**	**68,75**

A categoria de ataque U2R possui poucos fluxos no conjunto de dados. Conforme pode ser visto na Figura 23, as abordagens do estado da arte e métodos clássicos de aprendizado de máquina apresentaram dificuldades em detectar esses ataques (Gp; D'Souza, 2017; Diwan et al., 2021; Ponnusamy; Sharma, 2021; Zhao et al., 2022; Hou et al., 2022). A abordagem proposta por Du et al. (2020) mostrou boas taxas de identificação para tráfego benigno e ataques DoS, *probe* e R2L, mas detectou menos de 40% dos ataques U2R. O trabalho de Kasongo (2023) teve um bom desempenho na maioria das classes, mas não foi capaz de detectar ataques U2R. A abordagem proposta atingiu aproximadamente 69% de detecção de ataques U2R, sendo superada apenas por três abordagens. O trabalho de Wang, Xu e Liu (2023) obteve uma das melhores taxas de identificação de ataques U2R, aproximadamente 87%. Esse bom desempenho, porém, não se manteve para as demais classes, chegando a apresentar baixo índice de identificação de tráfego benigno. A abordagem de Almiani et al. (2020) alcançou 77,2% de detecção de tráfego U2R. No entanto, esta abordagem não apresentou taxa de detecção para fluxos benignos. Uma taxa baixa para esta categoria indica que a abordagem gera muitos falsos positivos, este é um

grande problema nas abordagens de detecção de intrusão. Isso é justamente oque ocorre em Moussa e Alazzawi (2020), onde a abordagem detectou 95% do tráfego U2R, mas menos de 20% do tráfego benigno foi identificado corretamente, gerando assim uma alta taxa de falsos positivos. Além disso, o trabalho de Almiani et al. (2020) apresentou dificuldades de detecção na categoria R2L, obtendo apenas 65% de detecção, enquanto a abordagem proposta obteve 95%, uma diferença de 30%.

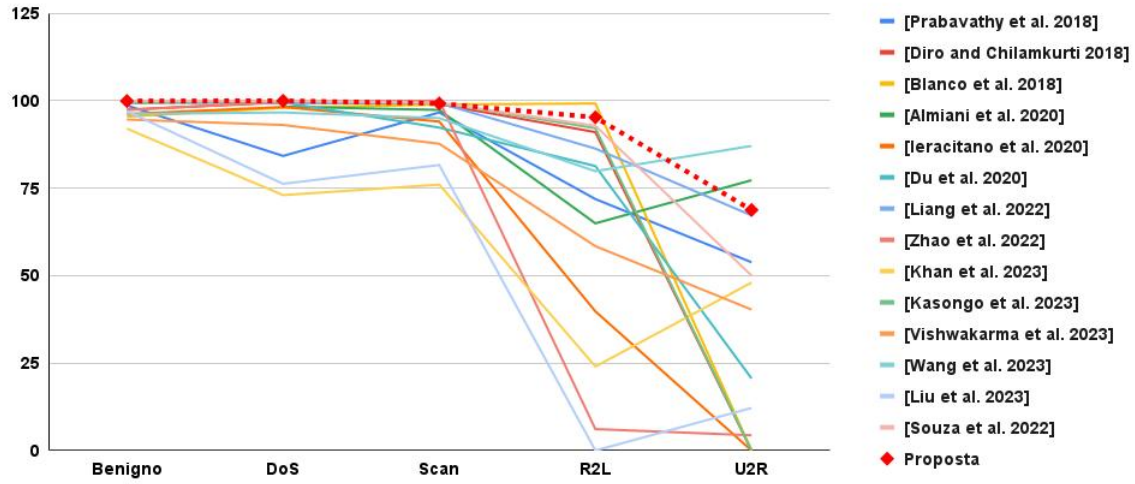


Figura 23 – Comparação dos experimentos com o conjunto de dados NSLKDD em relação a abordagens do estado da arte.

Conforme apresentado no Capítulo 3, as abordagens do estado da arte apresentam dificuldades relacionadas à identificação de tráfego benigno (Blanco et al., 2018; Ieracitano et al., 2020; Zhao et al., 2022; Vishwakarma; Kesswani, 2023). A abordagem de Blanco et al. (2018) apresentou 95,4%, e a de Vishwakarma e Kesswani (2023) obteve 94,6% na identificação de fluxos benignos. Isto pode ser considerado um ponto fraco das abordagens, indicando que uma grande quantidade de tráfego benigno pode ser erroneamente detectada como um ataque. Os falsos positivos são um grande problema com técnicas baseadas em anomalias e podem prejudicar a rede. A abordagem proposta obteve aproximadamente 99,89% de detecção de fluxos normais, sendo a abordagem que obteve a maior métrica. Portanto, realiza a detecção de ataques gerando poucos falsos positivos.

A seguir são apresentados os resultados obtidos nos experimentos realizados com o conjunto de dados IoTID20.

6.2.2 IoTID20

Os resultados dos experimentos com conjunto de dados IoTID20 são apresentados nesta seção. Este conjunto de dados foi construído com o objetivo de representar tanto o tráfego IoT benigno quanto o intrusivo. A Tabela 22 apresenta os resultados obtidos pelo método RF

no experimento com este conjunto de dados. A RF apresentou taxas de detecção, precisão e *F1-Score* superiores a 99% na identificação do tráfego DoS e *Mirai*. A taxa de acertos na identificação do tráfego *Scan* foi de 97,5% com uma precisão de 98,5%. A menor taxa de acertos foi para a identificação do tráfego MITM, ficando abaixo de 95% de *recall*.

Tabela 22 – Resultados dos experimentos da abordagem RF com conjunto de dados IoTID20.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,76%	98,55%	99,15%
DoS		99,99%	99,91%	99,95%
Mirai		99,19%	99,66%	99,42%
Scan		98,53%	97,52%	98,02%
MITM		96,59%	94,89%	95,74%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
149,9009	3,8364,	99,08%	98,10%	99,08%

No que diz respeito ao tráfego benigno, observa-se que a taxa de detecção (*recall*), ficou próxima de 99%, porém inferior. Essa métrica em relação ao tráfego benigno é significativa, pois permite visualizar que mais de 1% do tráfego que é benigno não foi corretamente identificado como benigno, resultando em falsos positivos. Os falsos positivos representam um desafio nas abordagens de detecção baseadas em anomalia, uma vez que nem todo comportamento anômalo constitui uma intrusão. Abordagens com altas taxas de falsos positivos podem prejudicar o funcionamento da rede, pois parte do tráfego benigno é bloqueada erroneamente pelo sistema de detecção.

Na Tabela 23, são apresentados os resultados obtidos pelo método ET nos experimentos com o conjunto de dados IoTID20. A ET obteve resultados semelhantes à RF na identificação do tráfego DoS e *Mirai*. No entanto, para os tráfegos *Scan* e MITM, os resultados foram inferiores aos da RF, especialmente no caso do tráfego MITM, o qual foi identificado em apenas 92,1% dos casos. Em relação à identificação do tráfego benigno, a ET também apresentou uma taxa de detecção inferior a 99%, resultando em mais de 1% de falsos positivos.

Tabela 23 – Resultados dos experimentos da abordagem ET com conjunto de dados IoTID20.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,43%	98,23%	98,82%
DoS		99,96%	99,89%	99,92%
Mirai		98,83%	99,27%	99,05%
Scan		96,89%	96,09%	96,49%
MITM		94,01%	92,14%	93,06%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
107,3707	4,8039	98,47%	97,12%	98,47%

A abordagem DNN apresentou o maior valor de *recall* na identificação do tráfego benigno entre as técnicas clássicas avaliadas, atingindo 98,7%. No entanto, ainda ficou ligeiramente abaixo dos 99%, como pode ser observado na Tabela 24. Além disso, a DNN não

manteve o mesmo desempenho das técnicas anteriores na identificação do tráfego *Mirai*, registrando uma taxa de 97,9%. Em relação ao tráfego *Scan* e MITM, as taxas permaneceram em 97% e 90,9% de *recall*, respectivamente.

Tabela 24 – Resultados dos experimentos da abordagem DNN com conjunto de dados IoTID20.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,47%	98,72%	99,09%
DoS		99,95%	99,85%	99,90%
Mirai		98,88%	97,93%	98,41%
Scan		90,32%	97,42%	93,73%
MITM		95,16%	90,95%	92,99%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
2035,4312	10,7406	97,71%	96,97%	97,78%

Na Tabela 25, são apresentados os resultados obtidos pela abordagem KNN nos experimentos com o conjunto de dados IoTID20. Observa-se que a técnica KNN apresentou um desempenho bastante semelhante com ao da técnica RF. As taxas de *recall*, precisão e *F1-Score* ficaram acima de 99% na identificação dos tráfegos DoS e *Mirai*. A técnica foi capaz de identificar corretamente 97,4% do tráfegos do tipo *Scan* e aproximadamente 95% do tráfego MITM. Assim como as demais técnicas clássicas, a taxa de *recall* para o tráfego benigno ficou abaixo dos 99%.

Tabela 25 – Resultados dos experimentos da abordagem KNN com conjunto de dados IoTID20.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,04%	98,43%	98,73%
DoS		99,93%	99,83%	99,88%
Mirai		99,30%	99,39%	99,35%
Scan		97,28%	97,41%	97,35%
MITM ARP Spoofing		95,56%	95,09%	95,32%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
12,8856	1567,6263	98,89%	98,03%	98,89%

Finalmente, na Tabela 26, são apresentados os resultados obtidos pela abordagem proposta nos experimentos com o conjunto de dados IoTID20. Destaca-se, em primeiro lugar, que a abordagem proposta foi capaz de obter taxas de *recall*, precisão e *F1-Score* superiores a 99% na identificação de tráfego benigno e de ataques como DoS, *Mirai* e *Scan*. Além disso, a abordagem proposta apresentou uma melhora em especial na taxa de acertos de identificação da classe de tráfego benigno, obtendo uma taxa de *recall* de aproximadamente 99,4%, sendo a única das abordagens a superar os 99% de acertos. Desse modo, podemos concluir que a abordagem proposta gerou menos falsos positivos que as demais técnicas. Em relação a classe de ataques MITM, a abordagem proposta também foi capaz de obter a maior taxa de detecção entre as técnicas avaliadas, alcançando taxa de *recall* de aproximadamente 98,4%.

Tabela 26 – Resultados dos experimentos da abordagem proposta com conjunto de dados IoTID20.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,64%	99,39%	99,51%
DoS		99,99%	99,91%	99,95%
Mirai		99,82%	99,76%	99,79%
Scan		99,53%	99,14%	99,33%
MITM		96,53%	98,46%	97,49%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
3373,5448	14,9077	99,60%	99,33%	99,60%

Na Tabela 27, é apresentada uma nova visualização comparativa dos dados gerais dos experimentos realizados com o conjunto de dados IoTID20. Observa-se que a abordagem proposta realmente teve o melhor desempenho de classificação em relação as técnicas clássicas avaliadas. As maiores taxas de acurácia, acurácia balanceada e precisão foram alcançadas pela abordagem proposta, todas acima de 99%. O fato de apresentar precisão de 99,6% permite concluir que a abordagem opera com uma taxa de falsos positivos inferior as demais técnicas, as quais apresentaram maior dificuldade na identificação do tráfego benigno. Destaca-se também que o bom desempenho de detecção apresentado em cada uma das classes reflete na métrica acurácia balanceada, a qual alcançou 99,33%, sendo a mais alta entre as técnicas avaliadas.

Tabela 27 – Resultados gerais obtidos pelas técnicas clássicas e a abordagem proposta em experimentos com o conjunto de dados IoTID20.

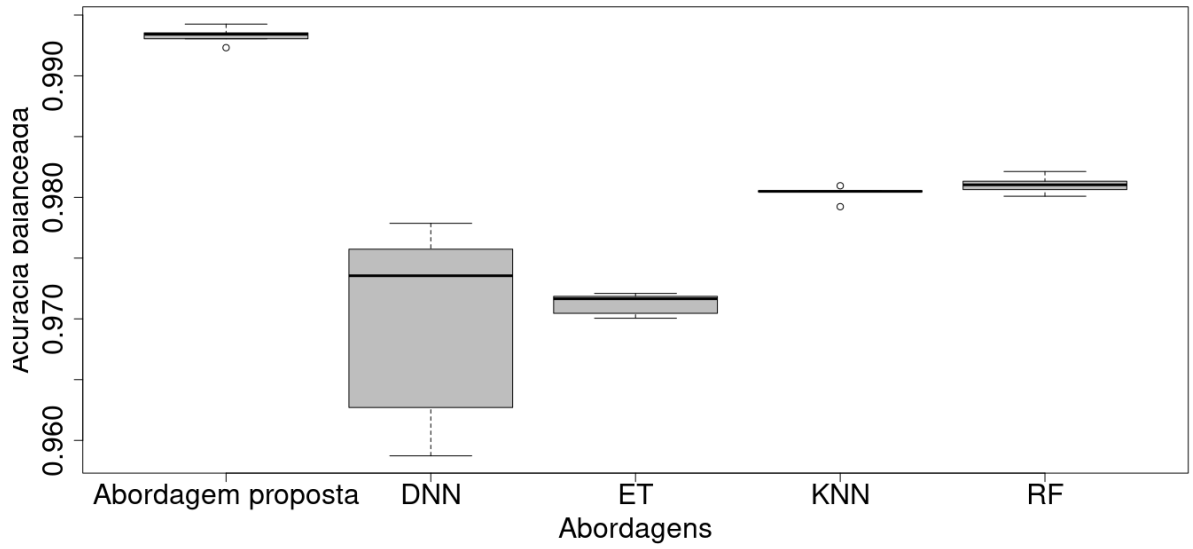
Abordagem	**ACC (%)**	**BACC (%)**	**PRE (%)**	**Treino (s)**	**Teste (s)**
RF	99,08%	98,10%	99,08%	149,9009	3,8364
ET	98,47%	97,12%	98,47%	107,3707	4,8039
DNN	97,71%	96,97%	97,78%	2035,4312	10,7406
kNN	98,89%	98,03%	98,89%	12,8856	1567,6263
Abordagem proposta	99,60%	99,33%	99,60%	3373,5448	14,9077

Os tempos de treinamento e teste foram maiores em relação ao conjunto de dados NSLKDD, devido à quantidade maior de instâncias existentes no conjunto IoTID20. Novamente, a técnica KNN apresentou o menor tempo de treino e o maior tempo de teste, cerca de 12 e 1567 segundos, respectivamente. Conforme mencionado anteriormente, essa é uma característica da técnica, pois realiza inúmeras comparações com a base de exemplos no momento da classificação para encontrar os vizinhos mais próximos. As técnicas RF e ET novamente demonstraram um bom desempenho de tempo tanto no treinamento quanto no teste. É importante destacar que a ET treinou mais rapidamente devido à introdução de um grau adicional de aleatoriedade na construção das árvores em comparação com a RF. Por outro lado, essa aleatoriedade a mais inserida na construção das árvores pode gerar árvores mais profundas, o que acarretou em um maior tempo de teste para a ET em relação a RF.

A DNN e a abordagem proposta apresentaram os maiores tempos de treinamento, principalmente devido ao custo associado ao treinamento do modelo neural. É importante ressaltar que nestes experimentos o treinamento foi realizado sem considerar a característica de execução paralela do treinamento federado, portanto, na prática este tempo tende a ser reduzido. Além disso, em relação ao tempo de teste, observa-se que a DNN precisou de $10,7$ segundos e a abordagem proposta $14,9$ segundos para classificar o conjunto de teste. No entanto, em um ambiente real o tempo da abordagem proposta tende a reduzir devido a grande parte do tráfego ser analisado somente pela primeira etapa binária e ser liberado, uma vez que é considerado benigno.

Na Figura 24 é apresentado o gráfico *boxplot* dos resultados de acurácia balanceada obtidos nos experimentos com o conjunto de dados IoTID20. Observa-se que a técnica DNN foi a abordagem que apresentou a maior diferença interquartil, enquanto as demais técnicas conseguiram obter resultados com menor variação. Além disso, a abordagem proposta obteve resultados melhores do que as demais, uma vez que o limite inferior foi maior do que os limites superiores obtidos pelas outras técnicas. Para verificar se a abordagem proposta realmente possui resultados superiores e diferenças estatisticamente significativas em relação às demais, foram aplicados testes estatísticos nos resultados dos experimentos.

Figura 24 – Gráfico *boxplot* da acurácia balanceada obtida pelas técnicas clássicas e pela abordagem proposta em experimentos com o conjunto de dados IoTID20.



Foram considerados os valores de acurácia balanceada gerados a partir das 5 execuções de cada técnica. As informações referentes a esta análise são apresentadas na Tabela 28. Primeiramente, foi utilizado o teste de normalidade de *Shapiro-Wilk* para verificar a normalidade dos dados. O teste de normalidade apresentou $p - value$ de $0,097$, que é maior que $0,05$. Isso indica que não há evidências significativas para rejeitar a hipótese de que os dados seguem uma

distribuição normal. Devido a isso, para a comparação das médias entre as técnicas foi utilizado o método ANOVA considerando um intervalo de confiança de 95% ($\alpha = 0,05$). Isso gerou como resultado um $p-value$ de $3.25e-08$, que é menor que $0,05$. Portanto, a hipótese de que todas as abordagens são iguais é rejeitada, concluindo-se que pelo menos uma abordagem difere significativamente das demais.

Tabela 28 – Testes estatísticos realizados com os resultados de acurácia balanceada dos experimentos realizados no conjunto de dados IoTID20.

Shapiro Wilk (Normalidade)		**ANOVA (Significância)**	
W	0,93206	F	30,15
$p-value$	0,097	$p-value$	3.25e-08

Tukey HSD (Pós-Teste)					
	RF	ET	DNN	KNN	Proposta
RF	1,0000000	**0,0052199**	**0,0012921**	0,9982936	**0,0005387**
ET	**0,0052199**	1,0000000	0,9704425	**0,0099979**	**0,0000001**
DNN	**0,0012921**	0,9704425	1,0000000	**0,0024962**	**0,0000000**
KNN	0,9982936	**0,0099979**	**0,0024962**	1,0000000	**0,0002807**
Proposta	**0,0005387**	**0,0000001**	**0,0000000**	**0,0002807**	1,0000000

Foi aplicado o pós-teste de Tukey HSD com um intervalo de confiança de 95% ($\alpha = 0,05$) para verificar quais amostras específicas apresentavam diferenças estatisticamente significativas. Conforme pode ser observado na Tabela 28, foram observadas diferenças estatisticamente significativas entre os resultados de acurácia balanceada da RF (98,1%) em relação à ET (97,7%) e à DNN (96,9%), com a RF apresentando um melhor desempenho. O algoritmo KNN (98%) também apresentou resultados estatisticamente superiores em relação à ET e à DNN. Por fim, é importante destacar que a abordagem proposta presentou diferenças estatisticamente significativas em relação a todas as demais técnicas. Uma vez que a abordagem proposta alcançou a maior acurácia balanceada (99,33%), conclui-se que ela é capaz de obter resultados de acurácia balanceada estatisticamente superiores às demais técnicas clássicas avaliadas.

A Tabela 29 apresenta os resultados obtidos por abordagens do estado da arte em experimentos com o conjunto de dados IoTID20. Observa-se que alguns trabalhos apresentaram dificuldades na detecção de certos tipos de ataques. Por exemplo, Qaddoura et al. (2021a) teve dificuldades na identificação de tráfego DoS e Mirai. Sarwar et al. (2022) obteve um *recall* de 50% na identificação de ataques *Scan* e apenas 13% para ataques MITM. Os autores Qaddoura et al. (2021b) identificaram apenas 55% dos ataques DoS e 74% dos ataques MITM.

Tabela 29 – Resultados obtidos por trabalhos do estado da arte em experimentos com a base de dados IoTID20.

Abordagem	**Métrica**	**Benigno**	**DoS**	***Mirai***	***Scan***	**MITM**
Qaddoura et al. (2021a)	Recall	69,82	80,78	74,22	91,86	99,43
Qaddoura et al. (2021b)	Recall	99,85	55,58	83,34	85,40	74,76
Dat-Thinh, Xuan-Ninh e Kim-Hung (2022)	ACC	97,83	90,99	99,98	91,54	92,60
	Precisão	43,80	99,99	99,68	96,38	97,85
	Recall	31,74	99,97	99,23	98,96	97,62
Albulayhi et al. (2022)	Precisão	100,0	100,0	99,7	99,7	98,8
	Recall	100,0	99,9	99,9	99,0	97,7
	F-Score	100,0	99,9	99,8	99,3	98,2
Sarwar et al. (2022)	Precisão	92,0	100,0	86,0	50,0	33,0
	Recall	81,0	100,0	92,0	50,0	13,0
	F-Score	86,0	100,0	89,0	50,0	18,7
Ayubkhan et al. (2023)	Recall	-	97,57	99,93	95,04	94,45
Proposta	Precisão	99,64	99,99	99,82	99,53	96,53
	Recall	99,39	99,91	99,76	99,14	98,46
	F-Score	99,51	99,95	99,79	99,33	97,49

A abordagem proposta por Dat-Thinh, Xuan-Ninh e Kim-Hung (2022) apresentou bons resultados de detecção para todos os tipos de ataques da base de dados, mantendo o *recall* para estas classes acima de 96%. No entanto, em relação ao tráfego benigno, observa-se que ela foi capaz de identificar apenas 31,7%, uma taxa muito baixa. O trabalho proposto por Ayubkhan et al. (2023) não avaliou a abordagem em relação à taxa de identificação de tráfego benigno. Além disso, outras abordagens também apresentaram dificuldades na identificação do tráfego normal, obtendo taxas de *recall* inferiores a 90% (Qaddoura et al., 2021a; Sarwar et al., 2022). Isso indica que a abordagem apresentou problemas de falsos positivos, onde o tráfego normal é erroneamente detectado como intrusivo.

Por outro lado, conforme pode ser observado na Figura 25, a abordagem proposta foi capaz de identificar corretamente aproximadamente 99,4% do tráfego benigno existente, gerando uma baixa taxa de falso positivo. Isto se deve principalmente à arquitetura proposta, que permite empregar um método robusto no segundo nível de detecção e corrigir eventuais falsos positivos do primeiro nível de detecção. A abordagem proposta alcançou taxas de *recall*, precisão e *F-Score* superiores a 96% para todos os tipos de ataques, inclusive para o tráfego benigno. As taxas foram superiores a 99% para o tráfego benigno e para ataques DoS, *Mirai*, *Scan* e MITM. As menores taxas foram na identificação de ataques MITM, mas ainda assim, superiores às obtidas pelas abordagens do estado da arte.

Ressalta-se também que o bom desempenho de detecção apresentado em cada uma das classes reflete a métrica de acurácia balanceada, que atingiu 99,33%, a maior entre as técnicas avaliadas, conforme apresentado anteriormente na Tabela 27. O fato de possuir um BACC de

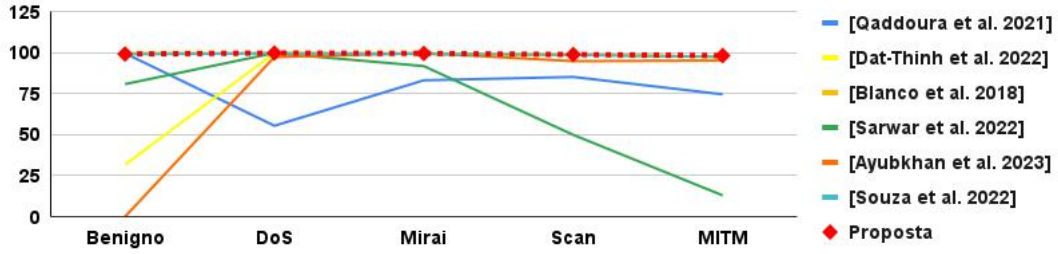


Figura 25 – Comparação dos experimentos com o conjunto de dados IoTID20 em relação a abordagens do estado da arte.

99,3% permite concluir que a abordagem opera com uma taxa de falsos positivos inferior às outras técnicas, que apresentaram maior dificuldade na identificação de tráfego benigno.

6.2.3 NF-UQ-NIDS

Os resultados dos experimentos com o conjunto de dados NF-UQ-NIDSv2 são apresentados nesta seção. Apesar de não conter todo o seu tráfego originado de dispositivos IoT, este conjunto de dados foi escolhido devido à sua relevância na área de detecção de intrusões. Nestes experimentos, a abordagem proposta também é comparada com abordagens clássicas de aprendizado de máquina, como DNN, KNN, ET e RF. Os resultados exibidos foram calculados de acordo com as métricas apresentadas na Seção 5.1. Os resultados individuais de detecção por classe são apresentados no Apêndice A.

Na Tabela 30 são apresentados os resultados gerais dos experimentos com o conjunto de dados NF-UQ-NIDSv2. Observa-se que em relação à acurácia e precisão as abordagens DNN e KNN não foram capazes de alcançar taxas superiores a 99%. A DNN apresentou a menor taxa de acurácia balanceada, aproximadamente 69,7%, indicando que apresentou dificuldades para identificar algumas classes, os resultados individuais por classes são apresentados no Apêndice A. As técnicas RF, ET e a abordagem proposta foram capazes de alcançar taxa de acurácia e precisão superiores a 99%. No entanto, a RF e a ET obtiveram acurácia balanceada de 80,6% e 79,6%, respectivamente, enquanto que a abordagem proposta foi capaz de superar esses valores e alcançar 85,5%. A melhora na acurácia balanceada apresentada pela abordagem proposta reflete nas taxas de detecção individuais de cada classe, que também melhoraram, conforme pode ser observado nas tabelas de resultados individuais apresentadas no Apêndice A.

Os resultados em relação ao tempo de treino e de teste mantiveram-se proporcionalmente similares aos experimentos com os demais conjuntos de dados. A técnica KNN apresentou o menor tempo de treino (69,21 segundos)e o maior tempo de teste (13542,53 segundos). A DNN e a abordagem proposta apresentaram tempos de treino elevado devido ao custo do treinamento dos modelos neurais. O tempo de teste da abordagem proposta nesse experimento mostra-se elevado em relação às técnicas DNN, ET e RF. No entanto, conforme mencionado

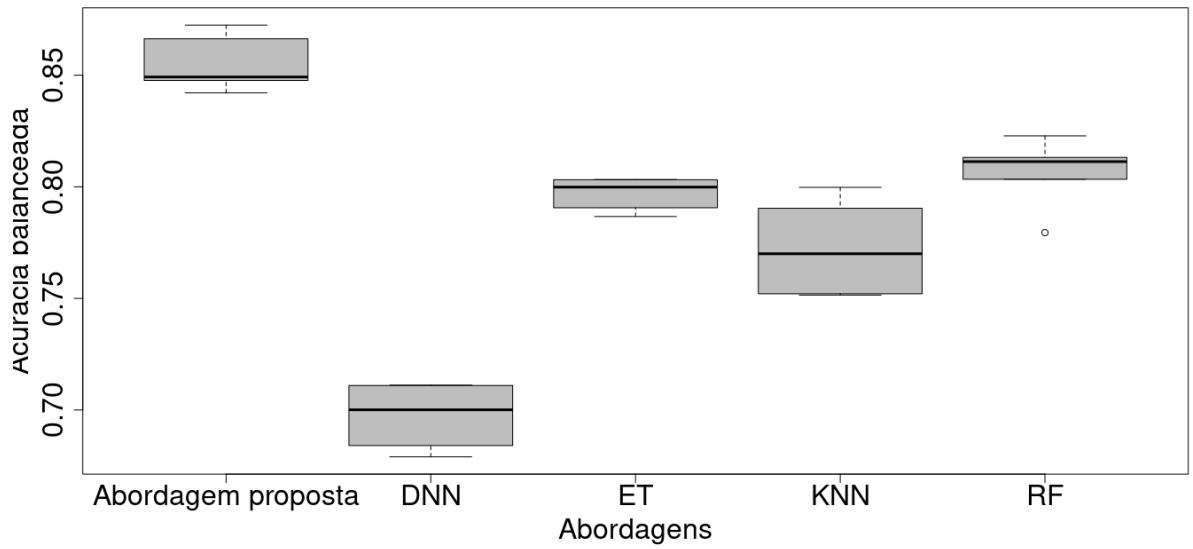
Tabela 30 – Resultados gerais obtidos pelas técnicas clássicas e a abordagem proposta em experimentos com o conjunto de dados NF-UQ-NIDSv2.

Abordagem	**ACC (%)**	**BACC (%)**	**PRE (%)**	**Treino (s)**	**Teste (s)**
RF	99,43%	80,60%	99,41%	1286,2352	54,1911
ET	99,40%	79,67%	99,39%	1028,1170	74,2497
DNN	98,27%	69,70%	98,26%	11611,8942	53,7094
kNN	98,29%	77,27%	98,27%	69,2065	13542,5296
Abordagem proposta	99,05%	85,55%	99,31%	15000,1419	262,9098

em experimentos anteriores, o tráfego intrusivo é analisado pela segunda etapa de detecção e, devido a alta quantidade de tráfego intrusivo presente neste conjunto de dados o tempo de teste aumenta. Em um cenário real, o tráfego benigno é predominante, assim, o tempo de teste tende a ser menor e se aproximar das demais técnicas.

Na Figura 26 é apresentado o gráfico *boxplot* dos resultados das abordagens nos experimentos com o conjunto de dados NF-UQ-NIDSv2. Novamente observa-se que a abordagem proposta apresentou os melhores resultados, o limite inferior obtido foi maior que os limites superiores obtidos pelas demais técnicas.

Figura 26 – Gráfico *boxplot* da acurácia balanceada obtidas pelas técnicas clássicas e a abordagem proposta em experimentos com o conjunto de dados NF-UQ-NIDSv2.



Nestes experimentos, também foram realizados testes estatísticos para verificar se os resultados das taxas de acurácia balanceada da abordagem proposta possuem diferenças estatísticas significativas em relação às demais técnicas. Os resultados dos testes são apresentados na Tabela 31.

Tabela 31 – Testes estatísticos realizados com os resultados de acurácia balanceada dos experimentos realizados no conjunto de dados NF-UQ-NIDSv2.

Shapiro Wilk (Normalidade)		**ANOVA (Significância)**	
W	0,94045	F	69,78
$p-value$	0,1516	$p-value$	1.84e-11

Tukey HSD (Pós-Teste)					
	RF	ET	DNN	KNN	Proposta
RF	1,0000000	0,8745341	**0,0000000**	**0,0216003**	**0,0005302**
ET	0,8745341	1,0000000	**0,0000000**	0,1446157	**0,0000651**
DNN	**0,0000000**	**0,0000000**	1,0000000	**0,0000019**	**0,0000000**
KNN	**0,0216003**	0,1446157	**0,0000019**	1,0000000	**0,0000005**
Proposta	**0,0005302**	**0,0000651**	**0,0000000**	**0,0000005**	1,0000000

O teste de normalidade *Shapiro-Wilk* apresentou um $p-value$ de $0,1516$, portanto, maior que $0,05$, indicando que não há evidências significativas para rejeitar a hipótese de que os dados seguem uma distribuição normal. Para a comparação das médias entre as técnicas, foi utilizado então o método ANOVA, considerando um intervalo de confiança de 95% ($\alpha = 0,05$). O qual gerou como resultado um $p-value$ de $1.84e-11$, portanto, menor que $0,05$. Desse modo, a hipótese de que todas as abordagens são iguais é rejeitada e conclui-se que existe pelo menos uma abordagem que difere significativamente das demais. A partir disso, aplicou-se o pós-teste de *Tukey HSD* com um intervalo de confiança de 95% ($\alpha = 0,05$). Foram observadas diferenças estatísticas significativas entre os resultados da DNN em relação a toda as demais técnicas, comprovando o seu desempenho inferior. Por outro lado, a taxa de acurácia balanceada da abordagem proposta mostraram-se estatisticamente superiores às das demais técnicas avaliadas.

6.3 EXPERIMENTO 03 - AVALIAÇÃO DO PROCESSO DE APRENDIZAGEM FEDERADA

Os resultados obtidos nos experimentos realizados para avaliação sobre a estratégia de treino federado são apresentados na Tabela 32. É importante destacar que em ambos os cenários dos experimentos federados foi possível gerar modelos de detecção capazes de alcançar resultados muito semelhantes aos da abordagem centralizada.

Tabela 32 – Resultados da avaliação realizada com a estratégia de treinamento federado.

Abordagens	**Acurácia**	**Precisão**	**Recall**
Centralizado	98.90	98.91	99.93
Federado 1	98.82	98.85	99.90
Federado 2	98.86	98.87	99.93

O experimento centralizado atingiu aproximadamente 98,9% de acurácia e uma taxa de detecção de ataques de 99,9%. Da mesma forma, o primeiro cenário federado alcançou

98,8% de acurácia e uma taxa de detecção de 99,9%. Por fim, os resultados do segundo cenário federado foram de 98,9% de acurácia e uma taxa de detecção de 99,9%. Portanto, as abordagens federadas utilizadas para treinar os modelos conseguiram atingir resultados muito semelhantes aos da abordagem centralizada. Isso demonstra que o treinamento federado pode ser uma alternativa altamente interessante para realizar o treinamento dos modelos de detecção neurais nesse contexto de computação em nevoeiro sem comprometer a privacidade dos dados.

6.4 ANÁLISE DE COMPLEXIDADE

A seguir, é apresentada uma análise sobre a complexidade de tempo envolvida no método para a classificação de um evento capturado do tráfego. As etapas com maior custo computacional estão relacionadas aos classificadores. As demais etapas, como comparações e atribuições, são consideradas constantes nesta análise.

A arquitetura DNN proposta para o método DNNET possui uma camada de entrada, duas camadas ocultas com m neurônios e uma camada de saída com c neurônios. A camada de entrada possui d neurônios, onde d é o número de dimensões/atributos do evento a ser classificado. Cada camada oculta possui m neurônios, onde m é igual a d. Como a DNN está totalmente conectada, cada nó de uma camada se conecta com um certo peso a todos os nós da próxima camada. Cada neurônio oculto executa uma combinação linear de suas entradas, seguida por uma função de ativação. Portanto, a primeira camada oculta executa $d * m$ operações. A segunda camada oculta executa $m * m$ operações. A camada de saída possui c neurônios e eles também realizam uma combinação linear de suas entradas, seguida por uma função de ativação. Como a arquitetura possui apenas dois neurônios na camada de saída, apenas $m * 2$ operações são realizadas. Desse modo, a complexidade computacional pode ser representada como:

$$\begin{aligned} T_{DNN} &= (d * m) + (m * m) + (m * c) \\ T_{DNN} &= (d * m) + (m * m) + (m * 2) \end{aligned} \tag{6.1}$$

Os custos de classificação da *Extra Tree* (ET) estão relacionados às comparações feitas entre os níveis das árvores de decisão que compõem a ET. Uma comparação por nível é realizada na árvore de decisão, e essa comparação ocorre em um dos nós de cada nível. A Equação 6.2 apresenta o cálculo do número de operações relacionadas à predição com ET. Onde (a) indica o número de árvores e p a profundidade das árvores, ou seja, o número de níveis que elas possuem.

$$T_{ET} = a * p \tag{6.2}$$

Na abordagem DNNET foi considerada uma ET com 10 árvores (a) com 10 níveis de profundidade (p). Portanto, $T_{ET} = 10 * 10$.

Assim, é possível definir a complexidade em relação à abordagem DNNET completa como a Equação 6.3.

$$\begin{aligned} T_{DNNET} &= (T_{DNN} + T_{ET}) \\ T_{DNNET} &= (((d*m) + (m*m) + (m*2)) + (10*10)) \end{aligned} \tag{6.3}$$

A complexidade do segundo nível é apresentada na Equação 6.4. Ela é composta pela soma da complexidade da ET, RF e DNN.

A DNN possui duas camadas ocultas com 150 neurônios e uma camada de saída com c neurônios, correspondentes às classes existentes. Assim, a complexidade pode ser representada por $T_{DNN} = (d*m) + (m*m) + (m*2) = (d*150) + (150*150) + (150*c)$. Conforme mencionado anteriormente, ET tem complexidade $T_{ET} = a*p$. Da mesma forma, RF tem complexidade semelhante $T_{RF} = a*p$. Como a estrutura de RF e ET foi definida com 100 árvores (a) com 100 níveis de profundidade (p), então as complexidades de ET e RF são, $T_{ET} = 100*100$ e $T_{RF} = 100*100$. Além disso, o método *ensemble* conta com uma combinação das classificações, cuja complexidade é irrelevante para esta análise.

$$\begin{aligned} T_{Ensemble} &= T_{DNN} + T_{ET} + T_{RF} \\ T_{Ensemble} &= ((d*150) + (150*150) + (150*c)) \\ &\quad + (100*100) + (100*100) \end{aligned} \tag{6.4}$$

Assim, pode-se concluir que em relação ao tempo de classificação de uma instância de tráfego, o melhor caso ocorre quando o tráfego é classificado diretamente pelo primeiro nível como benigno. Neste caso, a predição pelo segundo nível de detecção não é realizada. Assim, resta apenas a complexidade apresentada na Equação 6.3. Levando em consideração que o número de dimensões (d) de um evento a ser classificado, o número de neurônios em cada camada oculta (m) e o número de neurônios na camada de saída (c) podem ser considerados constantes, a complexidade geral pode ser considerada como $\Theta(1)$. Na melhor das hipóteses, pode-se considerar que a abordagem funciona na ordem $T = \Omega(1)$.

O pior caso ocorre quando o tráfego é classificado pela DNNET como intrusivo, gerando a necessidade de ser classificado pelo método *ensemble* multiclasse de segundo nível. Assim, a complexidade seria composta pela soma da complexidade do primeiro nível (Equação 6.3) com a complexidade do segundo nível (Equação 6.4).

Os parâmetros DNN podem ser considerados constantes, pois possuem relação com a estrutura dos classificadores e não com o tamanho da entrada. Assim, a complexidade geral pode ser pensada como $T = O(1)$. Como a profundidade das árvores é o único ponto que pode variar e gerar maiores operações, pois o nível de profundidade dependerá da etapa de treinamento. Desta forma, foram definidos limites máximos de profundidade para controlar o crescimento das árvores e evitar custos elevados. Além disso, ao analisar a detecção de fluxos em lote, pode-se considerar o tamanho do lote como o tamanho da entrada (n). Assim, a complexidade será $T(n) = O(n)$. Analisando essas complexidades envolvidas, é possível concluir que a abordagem é capaz de suportar a escalabilidade do problema.

A seguir, apresentam-se discussões a respeito dos resultados obtidos nos experimentos com a solução proposta, bem como de outros pontos relevantes.

6.5 DISCUSSÕES

Nesta seção são realizadas algumas discussões a respeito da abordagem proposta e dos resultados obtidos nos experimentos realizados.

A tarefa de identificação de classes é mais complexa que uma detecção binária. Os trabalhos do estado da arte apresentam dificuldades em manter um bom desempenho para as várias classes, com baixa taxa de falsos positivos e sem sobrecarregar o ambiente de nevoeiro e IoT com custos proibitivos.

A abordagem proposta neste trabalho obteve resultados promissores. Ela foi avaliada em relação a técnicas de aprendizado de máquina e a trabalhos do estado da arte. Nos experimentos realizados com as bases de dados NSL-KDD, IoTID20 e NF-UQ-NIDS, é possivel observar que a abordagem proposta foi capaz de manter um bom desempenho de identificação para todas as classes.

Além de apresentar as melhores taxas de detecção em quase todas as classes, a abordagem proposta também alcançou as melhores taxas de acurácia balanceada. A acurácia balanceada é uma métrica interessante para avaliar o desempenho da detecção em conjuntos de dados não balanceados, pois leva em consideração as taxas de detecção de todas as classes existentes. Os resultados positivos se comprovaram através de analise estatística, onde se verificou a existência de diferença estatisticamente significativa entre a abordagem proposta e as demais em termos de acurácia balanceada, tendo a abordagem proposta apresentado maior valor para esta métrica.

Outro ponto de preocupação observado no estado da arte está relacionado à capacidade das abordagens de detecção serem robustas contra problemas de falsos positivos. Através dos resultados dos experimentos, é possível observar que a abordagem proposta conseguiu manter uma baixa taxa de falsos positivos em todos os experimentos realizados. A arquitetura em duas camadas permite que o segundo nível de detecção corrija eventuais falsos positivos gerados no primeiro nível de detecção.

Apesar do bom desempenho de detecção apresentado pela abordagem proposta nos experimentos realizados, é possível observar que ela apresentou um tempo de classificação maior em relação a abordagens clássicas de aprendizado de máquina, cerca de 3 vezes maiores que o obtido pela DNN e pelas técnicas baseadas em árvores de decisão.

No entanto, é necessário destacar que a arquitetura da abordagem proposta possui o primeiro nível de detecção binária, e apenas os eventos detectados como intrusivos serão enviados para o segundo nível onde serão analisados por um método mais robusto. Ou seja, eventos benignos serão analisados somente pela primeira análise. Como a maior parte do tráfego real é geralmente benigna, a abordagem aproximará o tempo de detecção para o tempo obtido pela DNN, pois o primeiro nível da abordagem consiste em uma DNN simples. Isso não se refletiu nos experimentos realizados, pois os conjuntos de dados utilizados possuem muitos exemplos intrusivos. Desse modo, as duas etapas da abordagem são realizadas para um grande número de instâncias, aumentando o tempo de detecção, nesse caso.

Com base nos resultados obtidos nos experimentos realizados, pode-se discutir a pergunta de pesquisa deste trabalho. Os resultados permitem concluir que a abordagem proposta, combinando detecção binária e detecção multiclasse, é capaz de alcançar robustez de detecção e uma baixa taxa de falsos positivos em relação aos demais métodos do estado da arte. Além disso, considerando a análise em termos de tempo de detecção, conclui-se que a abordagem é capaz de operar com um custo computacional similar aos métodos clássicos de aprendizado de máquina.

Em relação ao tempo de treino, a abordagem proposta apresentou um tempo relativamente alto, isso se deve ao fato do método DNNET ser composto pela técnica DNN, que apresenta um custo oneroso de treinamento.

Existem alternativas promissoras para contornar estes problemas de treinamento apresentados pelas abordagens DNN, entre elas a realização do treinamento em dispositivos com maior pode computacional como a computação em nuvem. No entanto, a realização de treinamentos em um dispositivo como a nuvem traz alguns desafios relacionados a privacidade dos dados para estas soluções.

A abordagem proposta para treinamento federado nos nós do nevoeiro se mostrou interessante. Nos experimentos realizados foi possível observar que a solução foi capaz de manter o bom desempenho de detecção em relação ao modelo treinado de maneira centralizada. Além disso, essa abordagem oferece as vantagens relacionadas a preservação da privacidade dos dados locais. A solução permite também reduzir o custo do treinamento nos dispositivos do nevoeiro, pois cada um será treinado apenas com os dados locais. No entanto, essa redução do custo não foi observada nos resultados dos experimentos realizados neste trabalho, pois os experimentos executados não exploraram a característica de execução paralela do treinamento federado, foi realizada apenas uma simulação sequencial das etapas considerando-se os n clientes.

No entanto, há questões em aberto nesse ponto, pois as abordagens de treinamento federado também são suscetíveis a ameaças. Ataques de envenenamento de modelo podem ser realizados por meio de atualizações de modelos corrompidos enviados ao servidor. As abordagens de gerenciamento de confiança podem ser investigadas para se defender contra esses ataques internos. Além disso, devido às questões de privacidade empregadas no FL, é difícil verificar se os modelos recebidos realmente correspondem aos dados de treinamento local ou não (Lalouani; Younis, 2021).

A abordagem proposta também apresenta uma limitação em relação ao treinamento do método *ensemble* da segunda etapa de detecção na nuvem. Como a abordagem opera na nuvem existe a necessidade de enviar dados locais para a nuvem para realizar o treinamento, ferindo assim as restrições citadas anteriormente para preservar a privacidade dos dados. Portanto, encontrar maneiras de realizar o treinamento do método *ensemble* na nuvem preservando a privacidade dos dados locais é um ponto de pequisa em aberto no estado da arte.

Ainda sobre o treinamento da abordagem, existem vários pontos que precisam ser estudados e aprimorados. As abordagens de detecção baseadas em aprendizado de máquina

precisam ser treinadas novamente ao longo do tempo para evitar se tornarem obsoletas. No contexto de IoT, a rede muda ao longo do tempo, novos dispositivos podem ser inseridos e outros removidos. Portanto, os modelos de detecção baseados em técnicas de aprendizado de máquina precisam ser atualizados. Novos dados precisam ser coletados da rede, considerando os novos componentes, para gerar um modelo atualizado capaz de identificar comportamentos verdadeiramente anômalos na nova rede IoT. Além disso, as aplicações IoT podem ser inseridas em um contexto de alta mobilidade de dispositivos, o que, sem dúvida, torna o processo de detecção de intrusão ainda mais desafiador. Assim, outra questão em aberto é entender o tempo máximo que um modelo pode operar sem se tornar obsoleto (Abbasi; Shahraki; Taherkordi, 2021). A estratégia ideal seria retreinar os modelos de detecção sempre que houver alguma alteração na rede, como inserção e remoção de dispositivos. No entanto, é preciso considerar que o processo de treinamento pode ter um alto custo de recursos e sobrecarregar a rede. Os modelos de aprendizado de máquina geralmente sofrem de alta complexidade na fase de treinamento, pois consomem muitos recursos e tempo. Dispositivos no contexto de IoT possuem restrições de recursos, portanto, a complexidade dos modelos de detecção a serem retreinados pode ser considerado um grande desafio.

7 CONCLUSÃO

Com a ampla expansão da Internet e o crescimento das informações manipuladas em sistemas computacionais interligados pela rede, surgiram dificuldades para se manter a segurança dos sistemas. As técnicas de segurança tornaram-se essenciais nos sistemas informáticos modernos. As abordagens de detecção e prevenção de intrusão tem por objetivo detectar, identificar e evitar que ocorram intrusões em redes de computadores. Nesse contexto, é importante detectar e identificar o ataque em uma categoria, para fornecer maiores informações para o responsável e para que possam ser tomadas contramedidas direcionadas à ameaça detectada.

No entanto, a maioria das abordagens concentra-se em métodos de detecção binária e as abordagens de detecção multiclasse existentes têm taxas de acerto mais baixas do que os métodos binários. Ainda mais quando se considera as taxas de detecção de categorias específicas de ataques e a taxa de falsos positivos gerados. Os métodos *ensemble* robustos se mostram promissores mas possuem o desafio do custo computacional de serem implantados em dispositivos com restrições de recursos.

O objetivo deste trabalho foi alcançado através da proposta de uma abordagem hierárquica de múltiplas etapas para detectar e identificar intrusões em ambientes *Fog Computing* e IoT, chamada DNNET-Ensemble. A arquitetura hierárquica permite aproveitar as melhores características dos ambientes de nevoeiro e nuvem, viabilizando a utilização de métodos *ensemble* robustos para detecção de intrusão nesse cenário sem sobrecarregar os dispositivos com recursos limitados. Destaca-se também a proposta de uma abordagem híbrida de detecção binária chamada DNNET, capaz de obter alta taxa de detecção para operar na primeira etapa da abordagem de detecção. A melhoria proposta do *Soft-SMOTE* permite operar com um conjunto de dados balanceado sem gerar um grande aumento no tempo de treinamento, enquanto a estratégia proposta de Seleção Híbrida de Atributos pode encontrar um subconjunto ideal de atributos através de um método *wrapper* com menor custo de treinamento devido à pré-seleção com um método de *filter*. Além disso, a estratégia de treinamento neural federado proposta mostrou-se promissora, alcançando resultados semelhantes aos da abordagem centralizada.

Os resultados obtidos em experimentos com conjuntos de dados de intrusão renomados demonstram que o objetivo do trabalho foi atingido, pois a abordagem pode alcançar desempenho superior a outras técnicas clássicas de aprendizado de máquina em relação a métricas de detecção. O DNNET-Ensemble alcançou uma acurácia média balanceada de 92,6% para o NSLKDD e 99,3% para o IoTID20. Além disso, em comparação com as abordagens do estado da arte, observa-se a capacidade da abordagem proposta de gerar uma baixa taxa de falsos positivos.

Portanto, esses resultados permitem responder a pergunta de pesquisa deste trabalho concluindo que a abordagem proposta com detecção de duas etapas distribuídas em uma arquitetura multicamadas, com a combinação de detecção binária e multiclasse, é capaz de alcançar maior robustez de detecção e baixa taxa de falsos positivos em relação aos demais métodos do estado da arte, além de operar sem custos proibitivos.

7.1 LIMITAÇÕES

O método DNNET-Ensemble proposto neste trabalho foi projetado e avaliado para detectar ataques em ambientes de computação IoT e Fog. Não estava no escopo do projeto detectar outros ataques, aos quais a IoT também é vulnerável, como o roteamento da camada de rede. Ataques como *sinkhole*, *wormhole*, encaminhamento seletivo e ataque de classificação podem ser muito prejudiciais ao funcionamento da rede IoT. Felizmente, encontramos ótimas abordagens destinadas a resolver este problema (Raza; Wallgren; Voigt, 2013; Cervantes et al., 2015; Arshad et al., 2019).

Neste trabalho foram realizados apenas experimentos através de simulação. Portanto, outro ponto a ser destacado é a necessidade de realização de experimentos em um ambiente real para avaliação da abordagem proposta.

Outra limitação deste trabalho foi a realização de experimentos com treinamento federado sem explorar a característica de execução paralela. O objetivo era avaliar se o treinamento federado conseguiria gerar modelo com desempenho de detecção similar ao gerado pela abordagem centralizada. Foi realizada apenas uma simulação sequencial das etapas considerando-se os *n* clientes, devido a restrições na biblioteca utilizada, a *TensorFlow Federated*. No entanto, em trabalhos futuros será interessante realizar experimentos para avaliar também a redução no custo do treinamento da abordagem proposta nos dispositivos do nevoeiro com aprendizado federado.

Além disso, é evidente que o processo de treinamento dos métodos de detecção representa um desafio significativo, que não é totalmente abordado neste trabalho. Primeiramente, as estratégias de treinamento federado empregada para o treinamento do modelo neural da DNNET são suscetíveis a ameaças. Os ataques de envenenamento de modelo podem ser realizados por meio de atualizações de modelo corrompido enviadas ao servidor. As abordagens de gestão de confiança podem ser investigadas para defesa contra esses ataques internos. Além disso, devido às preocupações com a privacidade empregadas na aprendizagem federada, é difícil verificar se os modelos recebidos realmente correspondem aos dados de treinamento locais ou não (Lalouani; Younis, 2021). Notavelmente, pesquisas recentes começaram a abordar estas questões importantes (Liang et al., 2022; Oliveira et al., 2023; Habiba et al., 2023; Cámara et al., 2023). Além disso, conforme relatado anteriormente, existe a limitação da abordagem em relação ao treinamento do método *ensemble* da segunda etapa de detecção na nuvem. Atualmente é necessário enviar dados locais para a nuvem para realizar o treinamento, gerando assim preocupações em relação à privacidade dos dados. Outro ponto, é que neste trabalho não foram abordados desafios importantes, como o retreinamento dos modelos e a preservação da privacidade dos dados no treinamento do método multiclasse da nuvem.

Por fim, este trabalho apresenta apenas uma breve análise a respeito das estratégias existentes para a execução de contramedidas e mitigação de ataques no contexto de computação em nevoeiro e IoT. Em trabalhos futuros será importante realizar pesquisas focadas na etapa de execução de contramedidas através de arquitetura SDN, visando a realização de ações de

mitigação adequadas. Além disso, para prover um ambiente SDN seguro são necessárias mais pesquisas sobre a segurança do controlador SDN.

7.2 TRABALHOS FUTUROS

Como trabalhos futuros, destaca-se a realização de pesquisas focadas na etapa de execução de contramedidas através de arquitetura SDN, visando a realização de ações de mitigação adequadas. Além disso, é importante considerar nesse cenário os aspectos de segurança do próprio controlador SDN.

Outro ponto de pesquisa futuro são as estratégias de treinamento das abordagens de detecção. O treinamento federado no contexto do nevoeiro mostra-se promissor, mas ainda carece de maiores investigações para suportar as ameaças internas existentes. Além disso, este trabalho deixar em aberto a realização de experimentos para avaliar a redução no custo do treinamento nos dispositivos do nevoeiro com aprendizado federado. Realizar o treinamento da abordagem *ensemble* na nuvem preservando a privacidade dos dados locais é também um ponto de pequisa em aberto no estado da arte. Outro ponto de pesquisa futuro, que representa um grande desafio, é a otimização das estratégias para atualização e retreinamento dos modelos.

Neste trabalho, foram realizados apenas experimentos através de simulação. Portanto, outro ponto de trabalhos futuros é a realização de experimentos em um ambiente real para avaliação da abordagem proposta.

REFERÊNCIAS

ABBASI, M.; SHAHRAKI, A.; TAHERKORDI, A. Deep learning for network traffic monitoring and analysis (ntma): A survey. **Computer Communications**, v. 170, p. 19–41, 2021. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366421000426.

ABDEL-BASSET, M. et al. Deep-ifs: Intrusion detection approach for iiot traffic in fog environment. **IEEE Transactions on Industrial Informatics**, IEEE, 2020.

ABDEL-BASSET, M. et al. Semi-supervised spatio-temporal deep learning for intrusions detection in iot networks. **IEEE Internet of Things Journal**, p. 1–1, 2021.

ABDI, H.; WILLIAMS, L. J. Principal component analysis. **WIREs Computational Statistics**, v. 2, n. 4, p. 433–459, 2010. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101.

AHMAD, Z. et al. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. **Transactions on Emerging Telecommunications Technologies**, Wiley Online Library, v. 32, n. 1, p. e4150, 2021.

AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys & Tutorials**, IEEE, v. 17, n. 4, p. 2347–2376, 2015.

AL-GARADI, M. A. et al. A survey of machine and deep learning methods for internet of things (iot) security. **IEEE Communications Surveys Tutorials**, v. 22, n. 3, p. 1646–1685, 2020.

AL-KHAFAJIY, M. et al. Intelligent control and security of fog resources in healthcare systems via a cognitive fog model. **ACM Trans. Internet Technol.**, Association for Computing Machinery, New York, NY, USA, v. 21, n. 3, jun 2021. ISSN 1533-5399. Disponível em: https://doi.org/10.1145/3382770.

ALBDOUR, L.; MANASEER, S.; SHARIEH, A. Iot crawler with behavior analyzer at fog layer for detecting malicious nodes. **Int. J. Commun. Networks Inf. Secur.**, v. 12, n. 1, 2020.

ALBULAYHI, K. et al. Iot intrusion detection using machine learning with a novel high performing feature selection method. **Applied Sciences**, v. 12, n. 10, 2022. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/12/10/5015.

ALHOWAIDE, A.; ALSMADI, I.; TANG, J. Ensemble detection model for iot ids. **Internet of Things**, v. 16, p. 100435, 2021. ISSN 2542-6605. Disponível em: https://www.sciencedirect.com/science/article/pii/S2542660521000792.

ALIYU, F. et al. Human immune-based intrusion detection and prevention system for fog computing. **Journal of Network and Systems Management**, Springer, v. 30, n. 1, p. 1–27, 2022.

ALIYU, F.; SHELTAMI, T.; SHAKSHUKI, E. M. A detection and prevention technique for man in the middle attack in fog computing. **Procedia Computer Science**, v. 141, p. 24 – 31, 2018. ISSN 1877-0509. The 9th International Conference on Emerging

Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops. Disponível em: http://www.sciencedirect.com/science/article/pii/S1877050918317733.

ALMIANI, M. et al. Deep recurrent neural network for iot intrusion detection system. **Simulation Modelling Practice and Theory**, v. 101, p. 102031, 2020. ISSN 1569-190X. Modeling and Simulation of Fog Computing. Disponível em: http://www.sciencedirect.com/science/article/pii/S1569190X19301625.

ALOTAIBI, Y.; ILYAS, M. Ensemble-learning framework for intrusion detection to enhance internet of things devices security. **Sensors**, v. 23, n. 12, 2023. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/23/12/5568.

ALRASHDI, I. et al. Fbad: Fog-based attack detection for iot healthcare in smart cities. In: **2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)**. [S.l.: s.n.], 2019. p. 0515–0522.

ALSAEDI, A. et al. Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems. **IEEE Access**, v. 8, p. 165130–165150, 2020.

ÁLVAREZ-ESTÉVEZ, D. et al. Reducing dimensionality in a database of sleep eeg arousals. **Expert Systems with Applications**, Elsevier, v. 38, n. 6, p. 7746–7754, 2011.

ALY, M. et al. Enforcing security in internet of things frameworks: A systematic literature review. **Internet of Things**, v. 6, p. 100050, 2019. ISSN 2542-6605. Disponível em: https://www.sciencedirect.com/science/article/pii/S2542660518300805.

AMARAL, J. P. et al. Policy and network-based intrusion detection system for ipv6-enabled wireless sensor networks. In: **2014 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2014. p. 1796–1801.

AMMA, N. G. B.; SUBRAMANIAN, S. Vcdeepfl: Vector convolutional deep feature learning approach for identification of known and unknown denial of service attacks. In: **TENCON 2018 - 2018 IEEE Region 10 Conference**. [S.l.: s.n.], 2018. p. 0640–0645.

AN, X. et al. A novel differential game model-based intrusion response strategy in fog computing. **Security and Communication Networks**, Hindawi, v. 2018, 2018.

AN, X. et al. Hypergraph clustering model-based association analysis of ddos attacks in fog computing intrusion detection system. **EURASIP Journal on Wireless Communications and Networking**, SpringerOpen, v. 2018, n. 1, p. 1–9, 2018.

AN, X. et al. Sample selected extreme learning machine based intrusion detection in fog computing and mec. **Wirel. Commun. Mob. Comput.**, John Wiley and Sons Ltd., GBR, v. 2018, jan. 2018. ISSN 1530-8669. Disponível em: https://doi.org/10.1155/2018/7472095.

ANDERSON, J. P. Computer security threat monitoring and surveillance. **Technical Report, James P. Anderson Company**, 1980.

ARAUJO-FILHO, P. Freitas de et al. Intrusion detection for cyber–physical systems using generative adversarial networks in fog environment. **IEEE Internet of Things Journal**, v. 8, n. 8, p. 6247–6256, 2021.

ARBEX, G. V. et al. Iot ddos detection based on stream learning. In: **2021 12th International Conference on Network of the Future (NoF)**. [S.l.: s.n.], 2021. p. 1–8.

ARSHAD, J. et al. Colide: a collaborative intrusion detection framework for internet of things. **IET Networks**, v. 8, n. 1, p. 3–14, 2019.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.

AUBET, F.-X. **Machine Learning-Based Adaptive Anomaly Detection in Smart Spaces**. Tese (Doutorado) — PhD thesis, 04 2018.

AVERSANO, L. et al. A systematic review on deep learning approaches for iot security. **Computer Science Review**, v. 40, p. 100389, 2021. ISSN 1574-0137. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574013721000290.

AYUBKHAN, S. A. H. et al. A practical intrusion detection system based on denoising autoencoder and lightgbm classifier with improved detection performance. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 14, n. 6, p. 7427–7452, 2023.

AZARKASB, S. O.; KASHI, S. S.; KHASTEH, S. H. A network intrusion detection approach at the edge of fog. In: **2021 26th International Computer Conference, Computer Society of Iran (CSICC)**. [S.l.: s.n.], 2021. p. 1–6.

BACE, R.; MELL, P. **NIST special publication on intrusion detection systems**. [S.l.], 2001.

BARANAUSKAS, J. A. et al. Extração automática de conhecimento por múltiplos indutores. 2002.

BARTLETT, P. et al. Boosting the margin: A new explanation for the effectiveness of voting methods. **The annals of statistics**, Institute of Mathematical Statistics, v. 26, n. 5, p. 1651–1686, 1998.

BEBORTTA, S.; DAS, S. K.; CHAKRAVARTY, S. Fog-enabled intelligent network intrusion detection framework for internet of things applications. In: **2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence)**. [S.l.: s.n.], 2023. p. 485–490.

BELANCHE, L. A.; GONZÁLEZ, F. F. Review and evaluation of feature selection algorithms in synthetic problems. **arXiv preprint arXiv:1101.2320**, 2011.

BERRY, M. W.; MOHAMED, A.; YAP, B. W. **Supervised and unsupervised learning for data science**. [S.l.]: Springer, 2019.

BIRKINSHAW, C.; ROUKA, E.; VASSILAKIS, V. G. Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks. **Journal of Network and Computer Applications**, v. 136, p. 71 – 85, 2019. ISSN 1084-8045. Disponível em: http://www.sciencedirect.com/science/article/pii/S1084804519301109.

BLANCO, R. et al. Multiclass network attack classifier using cnn tuned with genetic algorithms. In: **2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)**. [S.l.: s.n.], 2018. p. 177–182.

BOLÓN-CANEDO, V.; ALONSO-BETANZOS, A. Ensembles for feature selection: A review and future trends. **Information Fusion**, Elsevier, v. 52, p. 1–12, 2019.

BOLÓN-CANEDO, V.; SÁNCHEZ-MAROÑO, N.; ALONSO-BETANZOS, A. **Feature selection for high-dimensional data**. [S.l.]: Springer, 2015.

BONOMI, F. et al. Fog computing and its role in the internet of things. In: ACM. **Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. [S.l.], 2012. p. 13–16.

BOUGUILA, N.; ZIOU, D.; VAILLANCOURT, J. Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application. **IEEE Transactions on Image Processing**, v. 13, n. 11, p. 1533–1543, 2004.

BOUKERCHE, A. et al. An agent based and biological inspired real-time intrusion detection and security model for computer network operations. **Computer Communications**, Elsevier, v. 30, n. 13, p. 2649–2660, 2007.

BOWYER, K. W. et al. SMOTE: synthetic minority over-sampling technique. **CoRR**, abs/1106.1813, 2011. Disponível em: http://arxiv.org/abs/1106.1813.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

BREIMAN, L. et al. **Classification and regression trees**. [S.l.]: CRC press, 1984.

BUCZAK, A. L.; GUVEN, E. A survey of data mining and machine learning methods for cyber security intrusion detection. **IEEE Communications Surveys Tutorials**, v. 18, n. 2, p. 1153–1176, 2016.

CÁMARA, X. Sáez-de et al. Clustered federated learning architecture for network anomaly detection in large scale heterogeneous iot networks. **Computers & Security**, Elsevier, v. 131, p. 103299, 2023.

CAMHI, J. Former cisco ceo john chambers predicts 500 billion connected devices by 2025. **Business Insider**, 2015.

CAMPELLO, R. S.; WEBER, R. F. Sistemas de detecçao de intrusao. **Minicurso procedente do 19º Simpósio Brasileiro de Redes de Computadores**, 2001.

Center for Applied Internet Data Analysis. **The CAIDA UCSD DDoS Attack 2007 Dataset**. 2007. Disponível em: https://www.caida.org/data/passive/ddos-20070804_dataset.xml.

Cervantes, C. et al. Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things. In: **2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)**. [S.l.: s.n.], 2015. p. 606–611.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, ACM, Aug 2016. Disponível em: http://dx.doi.org/10.1145/2939672.2939785.

CHOUDHARY, S.; KESSWANI, N. Detection and prevention of routing attacks in internet of things. In: **2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)**. [S.l.: s.n.], 2018. p. 1537–1540.

CHUA, L. O.; YANG, L. Cellular neural networks: Applications. **IEEE Transactions on circuits and systems**, IEEE, v. 35, n. 10, p. 1273–1290, 1988.

CHUANG, S.-H.; YANG, R.-C.; WANG, S.-D. Network intrusion detection system with stream machine learning in fog layer and online labeling in cloud layer. In: **2021 International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)**. [S.l.: s.n.], 2021. p. 53–59.

COSTA, K. A. da et al. Internet of things: A survey on machine learning-based intrusion detection approaches. **Computer Networks**, v. 151, p. 147 – 157, 2019. ISSN 1389-1286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1389128618308739.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE transactions on information theory**, IEEE, v. 13, n. 1, p. 21–27, 1967.

CREECH, G. **Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks.** Tese (Doutorado) — University of New South Wales, Canberra, Australia, 2014.

CREECH, G.; HU, J. Generation of a new ids test dataset: Time to retire the kdd collection. In: **2013 IEEE Wireless Communications and Networking Conference (WCNC)**. [S.l.: s.n.], 2013. p. 4487–4492.

CUNNINGHAM, P.; CARNEY, J.; JACOB, S. Stability problems with artificial neural networks and the ensemble solution. **Artificial Intelligence in Medicine**, v. 20, n. 3, p. 217 – 225, 2000. ISSN 0933-3657. Disponível em: http://www.sciencedirect.com/science/article/pii/S0933365700000658.

DALTON, J.; DESHMANE, A. Artificial neural networks. **IEEE Potentials**, IEEE, v. 10, n. 2, p. 33–36, 1991.

DAT-THINH, N.; XUAN-NINH, H.; KIM-HUNG, L. Midsiot: A multistage intrusion detection system for internet of things. **Wireless Communications and Mobile Computing**, Hindawi, v. 2022, 2022.

DEV, V. A.; EDEN, M. R. Gradient boosted decision trees for lithology classification. In: MUñOZ, S. G.; LAIRD, C. D.; REALFF, M. J. (Ed.). **Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design**. Elsevier, 2019, (Computer Aided Chemical Engineering, v. 47). p. 113 – 118. Disponível em: http://www.sciencedirect.com/science/article/pii/B9780128185971500199.

DIETTERICH, T. G. Ensemble methods in machine learning. In: **Multiple Classifier Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. p. 1–15. ISBN 978-3-540-45014-6.

DIRO, A.; CHILAMKURTI, N. Leveraging lstm networks for attack detection in fog-to-things communications. **IEEE Communications Magazine**, v. 56, n. 9, p. 124–130, 2018.

DIRO, A. A.; CHILAMKURTI, N. Deep learning: The frontier for distributed attack detection in fog-to-things computing. **IEEE Communications Magazine**, v. 56, n. 2, p. 169–175, 2018.

DIRO, A. A.; CHILAMKURTI, N. Distributed attack detection scheme using deep learning approach for internet of things. **Future Generation Computer Systems**, v. 82, p. 761 – 768, 2018. ISSN 0167-739X. Disponível em: http://www.sciencedirect.com/science/article/pii/S0167739X17308488.

DIWAN, T. D. et al. Feature entropy estimation (fee) for malicious iot traffic and detection using machine learning. **Mobile Information Systems**, Hindawi, v. 2021, 2021.

DU, R. et al. Support vector machine intrusion detection scheme based on cloud-fog collaboration. In: SPRINGER. **International Conference on Security and Privacy in New Computing Environments**. [S.l.], 2020. p. 321–334.

DUNN, O. J. Multiple comparisons among means. **Journal of the American statistical association**, Taylor & Francis, v. 56, n. 293, p. 52–64, 1961.

ELRAWY, M. F.; AWAD, A. I.; HAMED, H. F. Intrusion detection systems for iot-based smart environments: a survey. **Journal of Cloud Computing**, Springer, v. 7, n. 1, p. 21, 2018.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Kdd**. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

FARUKEE, M. B. et al. Ddos attack detection in iot networks using deep learning models combined with random forest as feature selector. In: SPRINGER. **International Conference on Advances in Cyber Security**. [S.l.], 2020. p. 118–134.

FERRAG, M. A. et al. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. **Journal of Information Security and Applications**, v. 50, p. 102419, 2020. ISSN 2214-2126. Disponível em: http://www.sciencedirect.com/science/article/pii/S2214212619305046.

FERRI, F. J. et al. Comparative study of techniques for large-scale feature selection. In: **Machine Intelligence and Pattern Recognition**. [S.l.]: Elsevier, 1994. v. 16, p. 403–413.

FISHER, R. A. Statistical methods for research workers. In: **Breakthroughs in statistics**. [S.l.]: Springer, 1992. p. 66–70.

FREUND, Y.; MASON, L. The alternating decision tree learning algorithm. In: **ICML**. [S.l.: s.n.], 1999.

FREUND, Y.; SCHAPIRE, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In: SPRINGER. **European conference on computational learning theory**. [S.l.], 1995. p. 23–37.

GARCIA-MORCHON, O. et al. Security considerations in the ip-based internet of things draft-garciacore-security-06. **Internet Engineering Task Force**, 2013.

GARCÍA, S. et al. An empirical comparison of botnet detection methods. **Computers & Security**, v. 45, p. 100 – 123, 2014. ISSN 0167-4048. Disponível em: http://www.sciencedirect.com/science/article/pii/S0167404814000923.

GARCIA, S.; PARMISANO, A.; ERQUIAGA, M. J. **IoT 23: A labeled dataset with malicious and benign IoT network traffic**. Zenodo, 2020. Disponível em: http://doi.org/10.5281/zenodo.4743746.

GAVEL, S.; RAGHUVANSHI, A. S.; TIWARI, S. Distributed intrusion detection scheme using dual-axis dimensionality reduction for internet of things (iot). **The Journal of Supercomputing**, Springer, p. 1–24, 2021.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. **Machine learning**, Springer, v. 63, n. 1, p. 3–42, 2006.

GHAZI, A. E.; RACHID, A. M. Machine learning and datamining methods for hybrid iot intrusion detection. In: **2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)**. [S.l.: s.n.], 2020. p. 1–6.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

GOODFELLOW, I. et al. **Deep learning**. [S.l.]: MIT press Cambridge, 2016. v. 1.

GOPALAKRISHNAN, B.; PURUSOTHAMAN, P. A new design of intrusion detection in iot sector using optimal feature selection and high ranking-based ensemble learning model. **Peer-to-Peer Networking and Applications**, Springer, p. 1–28, 2022.

GOVINDARAJAN, M.; CHANDRASEKARAN, R. Intrusion detection using neural based hybrid classification methods. **Computer Networks**, v. 55, n. 8, p. 1662 – 1671, 2011. ISSN 1389-1286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1389128610003750.

GP, S.; D'SOUZA, R. Multiclass genetic programming based approach for classification of intrusions. In: **2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)**. [S.l.: s.n.], 2017. p. 74–78.

GUERRA-MANZANARES, A.; BAHSI, H.; NÕMM, S. Hybrid feature selection models for machine learning based botnet detection in iot networks. In: IEEE. **2019 International Conference on Cyberworlds (CW)**. [S.l.], 2019. p. 324–327.

GUYON, I. et al. Gene selection for cancer classification using support vector machines. **Machine learning**, Springer, v. 46, n. 1, p. 389–422, 2002.

HABIBA, M. et al. Edge intelligence for network intrusion prevention in iot ecosystem. **Computers and Electrical Engineering**, Elsevier, v. 108, p. 108727, 2023.

HAGOS, D. H. et al. Enhancing security attacks analysis using regularized machine learning techniques. In: **2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)**. [S.l.: s.n.], 2017. p. 909–918.

HAJIHEIDARI, S. et al. Intrusion detection systems in the internet of things: A comprehensive investigation. **Computer Networks**, v. 160, p. 165 – 191, 2019. ISSN 1389-1286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1389128619306267.

HALEPLIDIS, E. et al. Software-defined networking (sdn): Layers and architecture terminology. In: **RFC 7426**. [S.l.]: IRTF, 2015.

HAMEED, S. S. et al. A hybrid lightweight system for early attack detection in the iomt fog. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 21, n. 24, p. 8289, 2021.

HAYKIN, S. S. **Redes neurais: Princípios e Práticas**. [S.l.]: Bookman, 2001.

HEADY, R. et al. **The architecture of a network level intrusion detection system**. [S.l.], August 1990.

HINDY, H. et al. Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset). In: GHITA, B.; SHIAELES, S. (Ed.). **Selected Papers from the 12th International Networking Conference**. Cham: Springer International Publishing, 2021. p. 73–84. ISBN 978-3-030-64758-2.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Disponível em: https://doi.org/10.1162/neco.1997.9.8.1735.

HOQUE, N.; SINGH, M.; BHATTACHARYYA, D. K. Efs-mi: an ensemble feature selection method for classification. **Complex & Intelligent Systems**, Springer, v. 4, n. 2, p. 105–118, 2018.

HOSSEINI, S.; SARDO, S. R. Network intrusion detection based on deep learning method in internet of thing. **Journal of Reliable Intelligent Environments**, Springer, p. 1–13, 2022.

HOSSEINPOUR, F. et al. An intrusion detection system for fog computing and iot based logistic systems using a smart data approach. **International Journal of Digital Content Technology and its Applications**, Advanced Institute of Convergence IT, v. 10, 2016.

HOU, H. et al. An intrusion detection method for cyber monintoring using attention based hierarchical lstm. In: **2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)**. [S.l.: s.n.], 2022. p. 125–130.

HUANG, G.-B. et al. On-line sequential extreme learning machine. In: . [S.l.: s.n.], 2005. v. 2005, p. 232–237.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: Theory and applications. **Neurocomputing**, v. 70, n. 1, p. 489–501, 2006. ISSN 0925-2312. Neural Networks. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231206000385.

IDRISSI, I.; AZIZI, M.; MOUSSAOUI, O. Iot security with deep learning-based intrusion detection systems: A systematic literature review. In: **2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)**. [S.l.: s.n.], 2020. p. 1–10.

IERACITANO, C. et al. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. **Neurocomputing**, v. 387, p. 51 – 62, 2020. ISSN 0925-2312. Disponível em: http://www.sciencedirect.com/science/article/pii/S0925231219315759.

ILLY, P. et al. Securing fog-to-things environment using intrusion detection system based on ensemble learning. In: **2019 IEEE Wireless Communications and Networking Conference (WCNC)**. [S.l.: s.n.], 2019. p. 1–7.

IWENDI, C. et al. Realizing an efficient iomt-assisted patient diet recommendation system through machine learning model. **IEEE Access**, v. 8, p. 28462–28474, 2020.

JARARWEH, Y. et al. Cloudexp: A comprehensive cloud computing experimental framework. **Simulation Modelling Practice and Theory**, v. 49, p. 180–192, 2014. ISSN 1569-190X. Disponível em: https://www.sciencedirect.com/science/article/pii/S1569190X14001464.

JAVANMARDI, S. et al. An sdn perspective iot-fog security: A survey. **Computer Networks**, v. 229, p. 109732, 2023. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128623001779.

KAFFASH, A.; KAMEL, S. R.; KHEIRABADI, M. **A Two-layer Intrusion Detection System based on Fog and Cloud using Improved KNN and MPNN**. Research Square, 2023. Disponível em: https://doi.org/10.21203/rs.3.rs-3127041/v1.

KALE, R. et al. A hybrid deep learning anomaly detection framework for intrusion detection. In: **2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)**. [S.l.: s.n.], 2022. p. 137–142.

KALNOOR, G.; GOWRISHANKAR, S. Iot-based smart environment using intelligent intrusion detection system. **Soft Computing**, Springer, v. 25, n. 17, p. 11573–11588, 2021.

KAREGOWDA, A. G.; MANJUNATH, A.; JAYARAM, M. Comparative study of attribute selection using gain ratio and correlation based feature selection. **International Journal of Information Technology and Knowledge Management**, v. 2, n. 2, p. 271–277, 2010.

KASONGO, S. M. A deep learning technique for intrusion detection system using a recurrent neural networks based framework. **Computer Communications**, v. 199, p. 113–125, 2023. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366422004601.

KAUR, J.; AGRAWAL, A.; KHAN, R. A. Security issues in fog environment: A systematic literature review. **International Journal of Wireless Information Networks**, Springer, v. 27, p. 467–483, 2020.

KAUR, K.; MITTAL, S. Classification of mammography image with cnn-rnn based semantic features and extra tree classifier approach using lstm. **Materials Today: Proceedings**, 2020. ISSN 2214-7853. Disponível em: http://www.sciencedirect.com/science/article/pii/S2214785320373442.

KAVIANI, S.; SOHN, I. Application of complex systems topologies in artificial neural networks optimization: An overview. **Expert Systems with Applications**, v. 180, p. 115073, 2021. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417421005145.

KDD Cup 1999 Data. **The UCI KDD Archive**. 1999. Disponível em: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

KEELE, S. et al. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.], 2007.

KHAN, S.; PARKINSON, S.; QIN, Y. Fog computing security: a review of current applications and security solutions. **Journal of Cloud Computing**, SpringerOpen, v. 6, n. 1, p. 19, 2017.

KHAN, S. S.; MAILEWA, A. B. Detecting network transmission anomalies using autoencoders-svm neural network on multi-class nsl-kdd dataset. In: **2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)**. [S.l.: s.n.], 2023. p. 0835–0843.

KHAN, Z. A.; HERRMANN, P. A trust based distributed intrusion detection mechanism for internet of things. In: **2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)**. [S.l.: s.n.], 2017. p. 1169–1176.

KHATER, B. S. et al. Classifier performance evaluation for lightweight ids using fog computing in iot security. **Electronics**, Multidisciplinary Digital Publishing Institute, v. 10, n. 14, p. 1633, 2021.

KHATER, B. S. et al. A lightweight perceptron-based intrusion detection system for fog computing. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 9, n. 1, p. 178, 2019.

KHRAISAT, A. et al. Survey of intrusion detection systems: techniques, datasets and challenges. **Cybersecurity**, v. 2, 12 2019.

KIM, J. et al. Long short term memory recurrent neural network classifier for intrusion detection. In: **2016 International Conference on Platform Technology and Service (PlatCon)**. [S.l.: s.n.], 2016. p. 1–5.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering–a systematic literature review. **Information and software technology**, Elsevier, v. 51, n. 1, p. 7–15, 2009.

KITCHENHAM, B.; BRERETON, P. A systematic review of systematic review process research in software engineering. **Information and software technology**, Elsevier, v. 55, n. 12, p. 2049–2075, 2013.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. Citeseer, 2007.

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, P. **Evidence-based software engineering and systematic reviews**. [S.l.]: CRC press, 2015. v. 4.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, v. 78, n. 9, p. 1464–1480, 1990.

KOLIAS, C. et al. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. **IEEE Communications Surveys Tutorials**, v. 18, n. 1, p. 184–208, 2016.

KOLIAS, C. et al. Ddos in the iot: Mirai and other botnets. **Computer**, v. 50, n. 7, p. 80–84, 2017. ISSN 0018-9162.

KOLIAS, C. et al. Learning internet-of-things security"hands-on". **IEEE Security & Privacy**, IEEE, v. 14, n. 1, p. 37–46, 2016.

KORONIOTIS, N. et al. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. **Future Generation Computer Systems**, v. 100, p. 779–796, 2019. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X18327687.

KRISHNAN, P.; DUTTAGUPTA, S.; ACHUTHAN, K. Sdn/nfv security framework for fog-to-things computing infrastructure. **Software: Practice and Experience**, Wiley Online Library, v. 50, n. 5, p. 757–800, 2020.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. **Journal of the American statistical Association**, Taylor & Francis, v. 47, n. 260, p. 583–621, 1952.

KUMAR, P.; GUPTA, G. P.; TRIPATHI, R. A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–18, 2020.

KUMAR, P.; GUPTA, G. P.; TRIPATHI, R. Design of anomaly-based intrusion detection system using fog computing for iot network. **Automatic Control and Computer Sciences**, Springer, v. 55, n. 2, p. 137–147, 2021.

KUMAR, P.; GUPTA, G. P.; TRIPATHI, R. An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for iomt networks. **Computer Communications**, v. 166, p. 110–124, 2021. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366420320090.

KUMAR, P. et al. A distributed framework for detecting ddos attacks in smart contract-based blockchain-iot systems by leveraging fog computing. **Transactions on Emerging Telecommunications Technologies**, n/a, n. n/a, p. e4112, 2020. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4112.

KUMAR, P. et al. Ppsf: A privacy-preserving and secure framework using blockchain-based machine-learning for iot-driven smart cities. **IEEE Transactions on Network Science and Engineering**, v. 8, n. 3, p. 2326–2341, 2021.

KUMAR, P.; TRIPATHI, R.; GUPTA, G. P. P2idf: A privacy-preserving based intrusion detection framework for software defined internet of things-fog (sdiot-fog). In: **Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking**. New York, NY, USA: Association for Computing Machinery, 2021. (ICDCN '21), p. 37–42. ISBN 9781450381840. Disponível em: https://doi.org/10.1145/3427477.3429989.

KUMAR, R. et al. A distributed intrusion detection system to detect ddos attacks in blockchain-enabled iot network. **Journal of Parallel and Distributed Computing**, v. 164, p. 55–68, 2022. ISSN 0743-7315. Disponível em: https://www.sciencedirect.com/science/article/pii/S0743731522000351.

KUMAR, R.; TRIPATHI, R. Dbtp2sf: a deep blockchain-based trustworthy privacy-preserving secured framework in industrial internet of things systems. **Transactions on Emerging Telecommunications Technologies**, Wiley Online Library, v. 32, n. 4, p. e4222, 2021.

LABIOD, Y.; KORBA, A. A.; GHOUALMI, N. Fog computing-based intrusion detection architecture to protect iot networks. **Wireless Personal Communications**, Springer, v. 125, n. 1, p. 231–259, 2022.

LALOUANI, W.; YOUNIS, M. Robust distributed intrusion detection system for edge of things. In: **2021 IEEE Global Communications Conference (GLOBECOM)**. [S.l.: s.n.], 2021. p. 01–06.

LAWAL, M. A.; SHAIKH, R. A.; HASSAN, S. R. An anomaly mitigation framework for iot using fog computing. **Electronics**, v. 9, n. 10, 2020. ISSN 2079-9292. Disponível em: https://www.mdpi.com/2079-9292/9/10/1565.

LAWAL, M. A.; SHAIKH, R. A.; HASSAN, S. R. A ddos attack mitigation framework for iot networks using fog computing. **Procedia Computer Science**, v. 182, p. 13–20, 2021. ISSN 1877-0509. Learning and Technology Conference 2020; Beyond 5G: Paving the way for 6G. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050921004671.

LAZZARINI, R.; TIANFIELD, H.; CHARISSIS, V. A stacking ensemble of deep learning models for iot intrusion detection. **Knowledge-Based Systems**, v. 279, p. 110941, 2023. ISSN 0950-7051. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950705123006913.

LE, T.-T.-H.; KIM, J.; KIM, H. An effective intrusion detection classifier using long short-term memory with gradient descent optimization. In: **2017 International Conference on Platform Technology and Service (PlatCon)**. [S.l.: s.n.], 2017. p. 1–6.

LE, T.-T.-H. et al. Network intrusion detection based on novel feature selection model and various recurrent neural networks. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 9, n. 7, p. 1392, 2019.

LI, C. et al. Detection and defense of ddos attack–based on deep learning in openflow-based sdn. **International Journal of Communication Systems**, Wiley Online Library, v. 31, n. 5, p. e3497, 2018.

LI, W.; AU, M. H.; WANG, Y. A fog-based collaborative intrusion detection framework for smart grid. **International Journal of Network Management**, Wiley Online Library, v. 31, n. 2, p. e2107, 2021.

LIANG, H. et al. An intrusion detection method for advanced metering infrastructure based on federated learning. **Journal of Modern Power Systems and Clean Energy**, p. 1–11, 2022.

LIAW, A.; WIENER, M. et al. Classification and regression by randomforest. **R news**, v. 2, n. 3, p. 18–22, 2002.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: **2008 Eighth IEEE International Conference on Data Mining**. [S.l.: s.n.], 2008. p. 413–422.

LIU, H.; LANG, B. Machine learning and deep learning methods for intrusion detection systems: A survey. **Applied Sciences**, v. 9, n. 20, 2019. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/9/20/4396.

LIU, H.; LANG, B. Machine learning and deep learning methods for intrusion detection systems: A survey. **Applied Sciences**, v. 9, n. 20, 2019. ISSN 2076-3417.

LIU, W. et al. A survey of deep neural network architectures and their applications. **Neurocomputing**, v. 234, p. 11–26, 2017. ISSN 0925-2312. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231216315533.

LIU, Y.; ZHANG, K.; WANG, Z. Intrusion detection of manifold regularized broad learning system based on lu decomposition. **The Journal of Supercomputing**, Springer, p. 1–49, 2023.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [S.l.], 1967. v. 1, n. 14, p. 281–297.

MAHARAJA, R.; IYER, P.; YE, Z. A hybrid fog-cloud approach for securing the internet of things. **Cluster Computing**, Springer, p. 1–9, 2019.

MAKHZANI, A.; FREY, B. **k-Sparse Autoencoders**. 2014.

MANIMURUGAN, S. Iot-fog-cloud model for anomaly detection using improved naïve bayes and principal component analysis. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–10, 2021.

MARÍN-TORDERA, E. et al. Do we all really know what a fog node is? current trends towards an open definition. **Computer Communications**, Elsevier, v. 109, p. 117–130, 2017.

MARQUÉS, A. I.; GARCÍA, V.; SÁNCHEZ, J. S. Exploring the behaviour of base classifiers in credit scoring ensembles. **Expert Systems with Applications**, Elsevier, v. 39, n. 11, p. 10244–10250, 2012.

MCHUGH, J. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. **ACM Transactions on Information and System Security (TISSEC)**, ACM, v. 3, n. 4, p. 262–294, 2000.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 38, n. 2, p. 69–74, 2008.

MCMAHAN, B. et al. Communication-efficient learning of deep networks from decentralized data. In: PMLR. **Artificial intelligence and statistics**. [S.l.], 2017. p. 1273–1282.

MEIDAN, Y. et al. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. **IEEE Pervasive Computing**, IEEE, v. 17, n. 3, p. 12–22, 2018.

MEIR, R.; RÄTSCH, G. An introduction to boosting and leveraging. In: **Advanced lectures on machine learning**. [S.l.]: Springer, 2003. p. 118–183.

MEJÍA-LAVALLE, M.; SUCAR, E.; ARROYO, G. Feature selection with a perceptron neural net. In: **Proceedings of the international workshop on feature selection for data mining**. [S.l.: s.n.], 2006. p. 131–135.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory National, 2011.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. **Ad hoc networks**, Elsevier, v. 10, n. 7, p. 1497–1516, 2012.

MIRANDA, C. et al. A collaborative security framework for software-defined wireless sensor networks. **IEEE Transactions on Information Forensics and Security**, v. 15, p. 2602–2615, 2020.

MITCHELL, R.; CHEN, I.-R. A survey of intrusion detection techniques for cyber-physical systems. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 46, n. 4, mar. 2014. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/2542049.

MITCHELL, T. M. Machine learning. **McGraw Hill Science/Engineering/Math**, p. 432, 1997.

MOURAD, A. et al. Ad-hoc vehicular fog enabling cooperative low-latency intrusion detection. **IEEE Internet of Things Journal**, IEEE, 2020.

MOURAD, A. et al. Ad hoc vehicular fog enabling cooperative low-latency intrusion detection. **IEEE Internet of Things Journal**, v. 8, n. 2, p. 829–843, 2021.

MOUSSA, M. M.; ALAZZAWI, L. Cyber attacks detection based on deep learning for cloud-dew computing in automotive iot applications. In: **2020 IEEE International Conference on Smart Cloud (SmartCloud)**. [S.l.: s.n.], 2020. p. 55–61.

MOUSTAFA, N. **ToN_IoT datasets**. IEEE Dataport, 2019. Disponível em: https://dx.doi.org/10.21227/fesz-dm97.

MOUSTAFA, N.; AHMED, M.; AHMED, S. Data analytics-enabled intrusion detection: Evaluations of ton_iot linux datasets. In: **2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)**. [S.l.: s.n.], 2020. p. 727–735.

MOUSTAFA, N. et al. Outlier dirichlet mixture mechanism: Adversarial statistical learning for anomaly detection in the fog. **IEEE Transactions on Information Forensics and Security**, v. 14, n. 8, p. 1975–1987, 2019.

MOUSTAFA, N. et al. Dad: A distributed anomaly detection system using ensemble one-class statistical learning in edge networks. **Future Generation Computer Systems**, v. 118, p. 240–251, 2021. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X21000212.

MOUSTAFA, N.; SLAY, J. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: **2015 Military Communications and Information Systems Conference (MilCIS)**. [S.l.: s.n.], 2015. p. 1–6.

MUHAMMAD, F.; ANJUM, W.; MAZHAR, K. S. A critical analysis on the security concerns of internet of things (iot). **International Journal of Computer Applications**, v. 111, n. 7, 2015.

MUKHERJEE, B.; HEBERLEIN, L. T.; LEVITT, K. N. Network intrusion detection. **IEEE network**, IEEE, v. 8, n. 3, p. 26–41, 1994.

NASIR, M. et al. Feature engineering and deep learning-based intrusion detection framework for securing edge iot. **The Journal of Supercomputing**, Springer, v. 78, n. 6, p. 8852–8866, 2022.

NAVAS, R. E. et al. Do not trust your neighbors! a small iot platform illustrating a man-in-the-middle attack. In: SPRINGER. **International Conference on Ad-Hoc Networks and Wireless**. [S.l.], 2018. p. 120–125.

NESHENKO, N. et al. Demystifying iot security: an exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. **IEEE Communications Surveys & Tutorials**, IEEE, v. 21, n. 3, p. 2702–2733, 2019.

NG, A. et al. Sparse autoencoder. **CS294A Lecture notes**, v. 72, n. 2011, p. 1–19, 2011.

NG, B. A.; SELVAKUMAR, S. Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment. **Future Generation Computer Systems**, Elsevier, v. 113, p. 255–265, 2020.

NGUYEN, T. G. et al. Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks. **IEEE Access**, v. 7, p. 107678–107694, 2019.

NI, J. et al. Securing fog computing for internet of things applications: Challenges and solutions. **IEEE Communications Surveys & Tutorials**, IEEE, 2018.

NIU, Z. et al. A novel anomaly detection approach based on ensemble semi-supervised active learning (adessa). **Computers & Security**, v. 129, p. 103190, 2023. ISSN 0167-4048. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167404823001001.

NOBAKHT, M.; SIVARAMAN, V.; BORELI, R. A host-based intrusion detection and mitigation framework for smart home iot using openflow. In: **2016 11th International Conference on Availability, Reliability and Security (ARES)**. [S.l.: s.n.], 2016. p. 147–156.

NORTHCUTT, S. et al. **Intrusion Signatures and Analysis**. New Jersey: New Riders, 2001.

NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. **CoRR**, abs/1811.03378, 2018. Disponível em: http://arxiv.org/abs/1811.03378.

OLIVEIRA, J. A. de et al. F-nids—a network intrusion detection system based on federated learning. **Computer Networks**, Elsevier, p. 110010, 2023.

OMAR, H. O. M.; GOYAL, S. B.; VARADARAJAN, V. Application of sliding window deep learning for intrusion detection in fog computing. In: **2021 Emerging Trends in Industry 4.0 (ETI 4.0)**. [S.l.: s.n.], 2021. p. 1–6.

ONAH, J. O. et al. Genetic algorithm based feature selection and naïve bayes for anomaly detection in fog computing environment. **Machine Learning with Applications**, v. 6, p. 100156, 2021. ISSN 2666-8270. Disponível em: https://www.sciencedirect.com/science/article/pii/S2666827021000785.

OR-MEIR, O. et al. Dynamic malware analysis in the modern era—a state of the art survey. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 5, set. 2019. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3329786.

ORIOL, M.; MARCO, J.; FRANCH, X. Quality models for web services: A systematic mapping. **Information and software technology**, Elsevier, v. 56, n. 10, p. 1167–1182, 2014.

PACHECO, J.; BENITEZ, V.; FéLIX-HERRáN, L. Anomaly behavior analysis for iot network nodes. In: **Proceedings of the 3rd International Conference on Future Networks and Distributed Systems**. New York, NY, USA: Association for Computing Machinery, 2019. (ICFNDS '19). ISBN 9781450371636. Disponível em: https://doi.org/10.1145/3341325.3342008.

PACHECO, J. et al. Artificial neural networks-based intrusion detection system for internet of things fog nodes. **IEEE Access**, v. 8, p. 73907–73918, 2020.

PAL, N. et al. A possibilistic fuzzy c-means clustering algorithm. **IEEE Transactions on Fuzzy Systems**, v. 13, n. 4, p. 517–530, 2005.

PAN, J.-S. et al. A lightweight intelligent intrusion detection model for wireless sensor networks. **Security and Communication Networks**, Hindawi, v. 2021, 2021.

PAN, Z.; PACHECO, J.; HARIRI, S. Anomaly behavior analysis for building automation systems. In: **2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)**. [S.l.: s.n.], 2017. p. 1–8.

PANG, K. Self-organizing maps. **J Neural Networks**, 2003.

PARANJOTHI, A.; ATIQUZZAMAN, M. A statistical approach for enhancing security in vanets with efficient rogue node detection using fog computing. **Digital Communications and Networks**, 2021. ISSN 2352-8648. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352864821000705.

PATEL, A.; QASSIM, Q.; WILLS, C. A survey of intrusion detection and prevention systems. **Information Management & Computer Security**, Emerald Group Publishing Limited, 2010.

PENG, K. et al. Intrusion detection system based on decision tree over big data in fog environment. **Wireless Communications and Mobile Computing**, Hindawi, v. 2018, 2018.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, Elsevier, v. 64, p. 1–18, 2015.

PIMENTEL, B. A. Métodos de agrupamento difuso multivariado baseados no fuzzy c-means. Universidade Federal de Pernambuco, 2017.

PIROZMAND, P. et al. Intrusion detection into cloud-fog-based iot networks using game theory. **Wireless Communications and Mobile Computing**, Hindawi, v. 2020, 2020.

PONNUSAMY, V.; SHARMA, B. Investigation on iot intrusion detection in wireless environment. In: **2021 International Conference on Computer Information Sciences (ICCOINS)**. [S.l.: s.n.], 2021. p. 7–13.

POTRINO, G.; RANGO, F. D.; FAZIO, P. A distributed mitigation strategy against dos attacks in edge computing. In: **2019 Wireless Telecommunications Symposium (WTS)**. [S.l.: s.n.], 2019. p. 1–7.

POTRINO, G.; RANGO, F. de; SANTAMARIA, A. F. Modeling and evaluation of a new iot security system for mitigating dos attacks to the mqtt broker. In: **2019 IEEE Wireless Communications and Networking Conference (WCNC)**. [S.l.: s.n.], 2019. p. 1–6.

PRABAVATHY, S.; SUNDARAKANTHAM, K.; SHALINIE, S. M. Design of cognitive fog computing for intrusion detection in internet of things. **Journal of Communications and Networks**, v. 20, n. 3, p. 291–298, June 2018. ISSN 1229-2370.

PRIYADARSHINI, R.; BARIK, R. K. A deep learning based intelligent framework to mitigate ddos attack in fog environment. **Journal of King Saud University - Computer and Information Sciences**, v. 34, n. 3, p. 825–831, 2022. ISSN 1319-1578. Disponível em: https://www.sciencedirect.com/science/article/pii/S1319157818310140.

PROTOGEROU, A. et al. A graph neural network method for distributed anomaly detection in iot. **Evolving Systems**, Springer, p. 1–18, 2020.

QADDOURA, R. et al. Predicting different types of imbalanced intrusion activities based on a multi-stage deep learning approach. In: **2021 International Conference on Information Technology (ICIT)**. [S.l.: s.n.], 2021. p. 858–863.

QADDOURA, R. et al. A multi-layer classification approach for intrusion detection in iot networks based on deep learning. **Sensors**, v. 21, n. 9, 2021. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/21/9/2987.

QUINLAN, J. R. Induction of decision trees. **Machine learning**, Springer, v. 1, n. 1, p. 81–106, 1986.

QUINLAN, J. R. C4.5: Programming for machine learning. **Morgan Kauffmann**, v. 38, 1993.

RAHMAN, M. A. et al. Scalable machine learning-based intrusion detection system for iot-enabled smart cities. **Sustainable Cities and Society**, p. 102324, 2020. ISSN 2210-6707. Disponível em: http://www.sciencedirect.com/science/article/pii/S221067072030545X.

RANGISETTI, A. K.; DWIVEDI, R.; SINGH, P. Denial of arp spoofing in sdn and nfv enabled cloud-fog-edge platforms. **Cluster Computing**, Springer, v. 24, n. 4, p. 3147–3172, 2021.

RATHORE, S.; PARK, J. H. Semi-supervised learning based distributed attack detection framework for iot. **Applied Soft Computing**, v. 72, p. 79 – 89, 2018. ISSN 1568-4946. Disponível em: http://www.sciencedirect.com/science/article/pii/S1568494618303508.

RATHORE, S.; Wook Kwon, B.; PARK, J. H. Blockseciotnet: Blockchain-based decentralized security architecture for iot network. **Journal of Network and Computer Applications**, v. 143, p. 167–177, 2019. ISSN 1084-8045. Disponível em: https://www.sciencedirect.com/science/article/pii/S1084804519302243.

RAVI, N.; SHALINIE, S. M. Semi-supervised learning based security to detect and mitigate intrusions in iot network. **IEEE Internet of Things Journal**, p. 1–1, 2020.

RAZA, S.; WALLGREN, L.; VOIGT, T. Svelte: Real-time intrusion detection in the internet of things. **Ad hoc networks**, Elsevier, v. 11, n. 8, p. 2661–2674, 2013.

RAZAQUE, A. et al. Energy-efficient and secure mobile fog-based cloud for the internet of things. **Future Generation Computer Systems**, v. 127, p. 1–13, 2022. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X21003320.

REBBAH, M.; REBBAH, D. E. H.; SMAIL, O. Intrusion detection in cloud internet of things environment. In: **2017 International Conference on Mathematics and Information Technology (ICMIT)**. [S.l.: s.n.], 2017. p. 65–70.

REDDY, D. K. K. et al. Exact greedy algorithm based split finding approach for intrusion detection in fog-enabled iot environment. **Journal of Information Security and Applications**, v. 60, p. 102866, 2021. ISSN 2214-2126. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214212621000971.

REY, V. et al. Federated learning for malware detection in iot devices. **Computer Networks**, v. 204, p. 108693, 2022. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128621005582.

RING, M. et al. Flow-based benchmark data sets for intrusion detection. In: **Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI**. [S.l.: s.n.], 2017. p. 361–369.

ROKACH, L. Decision forest: Twenty years of research. **Information Fusion**, v. 27, p. 111 – 125, 2016. ISSN 1566-2535. Disponível em: http://www.sciencedirect.com/science/article/pii/S1566253515000561.

ROMAN, R.; LOPEZ, J.; MAMBO, M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. **Future Generation Computer Systems**, Elsevier, v. 78, p. 680–698, 2018.

ROOPAK, M.; TIAN, G. Y.; CHAMBERS, J. Deep learning models for cyber security in iot networks. In: **2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)**. [S.l.: s.n.], 2019. p. 0452–0457.

ROOPAK, M.; TIAN, G. Y.; CHAMBERS, J. Deep learning models for cyber security in iot networks. In: IEEE. **2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)**. [S.l.], 2019. p. 0452–0457.

ROSA, W. A. **Avaliação de Métodos de Inteligência Computacional para Detecção de Intrusão**. 2017. Monografia (Bacharel em Ciência da Computação), UNIOESTE (Universidade Estadual do Oeste do Paraná), Foz do Iguaçu, Brazil.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: A Modern Approach**. [S.l.]: Pearson Education, Inc., 2010.

SADAF, K.; SULTANA, J. Intrusion detection based on autoencoder and isolation forest in fog computing. **IEEE Access**, IEEE, v. 8, p. 167059–167068, 2020.

SAHAR, N.; MISHRA, R.; KALAM, S. Deep learning approach-based network intrusion detection system for fog-assisted iot. In: SPRINGER. **Proceedings of international conference on big data, machine learning and their applications**. [S.l.], 2021. p. 39–50.

SAHI, M.; SONI, M.; AULUCK, N. An intrusion detection system on fog architecture. In: **2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)**. [S.l.: s.n.], 2021. p. 591–596.

SAMAT, A. et al. E^2LMs : Ensemble extreme learning machines for hyperspectral image classification. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, v. 7, n. 4, p. 1060–1069, 2014.

SAMY, A.; YU, H.; ZHANG, H. Fog-based attack detection framework for internet of things using deep learning. **IEEE Access**, v. 8, p. 74571–74585, 2020.

SANDHU, R.; SOHAL, A. S.; SOOD, S. K. Identification of malicious edge devices in fog computing environments. **Information Security Journal: A Global Perspective**, Taylor & Francis, v. 26, n. 5, p. 213–228, 2017. Disponível em: https://doi.org/10.1080/19393555.2017.1334843.

SARHAN, M. et al. Netflow datasets for machine learning-based network intrusion detection systems. **arXiv preprint arXiv:2011.09144**, 2020.

SARHAN, M. et al. Towards a standard feature set of nids datasets. **arXiv preprint arXiv:2101.11315**, 2021.

SARWAR, A. et al. Design of an advance intrusion detection system for iot networks. In: **2022 2nd International Conference on Artificial Intelligence (ICAI)**. [S.l.: s.n.], 2022. p. 46–51.

SATYANARAYANAN, M. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. **GetMobile: Mobile Computing and Communications**, ACM, v. 18, n. 4, p. 19–23, 2015.

SCHAPIRE, R. E. The strength of weak learnability. **Machine learning**, Springer, v. 5, n. 2, p. 197–227, 1990.

SHAFI, Q. et al. Fog-assisted sdn controlled framework for enduring anomaly detection in an iot network. **IEEE Access**, PP, p. 1–1, 11 2018.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, JSTOR, v. 52, n. 3/4, p. 591–611, 1965.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: **Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP**. [S.l.: s.n.], 2018. p. 108–116.

SHARAFALDIN, I. et al. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: **2019 International Carnahan Conference on Security Technology (ICCST)**. [S.l.: s.n.], 2019. p. 1–8.

SHEN, S. et al. Multistage signaling game-based optimal detection strategies for suppressing malware diffusion in fog-cloud-based iot networks. **IEEE Internet of Things Journal**, v. 5, n. 2, p. 1043–1054, April 2018. ISSN 2327-4662.

SHEN, S. et al. Multistage signaling game-based optimal detection strategies for suppressing malware diffusion in fog-cloud-based iot networks. **IEEE Internet of Things Journal**, IEEE, v. 5, n. 2, p. 1043–1054, 2018.

SHIRAVI, A. et al. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. **Computers & Security**, v. 31, n. 3, p. 357 – 374, 2012. ISSN 0167-4048. Disponível em: http://www.sciencedirect.com/science/article/pii/S0167404811001672.

SHREENIVAS, D.; RAZA, S.; VOIGT, T. Intrusion detection in the rpl-connected 6lowpan networks. In: ACM. **Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security**. [S.l.], 2017. p. 31–38.

SICARI, S. et al. Security, privacy and trust in internet of things: The road ahead. **Computer Networks**, v. 76, p. 146 – 164, 2015. ISSN 1389-1286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1389128614003971.

SOHAL, A. S. et al. A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. **Computers & Security**, v. 74, p. 340 – 354, 2018. ISSN 0167-4048. Disponível em: http://www.sciencedirect.com/science/article/pii/S0167404817301827.

SOUZA, C. A.; WESTPHALL, C. B.; MACHADO, R. B. Two-step ensemble approach for intrusion detection and identification in iot and fog computing environments. **Computers & Electrical Engineering**, v. 98, p. 107694, 2022. ISSN 0045-7906. Disponível em: https://www.sciencedirect.com/science/article/pii/S0045790622000155.

SOUZA, C. A. d. et al. Método híbrido de detecção de intrusão aplicando inteligência artificial. Universidade Estadual do Oeste do Paraná, 2018.

SOUZA, C. A. de et al. Hybrid approach to intrusion detection in fog-based iot environments. **Computer Networks**, v. 180, p. 107417, 2020. ISSN 1389-1286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1389128619315439.

SOUZA, C. A. de et al. Intrusion detection and prevention in fog based iot environments: A systematic literature review. **Computer Networks**, p. 109154, 2022. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128622002651.

STIAWAN, D. et al. Cicids-2017 dataset feature analysis with information gain for anomaly detection. **IEEE Access**, IEEE, v. 8, p. 132911–132921, 2020.

SUGI, S. S. S.; RATNA, S. R. A novel distributed training on fog node in iot backbone networks for security. Springer, 2020.

SUNDHARI, S. S. A knowledge discovery using decision tree by gini coefficient. In: **2011 International Conference on Business, Engineering and Industrial Applications**. [S.l.: s.n.], 2011. p. 232–235.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. [S.l.]: MIT press, 2018.

TAKABI, H.; JOSHI, J. B. D.; AHN, G. Security and privacy challenges in cloud computing environments. **IEEE Security Privacy**, v. 8, n. 6, p. 24–31, 2010.

TAMA, B. A.; LIM, S. Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation. **Computer Science Review**, v. 39, p. 100357, 2021. ISSN 1574-0137. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574013720304573.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. Introduction to data mining, pearson education. **Inc., New Delhi**, 2006.

TANAKA, H.; YAMAGUCHI, S. On modeling and simulation of the behavior of iot malwares mirai and hajime. In: **2017 IEEE International Symposium on Consumer Electronics (ISCE)**. [S.l.: s.n.], 2017. p. 56–60.

TANEJA, S. et al. Mfz-knn — a modified fuzzy based k nearest neighbor algorithm. In: **2015 International Conference on Cognitive Computing and Information Processing(CCIP)**. [S.l.: s.n.], 2015. p. 1–5.

TAVALLAEE, M. et al. A detailed analysis of the kdd cup 99 data set. In: IEEE. **Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on**. [S.l.], 2009. p. 1–6.

THI-NGA, D. et al. An optimal packet assignment algorithm for multi-level network intrusion detection systems. In: SPRINGER. **International Conference on Industrial Networks and Intelligent Systems**. [S.l.], 2020. p. 301–313.

T.K., B.; ANNAVARAPU, C. S. R.; BABLANI, A. Machine learning algorithms for social media analysis: A survey. **Computer Science Review**, v. 40, p. 100395, 2021. ISSN 1574-0137. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574013721000356.

TONG, S.; KOLLER, D. Support vector machine active learning with applications to text classification. **J. Mach. Learn. Res.**, JMLR.org, v. 2, p. 45–66, mar. 2002. ISSN 1532-4435. Disponível em: https://doi.org/10.1162/153244302760185243.

TRAGANITIS, P. A.; PAGèS-ZAMORA, A.; GIANNAKIS, G. B. Blind multiclass ensemble classification. **IEEE Transactions on Signal Processing**, v. 66, n. 18, p. 4737–4752, 2018.

TSAI, C.-F.; HSU, Y.-F.; YEN, D. C. A comparative study of classifier ensembles for bankruptcy prediction. **Applied Soft Computing**, Elsevier, v. 24, p. 977–984, 2014.

TU, S. et al. Security in fog computing: A novel technique to tackle an impersonation attack. **IEEE Access**, v. 6, p. 74993–75001, 2018.

TUKEY, J. W. Comparing individual means in the analysis of variance. **Biometrics**, JSTOR, p. 99–114, 1949.

ULLAH, I.; MAHMOUD, Q. H. A scheme for generating a dataset for anomalous activity detection in IoT networks. In: **Advances in Artificial Intelligence**. Springer International Publishing, 2020. p. 508–520. Disponível em: https://doi.org/10.1007/978-3-030-47358-7_52.

ULLAH, I. et al. Software defined network enabled fog-to-things hybrid deep learning driven cyber threat detection system. **Security and Communication Networks**, Hindawi, v. 2021, 2021.

VACCARI, I. et al. Mqttset, a new dataset for machine learning techniques on mqtt. **Sensors**, v. 20, n. 22, 2020. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/20/22/6578.

VALENCIO, J. D. G. **Abordagem de detecção de intrusão em ambientes fog computing e internet of things**. 106 p. Dissertação (Mestrado) — Programa de Pós-Graduação em Engenharia Elétrica e Computação) - Universidade Estadual do Oeste do Paraná, Foz do Iguaçu-PR, 2022.

VERMA, A.; RANGA, V. Evaluation of network intrusion detection systems for rpl based 6lowpan networks in iot. **Wireless Personal Communications**, Springer, v. 108, n. 3, p. 1571–1594, 2019.

VERMA, A.; RANGA, V. Machine learning based intrusion detection systems for iot applications. **Wireless Personal Communications**, Springer, v. 111, n. 4, p. 2287–2310, 2020.

VINAYAKUMAR, R.; SOMAN, K. P.; POORNACHANDRAN, P. Evaluating effectiveness of shallow and deep networks to intrusion detection system. In: **2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [S.l.: s.n.], 2017. p. 1282–1289.

VISHWAKARMA, M.; KESSWANI, N. A new two-phase intrusion detection system with naïve bayes machine learning for data classification and elliptic envelop method for anomaly detection. **Decision Analytics Journal**, v. 7, p. 100233, 2023. ISSN 2772-6622. Disponível em: https://www.sciencedirect.com/science/article/pii/S2772662223000735.

WANG, G. et al. A comparative assessment of ensemble learning for credit scoring. **Expert systems with applications**, Elsevier, v. 38, n. 1, p. 223–230, 2011.

WANG, S.; XU, W.; LIU, Y. Res-tranbilstm: An intelligent approach for intrusion detection in the internet of things. **Computer Networks**, v. 235, p. 109982, 2023. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128623004279.

WANG, Y. et al. A fog-based privacy-preserving approach for distributed signature-based intrusion detection. **Journal of Parallel and Distributed Computing**, v. 122, p. 26 – 35, 2018. ISSN 0743-7315. Disponível em: http://www.sciencedirect.com/science/article/pii/S0743731518305057.

XU, H. et al. A data-driven approach for intrusion and anomaly detection using automated machine learning for the internet of things. **Soft Computing**, Springer, p. 1–13, 2023.

XU, S.; QIAN, Y.; HU, R. Q. A semi-supervised learning approach for network anomaly detection in fog computing. In: **ICC 2019 - 2019 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2019. p. 1–6.

YAHYAOUI, A. et al. Read-iot: Reliable event and anomaly detection framework for the internet of things. **IEEE Access**, v. 9, p. 24168–24186, 2021.

YAN, Q. et al. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. **IEEE Communications Surveys Tutorials**, v. 18, n. 1, p. 602–622, 2016.

YASEEN, Q. et al. Leveraging fog computing and software defined systems for selective forwarding attacks detection in mobile wireless sensor networks. **Transactions on Emerging Telecommunications Technologies**, v. 29, n. 4, p. e3183, 2017. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3183.

YASEEN, Q. et al. A fog computing based system for selective forwarding detection in mobile wireless sensor networks. In: **2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)**. [S.l.: s.n.], 2016. p. 256–262.

YASEEN, Q. et al. Collusion attacks mitigation in internet of things: a fog based model. **Multimedia Tools and Applications**, v. 77, n. 14, p. 18249–18268, Jul 2018. ISSN 1573-7721. Disponível em: https://doi.org/10.1007/s11042-017-5288-3.

YASEEN, Q. et al. Collusion attacks in internet of things: Detection and mitigation using a fog based model. In: **2017 IEEE Sensors Applications Symposium (SAS)**. [S.l.: s.n.], 2017. p. 1–5.

ZAHRA, S. R.; CHISHTI, M. A. Fuzzy logic and fog based secure architecture for internet of things (flfsiot). **Journal of ambient intelligence and humanized computing**, Springer, p. 1–25, 2020.

ZAHRA, S. R.; CHISHTI, M. A. A generic and lightweight security mechanism for detecting malicious behavior in the uncertain internet of things using fuzzy logic-and fog-based approach. **Neural Computing and Applications**, Springer, p. 1–26, 2022.

ZAINAL, A. et al. Ensemble classifiers for network intrusion detection system. **Journal of Information Assurance and Security**, v. 4, n. 3, p. 217–225, 2009.

ZARPELÃO, B. B. et al. A survey of intrusion detection in internet of things. **Journal of Network and Computer Applications**, v. 84, p. 25 – 37, 2017. ISSN 1084-8045. Disponível em: http://www.sciencedirect.com/science/article/pii/S1084804517300802.

ZHANG, C. et al. A novel framework design of network intrusion detection based on machine learning techniques. **Security and Communication Networks**, Hindawi, v. 2021, 2021.

ZHANG, Y.; WEI, J.; WANG, K. An edge ids based on biological immune principles for dynamic threat detection. **Wireless Communications and Mobile Computing**, Hindawi, v. 2020, 2020.

ZHAO, R. et al. A hybrid intrusion detection system based on feature selection and weighted stacking classifier. **IEEE Access**, p. 1–14, 2022.

ZHOU, J. et al. Graph neural networks: A review of methods and applications. **AI Open**, v. 1, p. 57–81, 2020. ISSN 2666-6510. Disponível em: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

ZHOU, L.; GUO, H.; DENG, G. A fog computing based approach to ddos mitigation in iiot systems. **Computers & Security**, Elsevier, v. 85, p. 51–62, 2019.

ZHOU, Y. et al. Building an efficient intrusion detection system based on feature selection and ensemble classifier. **Computer Networks**, Elsevier, p. 107247, 2020.

Apêndices

APÊNDICE A – RESULTADOS DOS EXPERIMENTOS COM O CONJUNTO DE DADOS NF-UQ-NIDSV2

Tabela 33 – Resultados dos experimentos da abordagem RF com conjunto de dados NF-UQ-NIDSv2.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,54%	99,76%	99,65%
DoS		99,78%	99,94%	99,86%
DDoS		99,89%	99,70%	99,79%
Reconnaissance		99,75%	99,72%	99,74%
XSS		96,54%	99,03%	97,77%
Password		97,25%	96,86%	97,06%
Scanning		98,91%	98,43%	98,67%
Injection		95,34%	89,96%	92,57%
Bot		100,00%	100,00%	100,00%
BruteForce		99,31%	97,81%	98,55%
Infiltration		79,25%	37,65%	51,04%
Exploits		83,72%	91,18%	87,28%
Backdoor		99,84%	98,34%	99,09%
Fuzzers		83,16%	90,30%	86,57%
MITM		92,43%	32,74%	48,30%
Ransomware		98,55%	94,80%	96,63%
Generic		73,19%	51,43%	60,20%
Theft		96,67%	88,48%	92,35%
Shellcode		84,59%	82,86%	83,66%
Analysis		26,67%	3,64%	6,19%
Worms		51,67%	40,00%	44,05%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
1286,2352	54,1911	99,43%	80,60%	99,41%

Tabela 34 – Resultados dos experimentos da abordagem ET com conjunto de dados NF-UQ-NIDSv2.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,51%	99,77%	99,64%
DoS		99,80%	99,91%	99,85%
DDoS		99,86%	99,72%	99,79%
Reconnaissance		99,73%	99,72%	99,73%
XSS		96,46%	98,78%	97,61%
Password		96,82%	96,68%	96,75%
Scanning		98,36%	98,37%	98,36%
Injection		94,71%	89,36%	91,96%
Bot		100,00%	100,00%	100,00%
BruteForce		99,17%	97,78%	98,47%
Infilteration		91,51%	33,89%	49,45%
Exploits		82,12%	89,23%	85,52%
Backdoor		99,77%	98,61%	99,18%
Fuzzers		81,52%	86,80%	84,06%
MITM		77,41%	32,65%	45,92%
Ransomware		97,34%	94,20%	95,73%
Generic		69,39%	42,86%	52,92%
Theft		93,59%	93,03%	93,30%
Shellcode		89,37%	78,10%	82,79%
Analysis		10,67%	3,64%	5,36%
Worms		46,33%	40,00%	41,98%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
1028,1170	74,2497	99,40%	79,67%	99,39%

Tabela 35 – Resultados dos experimentos da abordagem DNN com conjunto de dados NF-UQ-NIDSv2.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,24%	99,44%	99,34%
DoS		98,38%	99,28%	98,83%
DDoS		99,26%	98,62%	98,94%
Reconnaissance		98,42%	95,06%	96,71%
XSS		92,36%	97,11%	94,67%
Password		91,86%	92,31%	92,06%
Scanning		93,41%	95,20%	94,29%
Injection		89,03%	76,67%	82,38%
Bot		99,95%	99,93%	99,94%
Brute Force		99,79%	97,48%	98,62%
Infilteration		95,31%	26,09%	40,93%
Exploits		78,44%	73,74%	75,99%
Backdoor		99,96%	95,83%	97,85%
Fuzzers		63,45%	73,81%	67,65%
MITM		78,43%	22,83%	34,72%
Ransomware		86,08%	77,20%	81,15%
Generic		44,52%	7,27%	15,42%
Theft		87,83%	72,12%	78,80%
Shellcode		54,66%	63,81%	55,91%
Analysis		0,00%	0,00%	0,00%
Worms		0,00%	0,00%	0,00%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
11611,8942	53,7094	98,27%	69,70%	98,26%

Tabela 36 – Resultados dos experimentos da abordagem KNN com conjunto de dados NF-UQ-NIDSv2.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,06%	99,03%	99,04%
DoS		99,10%	99,28%	99,19%
DDoS		99,32%	99,17%	99,25%
Reconnaissance		97,10%	97,21%	97,16%
XSS		92,97%	93,21%	93,09%
Password		93,40%	93,67%	93,54%
Scanning		94,02%	94,09%	94,06%
Injection		85,43%	84,56%	84,99%
Bot		100,00%	100,00%	100,00%
BruteForce		97,09%	97,75%	97,42%
Infilteration		38,37%	33,42%	35,72%
Exploits		77,79%	76,53%	77,15%
Backdoor		98,33%	97,68%	98,01%
Fuzzers		70,50%	77,19%	73,69%
MITM		32,48%	30,80%	31,61%
Ransomware		93,05%	90,40%	91,68%
Generic		45,70%	32,47%	37,89%
Theft		83,65%	88,18%	85,75%
Shellcode		67,72%	68,57%	67,53%
Analysis		10,48%	14,55%	12,14%
Worms		63,00%	55,00%	51,57%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
69,2065	13542,5296	98,29%	77,27%	98,27%

Tabela 37 – Resultados dos experimentos da abordagem proposta com conjunto de dados NF-UQ-NIDSv2.

Classes		**Precisão**	***Recall***	***F1-Score***
Benigno		99,46%	99,69%	99,58%
DoS		99,81%	99,07%	99,44%
DDoS		99,84%	99,62%	99,73%
Reconnaissance		99,73%	99,56%	99,64%
XSS		97,11%	97,38%	97,24%
Password		96,55%	96,65%	96,60%
Scanning		96,90%	98,75%	97,82%
Injection		90,33%	91,28%	90,81%
Bot		100,00%	100,00%	100,00%
Brute Force		99,60%	97,73%	98,66%
Infilteration		97,21%	28,58%	44,16%
Exploits		76,31%	92,80%	83,74%
Backdoor		99,22%	98,69%	98,95%
Fuzzers		73,41%	93,81%	82,35%
MITM		3,75%	78,94%	7,16%
Ransomware		94,71%	96,40%	95,54%
Generic		49,22%	48,05%	48,44%
Theft		86,80%	93,64%	90,09%
Shellcode		77,43%	81,90%	79,30%
Analysis		14,32%	49,09%	21,60%
Worms		32,03%	55,00%	39,07%
Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
15000,1419	262,9098	99,05%	85,55%	99,31%

APÊNDICE B – REVISÃO SISTEMÁTICA DA LITERATURA

É importante saber qual o atual estado da arte em detecção e prevenção de intrusão em *Fog Computing* para proteger rede de sensores da *Internet of Things* (IoT). Conhecer a quais ataques estes ambientes estão vulneráveis permite o desenvolvimento de novos mecanismos para tornar este ambiente seguro. Para isso uma revisão da literatura é necessária. Neste trabalho foi realizada uma Revisão Sistemática da Literatura (RSL), pois ela sintetiza o trabalho existente de maneira justa. Elas são realizadas de acordo com uma estratégia de pesquisa predefinida. A estratégia de pesquisa deve permitir que a integridade da pesquisa seja avaliada.

B.1 JUSTIFICATIVA DA NECESSIDADE

Antes de realizar um procedimento de revisão sistemática é preciso assegurar que o mesmo é necessário, ou seja, verificar que não existe uma revisão similar e de alta qualidade na literatura. No entanto, não há um procedimento definido para implementar a pesquisa para identificar a necessidade de realizar um processo revisão sistemática (Kitchenham; Charters, 2007). Desse modo, realizamos várias pesquisas baseadas na metodologia descrita por Oriol, Marco e Franch (2014) para tornar esse processo sistemático também.

Com objetivo de aumentar o número de resultados, consideramos não só estudos sistemáticos mas também outros tipos de revisões da literatura, independentemente da metologia utilizada para desenvolvê-los. Além disso seguimos um procedimento fiel ao da busca principal deste mapeamento. Desse modo, seguimos um protocolo de pesquisa para identificar outras revisões, esse protocolo é baseado no protocolo definido na próxima seção. Em resumo, realizamos buscas nas mesmas quatro base de dados usando as mesmas palavras-chave com a adição de termos referentes a estudos de revisão: *"state of the art"*, *"RSL"*, *"survey"*, *"review"* and *"systematic mapping"*. Desse modo, a *string* de busca ficou da seguinte forma: *("Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR ("Fog Computing" OR Fog)) AND ("SLR" OR "survey" OR "systematic mapping" OR "state of the art" OR "review")*. Realizamos a busca considerando o título, *abstract* e *keywords* dos artigos.

Com esta busca encontramos 15 artigos que atenderam os critérios, eles são apresentados nas Tabelas 38 e 39. Nesta tabela, a coluna **IoT** indica que o trabalho está inserido no contexto de *Internet of Things*. O campo **Fog** atesta que o trabalho considera o ambiente fog. Os trabalhos marcados na coluna **IDS** buscaram realizar uma revisão considerando métodos e abordagens de detecção de intrusão e a os marcados na coluna **CM** fizeram um levantamento dos mecanismos de contramedidas existentes nestes ambientes.

Zarpelão et al. (2017) apresentou uma pesquisa sobre os esforços de pesquisa de IDS para IoT para identificar tendências principais, questões em aberto e possibilidades para pesquisas futuras. Além disso, classificaram as abordagens propostas na literatura de acordo com método de detecção, estratégia de implantação, ameaça à segurança e estratégia de validação. A pesquisa fornece uma revisão da literatura abrangente sobre detecção de intrusão em IoT, no

entanto, faltam alguns aspectos críticos essenciais para uma RSL. Os autores discutem a necessidade de pesquisas sobre abordagens pós-detecção em sistemas IoT, assim, buscamos incluir nesta revisão abordagens de contramedidas para construir uma visão sobre o estado da arte neste contexto.

Este trabalho de Khan, Parkinson e Qin (2017) examinou a literatura existente sobre aplicações de computação Fog para identificar falhas de segurança comuns e determinou o impacto desses problemas de segurança e possíveis soluções, fornecendo orientações de segurança futuras relevantes para os responsáveis por projetar, desenvolver e manter sistemas Fog, no entanto, não concentre-se no escopo da detecção de intrusão.

Elrawy, Awad e Hamed (2018) apresentou uma pesquisa abrangente dos mais recentes IDSs projetados para IoT, com foco nos métodos, recursos e mecanismos correspondentes. Além disso, eles fornecem *insights* profundos sobre a arquitetura de IoT, vulnerabilidades de segurança emergentes e sua relação com as camadas da arquitetura de IoT. Este trabalho demonstra que, apesar de estudos anteriores sobre o *design* e implementação de IDSs para o paradigma IoT, o desenvolvimento de IDSs eficientes, confiáveis e robustos para ambientes inteligentes baseados em IoT ainda é crucial. Eles salientam na sua análise que a localização da implantação do IDS é uma questão séria que deve ser considerada ao conceber qualquer IDS, seja ele baseado em rede ou em *host*. O trabalho carece de algumas caracteristicas importantes para uma RSL.

Hajiheidari et al. (2019) realizou categorizações detalhadas sobre método de detecção, tipo de arquitetura e ameaças abordadas em ambientes IoT. Em seguida, são discutidas as vantagens e desvantagens dos mecanismos selecionados, questões em aberto e direções para tendências futuras. O trabalho apresenta uma RSL com processo bem definido. Contudo, não se pretende construir uma visão sobre soluções de detecção baseadas em *Machine Learning* (ML) e *Deep Learning* (DL), que atualmente estão sendo fortemente estudadas. Em vez disso, concentra-se na categorização de estudos com base em assinaturas e anomalias. O ponto positivo é que apresenta uma RSL muito bem documentada, o que inspirou diversas decisões metodológicas a respeito da RSL do nosso trabalho.

Costa et al. (2019) apresentou pesquisas interessantes sobre ML aplicado à IoT e detecção de intrusões para segurança de redes de computadores. Porém, o trabalho carece de alguns detalhes importantes de uma SRL. Foram pesquisados mais de 95 artigos sobre o assunto, abordando diversos temas relacionados a questões de segurança em ambientes IoT. O trabalho apresenta uma lista interessante de conjuntos de dados utilizados para avaliar as abordagens. Porém, a maioria não considera o ambiente IoT em sua construção. Buscamos em nosso trabalho trazer uma nova lista de conjuntos de dados atuais que possam servir de base para futuros pesquisadores, fornecendo indicações de qual conjunto de dados é mais adequado para o contexto de seus respectivos trabalhos. Além disso, o trabalho considerou artigos publicados até 2017, portanto, encontra-se desatualizado em quatro anos.

Aly et al. (2019) forneceu diretrizes para pesquisadores interessados em compreender questões de segurança de IoT por meio de uma extensa descrição de ameaças e desafios de

segurança em diferentes camadas da arquitetura de sistemas de IoT. Além disso, apresentam algumas soluções baseadas em ML para resolver esses problemas de segurança. No entanto, os estudos considerados já estão desatualizados há mais de 3 anos e acreditamos que surgiram novas abordagens promissoras baseadas em ML e DL.

O trabalho de Ferrag et al. (2020) apresenta uma ótima pesquisa sobre DL para detecção de intrusão além de um estudo comparativo com experimentos. No entanto, não é fornecida uma discussão sobre os principais problemas existentes e direções de pesquisas futuras. O trabalho também apresenta 35 conjuntos de dados cibernéticos para diferentes cenários. No entanto, a maioria dos conjuntos de dados não possui características de IoT. Além disso, não abrange abordagens colocadas no contexto da computação em nevoeiro e contramedidas ativas.

Os autores Idrissi, Azizi e Moussaoui (2020) desenvolveu uma revisão interessante sobre DL para detecção de intrusões. Contudo, o trabalho não apresenta vários itens fundamentais da RSL. Além disso, foca apenas na DL, não considerando os demais aspectos presentes em nosso trabalho.

Kaur, Agrawal e Khan (2020) apresentou uma RSL muito bem documentada que inspirou algumas decisões metodológicas da RSL tomadas em nosso trabalho. No entanto, eles se concentraram na discussão de questões gerais de segurança em ambientes de nevoeiro, e não na detecção de intrusões.

Al-Garadi et al. (2020) conduziu uma ótima pesquisa contendo diversos trabalhos considerando ML e DL para segurança em ambientes IoT. O estudo é bastante completo e apresenta discussões sobre diversos aspectos. Os autores destacam que mais pesquisas precisam ser realizadas para explorar e desenvolver estratégias eficazes para implementar DL e ML mais perto dos dispositivos para fornecer segurança de IoT em tempo real, o que de certa forma motiva nosso trabalho. Além disso, realçam que embora as estratégias *ensemble* tenham alcançado um sucesso notável em muitas aplicações, a aplicação na segurança da IoT necessita de mais investigação, particularmente a possibilidade de implementar classificadores homogêneos ou heterogêneos leves num ambiente distribuído para melhorar a precisão e o desempenho de um sistema de segurança da IoT. e resolver desafios relacionados à complexidade computacional.

O trabalho dos autores Tama e Lim (2021) apresenta um RSL extenso e bem documentado focado em métodos conjuntos para detecção de intrusão. O estudo preenche a lacuna na literatura atual sobre um estudo de mapeamento sistemático atualizado, sem mencionar uma extensa avaliação empírica dos avanços recentes nas técnicas de aprendizagem em conjunto aplicadas ao IDS. Porém, o contexto de trabalho é geral e não se concentra em ambientes mais restritos.

Abbasi, Shahraki e Taherkordi (2021) apresentou uma extensa pesquisa sobre métodos DL para detectar anomalias no tráfego de redes de computadores. O trabalho destaca a importância e analisa profundamente as abordagens de ML/DL. Assim, buscamos também trazer esse nível de revisão para o nosso trabalho, considerando o nevoeiro e o ambiente IoT. O trabalho não foca em ambientes IoT e neblina, porém, faz considerações importantes a respeito. Os autores destacam que treinar um algoritmo DL com um grande número de amostras e parâmetros

Tabela 38 – Revisões da literatura relacionadas.

Trabalho	RSL	IoT	Fog	IDS	CM	Observações
Zarpelão et al. (2017)	x	✓	x	✓	x	Revisão da literatura sobre detecção de intrusão em IoT. Alguns aspectos críticos essenciais para um RSL estão ausentes. Os métodos de *Machine Learning* (ML) e *Deep Learning* (DL) não são considerados na revisão.
Khan, Parkinson e Qin (2017)	x	✓	✓	x	x	Fornece uma visão dos principais aspectos de segurança em IoT e computação de nevoeiro. Não tem foco em detecção de intrusão.
Elrawy, Awad e Hamed (2018)	x	✓	x	✓	x	Extensa pesquisa sobre IDSs projetados para o ambiente IoT. Carece de alguns recursos importantes para uma RSL. Soluções baseadas em DL e *Ensemble Learning* (EL) não foram analisadas.
Hajiheidari et al. (2019)	✓	✓	x	✓	x	RSL com um processo bem definido. Não se concentra em encontrar soluções de detecção baseadas em ML e DL. Concentra-se na categorização de estudos com base na assinatura e anomalia.
Costa et al. (2019)	Partly	✓	x	✓	x	Pesquisa sobre ML para detecção de intrusão em IoT. Carece de vários detalhes importantes de uma RSL.
Or-Meir et al. (2019)	x	x	x	x	x	Pesquisa sobre análise de malware. Não tem foco em detecção de intrusão de IoT.
Aly et al. (2019)	✓	✓	x	✓	x	RSL focado estritamente em DL para detecção de intrusão em IoT. Não se concentra no ambiente de computação em nevoeiro e nos mecanismos de contramedidas.
Ferrag et al. (2020)	x	✓	x	✓	x	Extensa pesquisa sobre DL para detecção de intrusão. Não cobre abordagens focadas em computação de nevoeiro e contramedidas ativas.
Idrissi, Azizi e Moussaoui (2020)	✓	✓	x	✓	x	Não tem itens fundamentais de RSL. Foca apenas em DL para detecção de intrusão, não considera os demais aspectos.
Kaur, Agrawal e Khan (2020)	✓	✓	✓	x	x	RSL sobre problemas de segurança em ambientes de nevoeiro. Não se concentra na detecção de intrusão.

Tabela 39 – Revisões da literatura relacionadas.

Trabalho	RSL	IoT	Fog	IDS	CM	Observações
Al-Garadi et al. (2020)	x	✓	x	✓	x	Levantamento contendo diversos trabalhos. Fornece pesquisas genéricas sobre vários aspectos de segurança, mas não se concentra na detecção e mitigação de intrusões.
Tama e Lim (2021)	✓	✓	x	✓	x	Extensa RSL focada em métodos de ensemble para detecção de intrusão em IoT. Não se concentra em ambientes de computação de nevoeiro.
Abbasi, Shahraki e Taherkordi (2021)	x	x	x	✓	x	Extensa pesquisa sobre métodos de DL para detectar anomalias no tráfego. Não se concentra em ambientes de IoT e nevoeiro.
Ahmad et al. (2021)	✓	x	x	✓	x	RSL focada em detecção de intrusão, no entanto, sem considerar sistemas IoT. Todos os artigos pesquisados e avaliados usaram conjuntos de dados não relacionados à IoT.
Aversano et al. (2021)	✓	✓	x	x	x	RSL bem definida com foco em métodos DL para segurança em IoT. Não se concentra em ambientes de computação em nevoeiro e contramedidas ativas.
Esta revisão	✓	✓	✓	✓	✓	**Extensa RSL em relação à detecção e prevenção de intrusão com foco em computação em nevoeiro e ambientes IoT.**

de treinamento requer dispositivos ricos em recursos, algo que os ambientes IoT e nevoeiro não possuem. Com base nisso, aponta como direção futura as abordagens de aprendizagem federada que parecem ser promissoras para superar essas limitações.

Os autores Ahmad et al. (2021) apresentou uma RSL focada em detecção de intrusões com ML e DL. Eles listaram os desafios da pesquisa, como a indisponibilidade de um conjunto de dados sistemático, menor precisão de detecção devido ao conjunto de dados desequilibrado, recursos consumidos por modelos complexos e a necessidade de IDS leves para IoT. No entanto, o trabalho não se concentrou em ambientes IoT e todos os artigos avaliados utilizaram conjuntos de dados não IoT.

Aversano et al. (2021) revisou e analisou sistematicamente o cenário de pesquisa sobre abordagens de DL aplicadas a diferentes cenários de segurança de IoT. As contribuições são classificadas de acordo com diferentes pontos de vista em uma taxonomia coerente e estruturada para identificar lacunas nesta área de pesquisa. Os autores ressaltam que, exceto em alguns casos, a maioria dos conjuntos de dados utilizados são bastante antigos e baseados em outros tipos de tráfego de rede. Portanto, encontrar um conjunto de dados de referência adequado para aplicar DL na segurança de IoT ainda é difícil. O trabalho não se concentra em ambientes de computação em nuvem e contramedidas ativas.

Uma RSL sintetiza o estado da arte de forma justa, pois é realizada de acordo com uma estratégia de pesquisa pré-definida. A estratégia de investigação deve permitir avaliar a integridade do estudo (Kitchenham; Charters, 2007; Kitchenham et al., 2009; Oriol; Marco; Franch, 2014). Contudo, em nossa busca por trabalhos relacionados, encontramos apenas alguns estudos que realizaram uma boa revisão sistemática da literatura. Hajiheidari et al. (2019) e Kaur, Agrawal e Khan (2020) apresentaram RSLs muito bem documentadas, o que inspirou diversas decisões metodológicas em relação à RSL do nosso trabalho. Contudo, conforme discutido anteriormente, apesar das ótimas avaliações, pontos importantes precisam ser abordados. Abbasi, Shahraki e Taherkordi (2021) destacam a necessidade de estudos sobre abordagens de ML e DL no contexto de redes com menor capacidade de recursos. Al-Garadi et al. (2020) também cita essa necessidade de pesquisa, destacando principalmente os classificadores *ensemble*. Abbasi, Shahraki e Taherkordi (2021) apontam como direção futura de pesquisa as abordagens colaborativas, principalmente com aprendizagem federada, no contexto da computação em nevoeiro que parecem ser promissoras para superar as dificuldades de treinamento e reciclagem de abordagens de DL. Este tema específico de pesquisa é recente e não possui uma boa revisão no estado da arte. Os autores Zarpelão et al. (2017) discutem a necessidade de pesquisas sobre abordagens pós-detecção em sistemas IoT. Além de detectar intrusões, é necessário contar com mecanismos capazes de realizar ações ativas a fim de evitar que a intrusão tenha sucesso. Este tópico também é muito importante e precisa de uma revisão para fornecer direções futuras de pesquisas. Por fim, Costa et al. (2019) e Aversano et al. (2021) destaca que são necessárias pesquisas sobre estratégias/conjuntos de dados de validação para abordagens de detecção de IoT. É necessário fornecer uma lista atualizada dos principais conjuntos de dados recentes para este contexto para servir de base para futuros pesquisadores, fornecendo indicações de qual conjunto

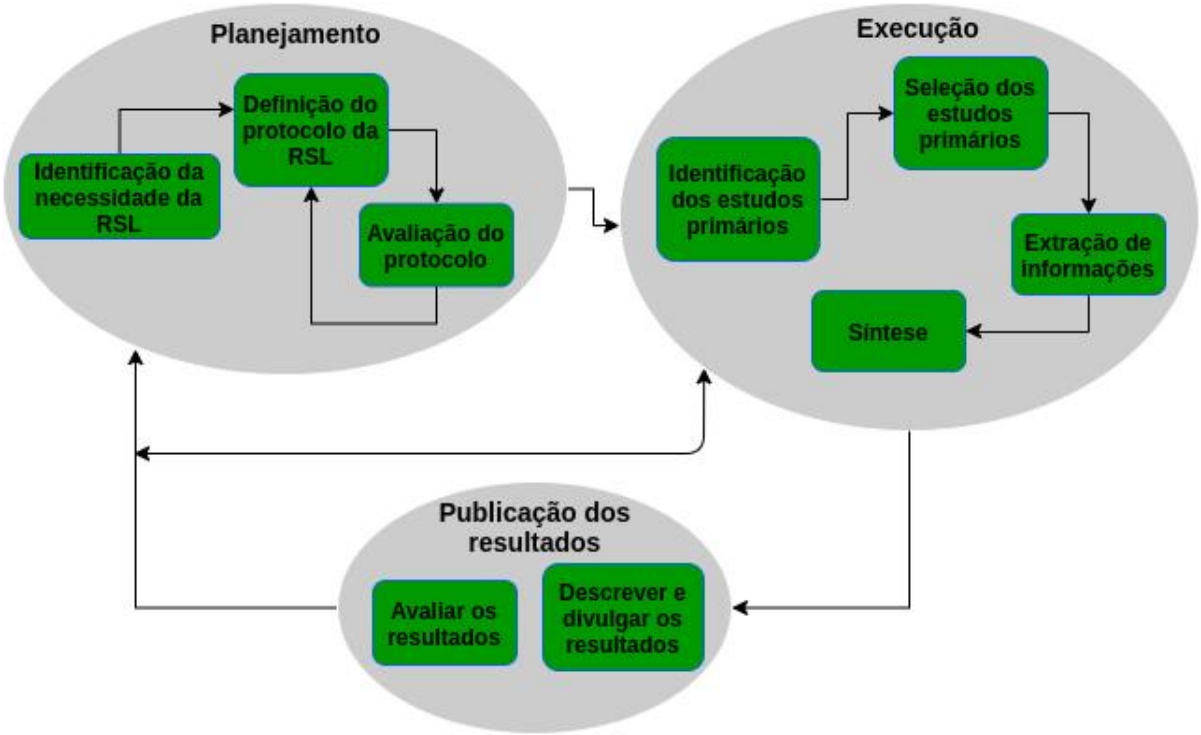
de dados é mais adequado para o contexto de seus respectivos trabalhos.

Esses trabalhos relacionados e os aspectos discutidos acima serviram de base para a definição desta RSL e das questões de pesquisa abordadas, que são apresentadas na próxima seção.

B.2 DEFINIÇÃO DA RSL

O processo de realização do mapeamento sistemático está dividido em três etapas: planejamento e construção do protocolo, execução e sumarização dos resultados da revisão. Estas etapas foram realizadas através das técnicas demonstradas em (Keele et al., 2007; Kitchenham; Brereton, 2013; Petersen; Vakkalanka; Kuzniarz, 2015). A Figura 27 apresenta uma ilustração do processo definido para a RSL (Kitchenham, 2004).

Figura 27 – Ilustração do processo definido para a RSL.



A seguir é apresentado o processo de planejamento da RSL.

B.3 PLANEJAMENTO

Nessa seção será apresentado o protocolo do mapeamento sistemático realizado. Está estratégia bem definida permite a replicação do trabalho e a avaliação da integridade da pesquisa. Desse modo, a seguir são apresentados os objetivos, peguntas de pesquisa e o processo definido para a seleção de trabalhos do estado da arte.

B.3.1 Objetivos e questões de pesquisa

O objetivo principal da RSL é identificar, classificar e comparar as abordagens atuais de detecção e prevenção de intrusão baseadas *fog computing* visando proteger ambientes IoT. Com base neste objetivo e na análise de questões abertas elencadas por outras revisões do estado da arte, foram definidas as seguintes **Q**uestões de **PE**squisa da **R**evisão **S**istemática (QPERS) apresentadas na Tabela 40.

Tabela 40 – Questões de pesquisa definidas para mapeamento.

	Questões de Pesquisas	**Motivações**
QPERS1	Que tipos de dados são analisados para detectar intrusões?	Os tipos de dados analisados é um importante ponto de uma abordagem de detecção.
QPERS2	Onde as soluções de detecção de intrusão são implantadas?	A localização da implantação do IDS é uma questão importante que deve ser considerada ao projetar qualquer tipo de IDS no contexto IoT-Fog-Cloud, seja um NIDS ou um HIDS (Elrawy; Awad; Hamed, 2018). Procuramos revisar as principais abordagens de detecção implantadas ao longo da IoT-Fog-Cloud.
QPERS3	Quais são as categorias de métodos de detecção presentes no estado da arte?	Este ponto é importante para entender que tipo de métodos de detecção são mais comumente empregados neste cenário.
QPERS4	As abordagens focam em identificação (binária) ou classificação (multi-classe)?	Este ponto é importante para entender que tipo de detecção é mais comumente realizada neste cenário.
QPERS5	Quais técnicas de aprendizado de máquina, aprendizado profundo e aprendizado *ensemble* são usadas na detecção de intrusões de última geração?	Abbasi, Shahraki e Taherkordi (2021) apresentou uma extensa pesquisa sobre métodos DL para detectar anomalias no tráfego de redes de computadores. O trabalho destaca a necessidade de estudar abordagens de ML e DL em redes com menor capacidade de recursos, como o nevoeiro e IoT. Al-Garadi et al. (2020) destaca que a aplicação de ML e DL em IoT precisa de mais investigação, particularmente a possibilidade de implementar classificadores leves de conjuntos homogêneos ou heterogêneos para melhorar a precisão e o desempenho. Procuramos trazer esse nível de revisão ao nosso trabalho, revisando as soluções de detecção existentes baseadas em ML, DL e *Ensemble Learning* no contexto da computação em nevoeiro e do ambiente IoT.
QPERS6	Quais são as estratégias de colaboração empregadas nas abordagens de IDS distribuídos?	Abbasi, Shahraki e Taherkordi (2021) apontam que a Aprendizagem Federada parece ser promissora para contornar as limitações de treinamento das abordagens de detecção. Desta forma, buscamos compreender o cenário atual das abordagens distribuídas e colaborativas, focando principalmente nas estratégias utilizadas para treinar modelos de detecção.
QPERS7	Quais são as principais abordagens para executar contramedidas no contexto da IoT e da computação em neblina?	Os autores (Zarpelão et al., 2017) discutem a necessidade de pesquisas sobre abordagens pós-detecção em sistemas IoT, assim, buscamos contemplar nesta revisão abordagens de contramedidas para construir uma visão sobre o estado da arte neste contexto.
QPERS8	Quais estratégias de avaliação são usadas para validar abordagens de detecção?	A grande maioria dos conjuntos de dados utilizados são antigos e baseados em outros tipos de tráfego de rede, é difícil encontrar um conjunto de dados de referência adequado para aplicar DL na segurança de IoT (Costa et al., 2019; Aversano et al., 2021). Desta forma, buscamos em nosso trabalho fazer um levantamento das estratégias de validação utilizadas no estado da arte e trazer uma nova lista atualizada de conjuntos de dados atuais que possam servir de base para futuros pesquisadores, fornecendo indicações de qual conjunto de dados é mais adequado. adequados ao contexto de suas respectivas obras.

Além disso, foram definidas algumas **Q**uestões de **PU**blicação da **R**evisão **S**istemática (QPURS). As QPURS são importantes para prover uma visão do estado da arte da área de pesquisa, identificando os principais locais de publicação, pesquisadores mais ativos e instituições que pesquisam sobre o assunto. As QPURS definidas são apresentadas na Tabela 41.

Tabela 41 – Questões de publicação definidas para a revisão.

	Questões de publicação
QPURS1	Quais os principais locais de publicação?
QPURS2	Quantos trabalhos foram publicados por ano?

B.3.2 Fontes de pesquisa

Nesta revisão sistemática foram pesquisados trabalhos publicados em língua inglesa, no período de 2010 a 2022, disponíveis *on-line*. A busca automática foi realizada em 4 diferentes bases de dados eletrônicas, elas são apresentadas na Tabela 42. A escolha dessas bases de pesquisa foi baseada nos seguintes critérios:

- Cobertura: Avaliar o número de *proceedings* e *journals* e o tipo de áreas abrangidas;
- Atualização de conteúdo: A base de dados deve ser atualizada regularmente com novas publicações;
- Disponibilidade: O texto completo deve estar disponível;
- Facilidade para exportação dos resultados: É importante que a base de dados possua forma automatizada para exportação dos resultados;
- Qualidade dos resultados: Precisão dos resultados retornados pelas bases;
- Usabilidade: A máquina de busca deve ser fácil de entender e operar.

Tabela 42 – Bases de dados pesquisadas.

Base de dados	**URL**
ACM Digital Library	https://dl.acm.org/
ScienceDirect	https://www.sciencedirect.com/
IEEE Xplore	https://ieeexplore.ieee.org/Xplore/home.jsp
Springer Link	https://link.springer.com/
Hindawi	https://www.hindawi.com/
MDPI	https://www.mdpi.com/
Wiley	https://onlinelibrary.wiley.com/

B.3.3 String de Busca

Na Tabela 43 são apresentados os resultados obtidos com as pesquisas nas bases supracitadas aplicando determinadas palavras-chave. As palavras-chave da categoria 1 geraram uma grande quantidade de resultados, que passam a reduzir-se a medida que a pesquisa é refinada.

Tabela 43 – Refinamento dos termos utilizados na pesquisa.

	Palavras-chave	**Resultados**
1	“Intrusion Detection”	15288
1	“Intrusion Prevention”	7250
1	“Internet of Things”	59730
1	“Fog Computing”	3040
1	IoT	43724
1	Fog	27388
2	“Intrusion Detection” AND (“Internet of Things” OR IoT)	1434
2	“Intrusion Detection” AND (“Fog Computing” OR Fog)	430
2	“Intrusion Prevention” AND (“Internet of Things” OR IoT)	253
2	“Intrusion Prevention” AND (“Fog Computing” OR Fog)	35
3	“Intrusion Detection” AND ((“Internet of Things” OR IoT) OR (“Fog Computing” OR Fog))	2629
3	“Intrusion Prevention” AND ((“Internet of Things” OR IoT) OR (“Fog Computing” OR Fog))	261
4	“Intrusion Detection” OR “Intrusion Prevention”) AND ((“Internet of Things” OR IoT) OR (“Fog Computing” OR Fog))	2712

A partir do refinamento realizado nas palavras-chave foi escolhida para a pesquisa a Categoria 4 apresentada na Tabela 43. A seguir são definidos os critérios de exclusão e inclusão aplicados nos artigos durante o processo de seleção.

B.3.4 Critérios de inclusão e exclusão

Os artigos pesquisados nas sete bases de dados foram submetidos a alguns critérios de exclusão e inclusão para obtenção de um conjunto de estudos pertencentes ao estado da arte do tema. Os critérios de inclusão e exclusão utilizados neste trabalho são apresentados e justificados na Tabela 44.

Tabela 44 – Critérios de inclusão e exclusão.

Critério	Descrição
Inclusão 1 - Serão incluídos trabalhos que proponham abordagens para detectar e/ou prevenir ataques em *fog computing* e IoT	Queremos construir uma visão clara do estado da arte em detecção e prevenção de intrusões em ambientes de *fog computing* e IoT, portanto, consideramos artigos que estejam diretamente inseridos neste contexto.
Inclusão 2 - Estudos revisados por pares serão incluídos.	Queremos considerar apenas estudos que tenham verificação prévia de qualidade e integridade.
Inclusão 3 - Serão incluídos trabalhos publicados entre 2010 e 2023.	Queremos considerar uma visão atual sobre o tema. Estudos anteriores a 2010 podem não representar corretamente o cenário atual.
Inclusão 4 - Serão incluídos estudos publicados em inglês.	Incluímos apenas artigos em inglês, pois os trabalhos de qualidade geralmente são em inglês.
Exclusão 1 - Serão excluídos trabalhos publicados como artigos curtos ou pôsteres.	Serão excluídos por serem demasiado pequenos e não poderem apresentar uma abordagem completa.
Exclusão 2 - Serão excluídos estudos em forma de revisões.	Os estudos de revisão não apresentam novas abordagens, portanto, são excluídos.
Exclusão 3 - Trabalhos duplicados serão excluídos.	Alguns trabalhos podem ser duplicados porque foram obtidos através de diferentes bases de artigos ou através do processo *snowballing*, que também pode gerar duplicatas. Além disso, serão excluídos trabalhos prolongados sem diferenças significativas.
Exclusão 4 - Serão excluídos trabalhos que não realizem a implementação e validação com o método proposto.	Excluímos trabalhos que não implementaram e avaliaram a abordagem proposta porque suposições não validadas poderiam comprometer as discussões do nosso estudo.

B.3.5 Processo de seleção de estudos

O processo de seleção dos estudos é uma importante etapa da revisão sistemática. Seu objetivo é obter um conjunto de artigos realmente relevante para o mapeamento. A Figura 28 ilustra o processo completo de seleção de estudos empregado neste trabalho.

Inicialmente os estudos são obtidos através de buscas nas 7 bases de dados. A partir disso, são excluídos os trabalhos duplicados e são aplicados os critérios de exclusão definidos na Seção B.3.4. Os artigos selecionados após este processo são incluídos no mapeamento final. No entanto, ainda é realizado um processo de busca de artigos relacionados aos estudos incluídos, esse processo se chama *Snowballing*. São realizadas buscas nas referências dos artigos selecionados (*snowballing backward*) e nos estudos que citam os artigos selecionados (*snowballing forward*). Os artigos obtidos com estas buscas passam pelo processo de exclusão de duplicados e de exclusão de acordo com os critérios definidos na Seção B.3.4. Após esse processo, caso existam novos artigos incluídos, o processo de *snowballing* é reiniciado com estes novos estudos. Caso não exista nenhum novo estudo incluído, o processo de seleção termina e tem-se um conjunto de trabalhos selecionados.

B.3.6 Ameaças à validade do mapeamento

O protocolo de revisão sistemática permite a verificação da metodologia, mas não a verificação de como ela foi realizada. Portanto, sempre haverá ameaças à validade dos resulta-

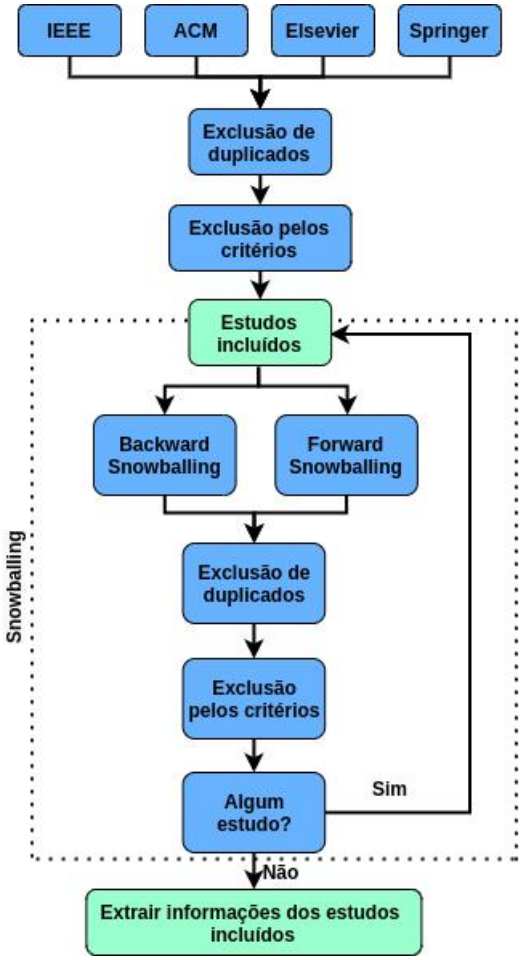


Figura 28 – Descrição do processo *snowballing*.

dos (Kitchenham; Budgen; Brereton, 2015). A seguir, serão identificadas as ameaças presentes nesta revisão e as medidas de minimização adotadas:

1. **Avaliação tendenciosa pelos pesquisadores**: Procuramos minimizar os problemas de viés de avaliação do pesquisador revisando com pesquisadores auxiliares.

2. **Viés de publicação**: Tentamos minimizar os problemas de viés de publicação com o uso de quatro bases de dados. Além disso, apenas artigos publicados em congressos e periódicos foram considerados para garantir a qualidade científica dos trabalhos.

3. **Ameaças à seleção de estudos primários**: O risco de o mapeamento não incluir artigos importantes na área foi minimizado por meio do planejamento da pesquisa de sequências e critérios de inclusão abrangentes, a fim de incluir estudos que abrangem assuntos como detecção, prevenção, IoT e computação em nevoeiro. Além disso, a partir do conjunto de artigos incluídos na busca inicial, foi realizado o processo de *snowballing* com as fases *backward* e *forward* para garantir a inclusão de artigos relacionados não incluídos na busca inicial.

B.4 EXECUÇÃO DA REVISÃO

Inicialmente, em 18 de junho de 2022, os estudos foram pesquisados nas bases de dados ACM, IEEE, *ScienceDirect*, *Springer*, *Hindawi*, *Wiley* e MDPI. Foi utilizada em todas as bases a *string* de busca apresentada na Seção B.3.3, sem nenhuma variação. No entanto, alguns mecanismos de filtros das bases foram utilizados para descartar trabalhos que estavam fora do escopo. As strings, os filtros utilizados em cada base e os resultados retornados são apresentados na Tabela 45. As buscas dos artigos nas bases de dados apresentadas na Tabela 42 foram realizadas considerando o intervalo temporal entre 2010 e 18 de junho de 2022.

Tabela 45 – As strings e filtros usados em cada pesquisa.

	String de busca	**Filtros**	**Resultados**
ACM	"Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR ("Fog Computing" OR Fog))	Publication Date: 2010-2023. Content Type: Research Article + Review Article	706
IEEE	"Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR ("Fog Computing" OR Fog))	Year: 2010-2023. Content Type: Conferences + Journals + Early Access Articles	931
ScienceDirect	"Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR ("Fog Computing" OR Fog))	Years: 2010-2023. Article type: Research Articles + Review Articles	1806
Springer	"Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR ("Fog Computing" OR Fog))	Date Published: 2010-2023. Content Type: Article + Conference Paper	1442
Hindawi	"Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR ("Fog Computing" OR Fog))	Date Published: 2010-2023. Content Type: Research Article + Conference Paper	71
MDPI	"Intrusion Detection" + OR "Intrusion Prevention" + AND "Internet of Things" + AND "Fog Computing"	Date Published: 2010-2023. Advanced Search. Content Type: Article + Review	175
Wiley	"Intrusion Detection" OR "Intrusion Prevention") AND (("Internet of Things" OR IoT) OR ("Fog Computing" OR Fog))	Date Published: 2010-2023. Content Type: Journals	454
Total			5585

A busca na *ACM* foi realizada na caixa de busca inicial, sem utilizar configurações avançadas e o total de artigos retornados foi 706. Na *IEEE* a busca retornou 931 artigos. Ela foi realizada na caixa de busca inicial, sem utilizar configurações avançadas. O campo *"Find articles with these terms"* foi utilizado na busca na *Science Direct*, sem utilizar configurações avançadas. O total de artigos retornados foi 1806. A busca na *Springer* foi realizada somente na caixa de busca inicial, sem utilizar configurações avançadas e retornou um total de 1442. O total de artigos retornados na busca na base de dados *Hindawi* foi 71 e na MDPI foi 175. Por fim, a busca na base de dados *Wiley* retornou 454 trabalhos. Em todas as buscas foram utilizados para cada base os filtros definidos na Tabela 45.

Portanto, inicialmente as buscas nas 7 bases de dados apresentadas na Tabela 42 retornaram um total de 5585 artigos (Estágio 1), conforme pode ser observado na Figura 29. A partir disso, foi realizada a exclusão de artigos duplicados e de estudos que não atendiam os critérios. Desse modo, com base no processo supracitado, foram incluídos um total de 88 estudos aceitos, conforme pode ser observado na Figura 29.

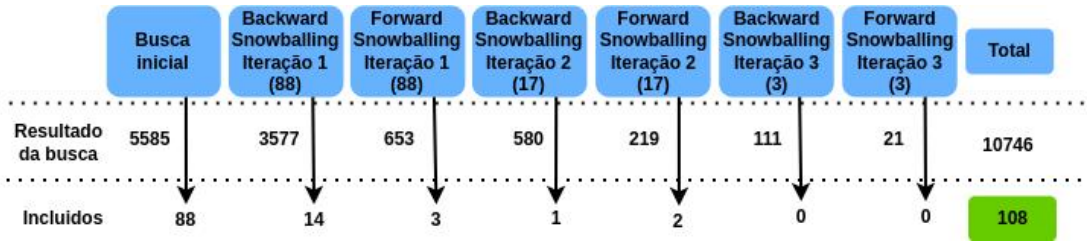


Figura 29 – Execução do processo de seleção de busca inicial.

A partir dos 88 estudos selecionados foi realizado o processo de *snowballing*. Conforme pode ser observado na Figura 29 foram realizadas 3 iterações do processo. Na primeira iteração foram buscadas as referências (*snowballing backward*) dos 88 estudos, resultado em 3577 trabalhos. Dos quais, 14 estudos foram incluídos. Ainda na primeira iteração foram buscadas as citações feitas aos 88 trabalhos iniciais (*snowballing forward*), resultando em 653 estudos. Destes, 3 trabalhos foram incluídos.

Na segunda iteração, foi realizado o mesmo processo para os 14 novos estudos incluídos, resultando em 580 artigos encontrados na fase *snowballing backward* e 219 na *snowballing forward*. A fase *forward* teve 2 artigos incluídos e a fase *backward* 1 artigo. Desse modo, uma terceira iteração foi realizada para os 3 novos artigos. As buscas resultaram em 21 artigos encontrados na fase *forward* e 111 na *backward*. Nenhum novo artigo foi incluído e o processo geral de seleção foi encerrado resultando em um conjunto de 108 estudos selecionados ao longo da execução.

Os 108 artigos resultantes do processo de execução do protocolo e incluídos no mapeamento foram inspecionados de modo a obter as informações necessárias para responder as perguntas de pesquisa definidas na Seção B.3.1. Os resultados obtidos e as respostas as perguntas de pesquisa são apresentados a seguir.

B.5 RESULTADOS

Após a execução do protocolo sistemático de revisão, apresentado na seção anterior, obteve-se um conjunto de 108 artigos que propõem abordagens de detecção e prevenção de intrusão em ambiente de *fog computing* e *Internet of Things* (IoT). A seguir são apresentados os artigos selecionados e as informações extraídas são utilizadas para responder as perguntas de pesquisa definidas na Seção B.3.1.

B.5.1 Trabalhos selecionados

Nesta seção são apresentados os trabalhos selecionados que propõem soluções de detecção e prevenção de intrusões no contexto de IoT e *fog computing*. Na Tabela B.5.1 são listados os 108 artigos do estado da arte, incluídos no mapeamento. Estes trabalhos propuseram abordagens para detecção e prevenção de intrusão em ambientes de *fog computing* e IoT. A Tabela B.5.1 apresenta a referência do trabalho e o número identificador do estudo que será considerado para exibição nos gráficos e tabelas.

Tabela 46 – Abordagens de detecção e prevenção de intrusão em ambientes de IoT e *fog computing*.

Trabalhos	**Id**
Hosseinpour et al. (2016)	1
Yaseen et al. (2016)	2
Kim et al. (2016)	3
Yaseen et al. (2017)	4
Yaseen et al. (2017)	5
Pan, Pacheco e Hariri (2017)	6
Sandhu, Sohal e Sood (2017)	7
Le, Kim e Kim (2017)	8
Aliyu, Sheltami e Shakshuki (2018)	9
An et al. (2018c)	10
An et al. (2018a)	11
An et al. (2018b)	12
Diro e Chilamkurti (2018a)	13
Diro e Chilamkurti (2018c)	14
Diro e Chilamkurti (2018b)	15
Peng et al. (2018)	16
Prabavathy, Sundarakantham e Shalinie (2018)	17
Rathore e Park (2018)	18
Shafi et al. (2018)	19
Shen et al. (2018a)	20
Sohal et al. (2018)	21
Tu et al. (2018)	22
Yaseen et al. (2018)	23
Wang et al. (2018)	24
Alrashdi et al. (2019)	25
Arshad et al. (2019)	26
Illy et al. (2019)	27
Khater et al. (2019)	28
Maharaja, Iyer e Ye (2019)	29
Nguyen et al. (2019)	30
Pacheco, Benitez e Félix-Herrán (2019)	31
Potrino, Rango e Santamaria (2019)	32
Potrino, Rango e Fazio (2019)	33

Priyadarshini e Barik (2022)	34
Roopak, Tian e Chambers (2019a)	35
Xu, Qian e Hu (2019)	36
Zhou, Guo e Deng (2019)	37
Moustafa et al. (2019)	38
Le et al. (2019)	39
Abdel-Basset et al. (2020)	40
Albdour, Manaseer e Sharieh (2020)	41
Almiani et al. (2020)	42
NG e Selvakumar (2020)	43
Miranda et al. (2020)	44
Mourad et al. (2020)	45
Pacheco et al. (2020)	46
Protogerou et al. (2020)	47
Rahman et al. (2020)	48
Ravi e Shalinie (2020)	49
Sugi e Ratna (2020)	50
Sadaf e Sultana (2020)	51
Samy, Yu e Zhang (2020)	52
Souza et al. (2020)	53
Zahra e Chishti (2020)	54
Moussa e Alazzawi (2020)	55
Ghazi e Rachid (2020)	56
Kumar, Gupta e Tripathi (2020)	57
Kumar et al. (2020)	58
Thi-Nga et al. (2020)	59
Zhang, Wei e Wang (2020)	60
Du et al. (2020)	61
Farukee et al. (2020)	62
Lawal, Shaikh e Hassan (2020)	63
Pirozmand et al. (2020)	64
Krishnan, Duttagupta e Achuthan (2020)	65
Mourad et al. (2021)	66
Yahyaoui et al. (2021)	67
Abdel-Basset et al. (2021)	68
Kumar, Gupta e Tripathi (2021b)	69
Kumar, Tripathi e Gupta (2021)	70
Moustafa et al. (2021)	71
Lawal, Shaikh e Hassan (2021)	72
Gavel, Raghuvanshi e Tiwari (2021)	73
Manimurugan (2021)	74
Chuang, Yang e Wang (2021)	75
Kumar et al. (2021)	76
Omar, Goyal e Varadarajan (2021)	77
Ponnusamy e Sharma (2021)	78
Araujo-Filho et al. (2021)	79
Alhowaide, Alsmadi e Tang (2021)	80
Onah et al. (2021)	81

Reddy et al. (2021)	82
Al-Khafajiy et al. (2021)	83
Hameed et al. (2021)	84
Khater et al. (2021)	85
Arbex et al. (2021)	86
Li, Au e Wang (2021)	87
Sahar, Mishra e Kalam (2021)	88
Zhang et al. (2021)	89
Ullah et al. (2021)	90
Diwan et al. (2021)	91
Kumar, Gupta e Tripathi (2021a)	92
Lalouani e Younis (2021)	93
Sahi, Soni e Auluck (2021)	94
Azarkasb, Kashi e Khasteh (2021)	95
Paranjothi e Atiquzzaman (2021)	96
Kumar e Tripathi (2021)	97
Rangisetti, Dwivedi e Singh (2021)	98
Kalnoor e Gowrishankar (2021)	99
Pan et al. (2021)	100
Hosseini e Sardo (2022)	101
Souza, Westphall e Machado (2022)	102
Razaque et al. (2022)	103
Kumar et al. (2022)	104
Rey et al. (2022)	105
Aliyu et al. (2022)	106
Labiod, Korba e Ghoualmi (2022)	107
Zahra e Chishti (2022)	108

B.5.2 QPERS1 - Que tipos de dados são analisados para detectar intrusões?

As abordagens de detecção podem analisar dados de várias fontes. Esses dados geralmente estão relacionados à forma como a abordagem é implementada. Existem duas categorias principais de implantação relacionadas ao foco da captura de informações, os sistemas de detecção baseados em *host* (*Host-Based Intrusion Detection System - HIDS)*, que buscam analisar as informações capturadas do *host* onde são implantados, e os sistemas de detecção de intrusão baseados em rede (Network-Based Intrusion Detection System - NIDS), que analisam o tráfego capturado da rede monitorada (Zarpelão et al., 2017). A Tabela 47 apresenta uma relação entre os trabalhos selecionados e o tipo de fonte de captura de informação empregada.

Tabela 47 – Categorias dos tipos de dados analisados.

Dados de rede	**Dados de *host***
[1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [12], [13], [14], [15], [16], [17], [19], [21], [22], [23], [25], [27], [29], [30], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [61], [62], [63], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [87], [88], [89], [90], [91], [92], [93], [95], [96], [99], [100], [101], [102], [103], [104], [105], [106], [107]	[26], [28], [31], [32], [33], [46], [84], [85], [94]

No estado da arte foram encontrados 87 trabalhos baseados na análise de informações da rede (NIDS), portanto, a maioria, correspondendo a aproximadamente 80% dos estudos incluídos no mapeamento. As abordagens NIDS são implantadas em um dispositivo capaz de capturar o tráfego da rede a ser monitorado. Esta estratégia possibilita a detecção de ataques externos e é mais independente de plataforma (Mukherjee; Heberlein; Levitt, 1994). As abordagens NIDS monitoram a rede IoT composta por vários dispositivos. O tráfego analisado é capturado do tráfego da rede em modo promíscuo e as informações extraídas do cabeçalho dos pacotes de rede capturados são analisadas. Os sensores também podem ser usados para capturar informações de maneira distribuída em vários pontos da rede. Prabavathy, Sundarakantham e Shalinie (2018) propôs uma abordagem de detecção distribuída no nevoeiro que analisa as informações capturadas do tráfego da rede IoT e visa atender à natureza de *streaming* do tráfego do aplicativo IoT. Illy et al. (2019) propuseram uma abordagem em que as informações analisadas referem-se ao tráfego da rede IoT vinculada ao nevoeiro onde a abordagem é implementada. Os autores Diro e Chilamkurti (2018a) propuseram abordagens distribuídas no nível de computação de nevoeiro e baseada em aprendizado de máquina para detectar intrusões no ambiente IoT com base em informações capturadas de pacotes que trafegam pela rede IoT (Diro; Chilamkurti, 2018b; Diro; Chilamkurti, 2018c). Kalnoor e Gowrishankar (2021) projetou um sistema inteligente de detecção de intrusões (I-IDS) baseado em computação de nevoeiro que analisa pacotes de sensores para detectar ataques na rede IoT. Outras diversos trabalhos também propuseram abordagens NIDS para estes cenários (Arbex et al., 2021; Razaque et al., 2022; Hosseini; Sardo, 2022). Uma das dificuldades encontradas ao se projetar NIDS é determinar os melhores locais para colocar os sensores de captura de informações.

Apenas 9 abordagens apresentaram análise baseada em *host* (HIDS). No caso dessas abordagens, os mecanismos de monitoramento e análise de eventos usam apenas informações capturadas do próprio dispositivo (Sahi; Soni; Auluck, 2021; Khater et al., 2021). Os eventos podem se originar de *logs* do sistema, dados sobre usuários, serviços e processos (Bace; Mell, 2001). Os autores Arshad et al. (2019) propuseram uma abordagem de detecção baseada em host que realiza detecção de assinatura em nós IoT. Os eventos detectados pelos nós finais são

relatados ao sistema centralizado que opera na camada de computação em nevoeiro. Essa camada correlaciona alertas de vários dispositivos IoT e aplica a detecção baseada em anomalias. No entanto, os detalhes dos algoritmos de detecção usados em cada etapa não são apresentados. Em Khater et al. (2019), um IDS leve baseado em Redes Neurais Artificiais (*Artificial Neural Networks* - ANN) foi proposto para identificar intrusões nos próprios dispositivos da camada do nevoeiro. Para isso, eles analisaram dados de chamadas do sistema dos próprios dispositivos. Pacheco et al. (2020) e Pacheco, Benitez e Félix-Herrán (2019) destacam que os dispositivos do nevoeiro podem ser alvos de invasores, prejudicando o fornecimento de informações importantes aos usuários finais ou a realização de ações automatizadas precisas. Assim, eles propõem uma abordagem baseada em ANN para implementar um IDS adaptativo capaz de detectar quando um nó do nevoeiro foi comprometido e realizar as ações necessárias para garantir a disponibilidade da comunicação. A abordagem de detecção analisa as informações do próprio dispositivo do nevoeiro para detectar comportamento anômalo. Entre as informações estão dados sobre memória, CPU, *buffers*, processo, *ethernet*, etc. (Pacheco; Benitez; Félix-Herrán, 2019). Em Potrino, Rango e Santamaria (2019) e Potrino, Rango e Fazio (2019), foram propostas abordagens de detecção e mitigação para um ambiente com comunicação através do protocolo *Message Queue Telemetry Transport Transport* (MQTT). O *broker* é inserido no nó do nevoeiro, que possui melhor fonte de energia e maior capacidade de processamento e armazenamento. O sistema de segurança é baseado em um HIDS no nevoeiro para mitigar os ataques *Denial Of Service* (DoS), aceitando pacotes com frequência limitada e monitorando a plenitude do *buffer* para garantir as prioridades do tópico. As abordagens baseadas em *host* permitem a independência da rede e a detecção de ataques internos, mas possui como desvantagem a dificuldade em tratar ataques da rede e ao próprio IDS (Bace; Mell, 2001). As soluções baseadas em *host* empregadas nos próprios nós IoT podem sofrer com as restrições de memória, por outro lado, as implementadas em dispositivos do nevoeiro permitem a detecção de ataques contra o próprio dispositivo, mas podem ter dificuldade em lidar com ataques à rede e dispositivos IoT.

B.5.3 QPERS2 - Onde as soluções de detecção de intrusão são implantadas?

O local de implantação da solução de detecção também é um problema sério que deve ser considerado ao projetar a abordagem, seja baseada em rede ou baseada em *host*. As restrições de recursos, geralmente existentes nos dispositivos inseridos no contexto de aplicações IoT, dificultam a implantação de abordagens robustas de detecção nos próprios dispositivos. No contexto de ambientes IoT, as abordagens de detecção e prevenção de intrusões podem ser implantadas em 3 níveis de locais: nos dispositivos IoT, nos dispositivos do nevoeiro e na nuvem. Nenhum dos estudos considerados neste trabalho propôs a implantação da abordagem completa de detecção nos próprios dispositivos IoT. No entanto, alguns trabalhos propuseram abordagens implementadas parcialmente em dispositivos IoT (Arshad et al., 2019), através do emprego de técnicas mais leves baseadas em assinatura. Isso se deve principalmente às restrições de recursos desse nível, o que torna impossível implementar soluções complexas nos próprios dis-

positivos. Outras abordagens propuseram suas soluções para operar inteiramente na camada do nevoeiro (Almiani et al., 2020; Sadaf; Sultana, 2020; Kumar; Gupta; Tripathi, 2021a). Além disso, alguns trabalhos também delegaram parte da análise para a camada de computação em nuvem (Nguyen et al., 2019; Illy et al., 2019). Nenhuma abordagem foi implantada totalmente na nuvem, que apesar de possuir grande capacidade de recursos de processamento, possui o problema da latência ocasionado pela distância entre os dispositivos IoT e os *datacenters* da computação em nuvem.

A computação de nevoeiro é uma das alternativas mais promissoras para implementar abordagens para monitorar a rede IoT, devido a proximidade dos dispositivos finais, a capacidade de detectar ataques externos e a independência da plataforma (Mukherjee; Heberlein; Levitt, 1994). É necessário destacar que a maioria dos trabalhos apresentou abordagens para o monitoramento da rede IoT com a implementação no dispositivo do nevoeiro e análise das informações capturadas dos pacotes da rede (Shafi et al., 2018; Kumar; Tripathi; Gupta, 2021; Sahi; Soni; Auluck, 2021; Hosseini; Sardo, 2022). Os autores Diro e Chilamkurti (2018a) propuseram abordagens distribuídas no nível do nevoeiro (Diro; Chilamkurti, 2018b; Diro; Chilamkurti, 2018c). A estrutura possui dois níveis de nevoeiro. Ele usa um dispositivo do nevoeiro como um mestre responsável por treinar e colocar o modelo de detecção nos outros nós. Sugi e Ratna (2020), também propuseram uma abordagem com mecanismo de treinamento distribuído em nós de nevoeiro. A abordagem proposta por Almiani et al. (2020) também foi implantada nos dispositivos de nevoeiro próximo aos dispositivos da rede IoT e monitora o tráfego externo que tem como destino dispositivos que estão dentro da rede IoT.

Além disso, algumas abordagens interessantes dividiram as tarefas de detecção e agregação em mais de um nível da arquitetura (Nguyen et al., 2019; Illy et al., 2019; Rey et al., 2022). Arshad et al. (2019) propuseram uma solução que conta com a colaboração entre o nevoeiro e sensores com recursos limitados para a detecção eficaz e oportuna de intrusos. O módulo de detecção de nível de sensor é um módulo leve, de modo a respeitar os recursos relativamente limitados disponíveis em nodos de sensores individuais. O sistema centralizado que opera na camada da computação em nevoeiro realiza a correlação de alertas de vários sensores e realiza análise nesses eventos correlacionados. Prabavathy, Sundarakantham e Shalinie (2018) propuseram uma abordagem de detecção em que a inteligência em nuvem centralizada existente na detecção de ataques é distribuída para os nós do nevoeiro locais para detectar o ataque em uma taxa mais rápida para o aplicativo IoT. As invasões detectadas em nós do nevoeiro são enviadas ao servidor em nuvem para gerar uma visão global do estado de segurança atual do ambiente IoT. A abordagem Search (Nguyen et al., 2019) apresenta uma arquitetura de detecção de intrusão colaborativa e inteligente para redes IoT que analisa informações de tráfego de rede. Ele compreende um arranjo hierárquico de três camadas de nós IDS inteligentes trabalhando de forma colaborativa para detectar anomalias e formular políticas em dispositivos de *gateway* IoT baseados em SDN para interromper o tráfego malicioso o mais rápido possível. Os elementos Edge-IDS atuam no nível de computação de ponta. Eles funcionam como mecanismos de detecção de intrusão que residem em um *gateway* IoT baseado em Redes Definidas

por Software (Software Defined Networks - SDN). As principais tarefas são coletar e extrair periodicamente informações estatísticas sobre o fluxo de tráfego de uma rede IoT local para derivar um conjunto de recursos desejados, que são alimentados em um mecanismo baseado em aprendizado de máquina. Além disso, o Edge-IDS encaminha ativamente os dados para a camada superior, Fog-IDS, para padrões de tráfego desconhecidos ou tráfego de entrada de alto volume. Da mesma forma, padrões de tráfego desconhecidos são enviados a um Cloud-IDS para análises mais complexas. Ghazi e Rachid (2020) propuseram uma abordagem detecção de intrusão híbrida instalada na nuvem alimentando outra abordagem de detecção de intrusão online e em tempo real no nevoeiro para monitorar a comunicação e detectar ataques antes que se espalhem pela rede, como no caso do*botnet*Mirai. Zahra e Chishti (2022) também propuseram uma arquitetura cobrindo a camada de dispositivo de borda, a camada de neblina e a camada de nuvem. Os dispositivos em cada camada executam tarefas de segurança específicas em sincronia para obter alta detecção de intrusos em tempo real. Alguns trabalhos propuseram abordagens de detecção em duas etapas ao longo de uma estrutura hierárquica *Fog-Cloud*. Essa estratégia de arquitetura foi inicialmente publicada em Souza et al. (2020), onde é empregada uma análise binária na camada de computação em nevoeiro e uma análise multiclasse na camada de nuvem. Os dispositivos da computação em nevoeiro estão mais próximos da rede IoT e conseguem fornecer respostas em um ritmo mais rápido. No entanto, não possuem recursos ilimitados como a computação em nuvem. Desse modo, apesar das dificuldades relacionadas à distância, a nuvem também pode ser uma importante aliada na tarefa de detecção, devido à grande capacidade de recursos. O objetivo é aproveitar as melhores características de cada camada. Desse modo, utiliza-se o nevoeiro para fornecer análises e respostas rápidas, principalmente para os eventos benignos, e a nuvem para realizar uma análise mais robusta nos eventos detectados como intrusivos. Outros trabalhos também utilizaram essa estratégia e realizaram melhorias (Qaddoura et al., 2021b; Labiod; Korba; Ghoualmi, 2022; Dat-Thinh; Xuan-Ninh; Kim-Hung, 2022).

O tempo e custo de processamento de modelos robustos de aprendizado de máquina pode ser alto, especialmente quando se considera estratégias que utilizam diversos métodos de detecção. Isso pode ser um problema para a implementação de uma abordagem robusta nos dispositivos IoT e até mesmo na camada do nevoeiro. Uma análise multiclasse robusta e lenta realizada no nevoeiro pode sobrecarregar o dispositivo e atrasar o fluxo da rede. É possível contornar essa limitação executando o método na camada de nuvem que possui maior capacidade de recursos, no entanto, essa estratégia possui a desvantagem de latência causada pela distância da rede IoT e dos data centers. A implantação no nível do nevoeiro evita o problema da latência e permite implantar uma estratégia distribuída nos dispositivos, buscando reduzir a quantidade de tráfego monitorado e aumentar a capacidade de processamento.

Portanto, cada estratégia de implantação possui vantagens e desvantagens. É necessário investigar novas abordagens de detecção de intrusão capazes de gerenciar e otimizar o processo de análise entre as várias camadas do ambiente de IoT. Ou seja, otimizar o uso dos recursos disponíveis em dispositivos IoT, nevoeiro e nuvem.

B.5.4 QPERS3 - Quais são as categorias de métodos de detecção presentes no estado da arte?

Nesta seção são discutidas as categorias de abordagens para detecção de intrusões encontradas no estado da arte, em relação ao tipo de análise empregada. Elas podem ser agrupadas em análise por assinatura ou análise por comportamento, está ultima também conhecida como análise por anomalia.

Na detecção por assinatura, as ações monitoradas são comparadas com padrões intrusivos predefinidos, esses padrões são chamados de assinaturas. Este método é utilizado pela maioria dos sistemas de antivírus comerciais (Bace; Mell, 2001). No entanto, possui a limitação de apenas detectar esses ataques conhecidos (Northcutt et al., 2001). Wang et al. (2018) propôs uma abordagem que aplica o algoritmo de impressão digital *Rabin* em assinaturas do IDS Snort [1] e em pacotes de rede para proteger a privacidade e detectar intrusões em ambientes baseados em nevoeiro. Os autores Sandhu, Sohal e Sood (2017) e Sohal et al. (2018), também utilizaram o IDS Snort, ele é empregado com um modelo de Markov oculto de dois estados para categorizar os dispositivos em diferentes categorias de status. Além disso, um *Virtual Honeypot Device* (VHD) foi projetado para armazenar e manter um repositório de log de todos os dispositivos maliciosos identificados, o que ajuda o sistema a se defender de quaisquer ataques desconhecidos no futuro. Os autores Mourad et al. (2020) também propuseram uma abordagem de detecção baseada em assinatura, mais precisamente no algoritmo Aho-Corasick, que compara chamadas de sistema coletadas em tempo real contra um conjunto de dados predefinido. A abordagem foi proposta no contexto de *Vehicular Edge Computing* (VEC). Embora as soluções baseadas e assinatura permitam uma detecção rápida e reduzam a ocorrência de alarmes falsos, elas possuem a limitação de detectar apenas esses ataques conhecidos. Em Lawal, Shaikh e Hassan (2020), Lawal, Shaikh e Hassan (2021) são propostas estruturas de detecção de intrusão baseadas em assinatura e anomalias, usando computação em nevoeiro para garantir detecção de ataques rápida e precisa, bem como escalabilidade. Primeiro, é realizada uma verificação em relação às assinaturas de ataque detectadas anteriormente. Modelos posteriores de aprendizado de máquina são responsáveis por distinguir entre fluxo de tráfego de rede normal e intrusivo. Ghazi e Rachid (2020) propôs uma abordagem híbrida de detecção de intrusões instalada na nuvem, alimentando outra abordagem de detecção de intrusões on-line e em tempo real no nevoeiro para monitorar a comunicação e detectar ataques antes que eles se espalhem pela rede. Embora as soluções baseadas em assinaturas permitam a detecção rápida e reduzam a ocorrência de alarmes falsos, elas têm a limitação de detectar apenas ataques conhecidos.

[1] https://www.snort.org/

Tabela 48 – Categorias de método de detecção.

Assinatura	Anomalia	Especificação	Híbrida
[7], [21], [24], [26], [45], [56], [63], [72]	[1], [3], [5], [6], [8], [10], [12], [13], [14], [15], [16], [17], [18], [19], [22], [23], [25], [27], [28], [29], [30], [31], [34], [35], [36], [38], [39], [40], [41], [42], [43], [44], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [57], [58], [59], [61], [62], [64], [67], [68], [69], [70], [71], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [87], [88], [89], [90], [91], [92], [93], [94], [95], [97], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108]	[2], [4], [9], [32], [33], [37]	[7], [21], [26], [56], [63], [72]

A maioria dos estudos propôs abordagens de detecção baseadas em comportamento, muitas vezes chamada de detecção por anomalia. A detecção de anomalias assume que qualquer atividade anômala é necessariamente uma intrusão (Bace; Mell, 2001). Qualquer atividade que não se enquadre nos modelos de comportamento definidos como normais é considerada um ataque. Este fato permite detectar novos ataques e/ou variações dos conhecidos. No entanto, pode sofrer com alta taxa de falsos positivos, pois, nem toda atividade anômala é uma intrusão (Boukerche et al., 2007). Conforme pode ser observado na Tabela 48, 92 trabalhos propuseram abordagens de detecção por anomalia. Como esta estratégia geralmente é modelada com a utilização de técnicas de Aprendizado de Máquina (Machine Learning - ML), maiores detalhes a respeito dessas técnicas são apresentados na QPERS6 (Seção B.5.7).

Além disso, 6 trabalhos consideram uma ramificação da análise por comportamento chamada de análise por especificação (Mitchell; Chen, 2014). Este tipo de solução emprega regras e limites que definem o comportamento padrão esperado para os componentes monitorados. É semelhante à detecção de anomalias. Ambos detectam intrusões quando o comportamento da rede se desvia do especificado. A principal diferença entre eles é que na análise baseada em especificações, um especialista humano é quem estabelece as regras (Mitchell; Chen, 2014; Zarpelão et al., 2017). Yaseen et al. (2016) e Yaseen et al. (2017) propuseram abordagem baseada em especificação de limites para detecção de ataques *selective forwarding* em *Wireless Sensor Networks* (WSNs). Ela se baseia em *watchdogs* que mantêm tabelas para pacotes de entrada e pacotes encaminhados. Essas tabelas são usadas para calcular as taxas de descarte de pacotes dentro das janelas de tempo. Os IDSs em servidores de nevoeiro analisam as informações recebidas dos *watchdogs* e com base em limites predefinidos determinam quais nós são maliciosos e quais deles são benignos. Aliyu, Sheltami e Shakshuki (2018) propuseram uma abordagem de detecção baseada em desafio/Resposta. Os nós de detecção interrogam

periodicamente os nós próximos, enviando pacotes de interrogação. O nó IDS observa o comportamento do receptor. Espera-se que ele decodifique o pacote e multiplique a carga útil por 2, então criptografe o resultado e o envie ao nó IDS. Quando o pacote retornado falha neste critério, o IDS pode concluir que o nó é malicioso. Os autores Zhou, Guo e Deng (2019) propuseram uma abordagem de detecção de intrusão baseada em regras para detecção no contexto da Internet das Coisas Industrial (*Industrial Internet of Things* - IIoT). Potrino, Rango e Santamaria (2019) e Potrino, Rango e Fazio (2019) propuseram abordagens de detecção e mitigação para um ambiente com comunicação através do protocolo MQTT. A abordagem usa restrições preventivas e políticas de descarte de pacote com base em limites nos diferentes tópicos definidos por meio do MQTT. A grande desvantagem deste tipo de análise é a especificidade necessária e o conhecimento necessário do domínio para especificar o comportamento benigno.

Além disso, conforme apresentado na Tabela 48, dentre os trabalhos selecionados foram encontradas 6 abordagens que combinaram as categorias de detecção por assinatura e por anomalia (Sandhu; Sohal; Sood, 2017; Sohal et al., 2018; Arshad et al., 2019). A abordagem COLIDE (Arshad et al., 2019), por exemplo, realiza detecção por assinatura nos dispositivos IoT e por anomalia nos nodos da *fog*. Lawal, Shaikh e Hassan (2020) e Lawal, Shaikh e Hassan (2021) propuseram estruturas de detecção de intrusões baseadas em assinatura e anomalia, usando computação em nevoeiro para garantir a detecção de ataques rápida e precisa, bem como escalabilidade. Primeiramente, é realizada uma verificação em relação a assinaturas de ataques detectados anteriormente. Posteriormente modelos de ML são responsáveis por distinguir entre o fluxo de tráfego de rede normal e intrusivo. Os autores Ghazi e Rachid (2020) propuseram uma abordagem detecção baseada em coleta de assinaturas de uma *honeypot* instalada na internet e em um sistema detector de anomalias baseado em Rede Neural Artificial (*Artificial Neural Networks* - ANN) que ajudará a aumentar a taxa de detecção de um IDS online.

As abordagens encontradas no estado da arte que focam na detecção por assinatura não são capazes de detectar novos ataques ou variações de ataques conhecidos (Sohal et al., 2018; Arshad et al., 2019; Lawal; Shaikh; Hassan, 2021). Por outro lado, as abordagens com análise por especificação, que detectam intrusões quando o comportamento da rede se desvia do especificado, estão intimamente ligadas a um protocolo ou ataque específico (Yaseen et al., 2016; Aly et al., 2019; Potrino; Rango; Santamaria, 2019). Finalmente, as abordagens baseadas em detecção por anomalia, que consideram que todo comportamento anômalo é uma intrusão e são capazes de detectar novos ataques, possuem dificuldades relacionadas a falsos positivos (Boukerche et al., 2007). Desse modo, pesquisas propondo abordagens de detecção híbrida, combinando categorias de detecção, são interessantes e necessárias para maximizar as vantagens e minimizar as desvantagens desses tipos de análise e também para obter de uma solução completa.

B.5.5 QPERS4 - As abordagens focam em identificação (binária) ou classificação (multiclasse)?

Os métodos de detecção podem ter como foco uma detecção binária (identificação) ou multi-classe (classificação). Na detecção binária os eventos são identificados como maliciosos ou não maliciosos. A maioria das abordagens encontradas no estado da arte focaram em realizar detecção binária (ataque ou não ataque). Conforme apresentado na Tabela 49, 60 trabalhos tiveram como foco realizar esse tipo de detecção. Neste caso é realizada apenas a identificação de que um ataque está ocorrendo. No entanto, os métodos binários não são capazes de identificar o tipo ou categoria de ataque.

Tabela 49 – Objetivo da detecção.

Detecção binária	**Detecção multiclasse**
[1], [2], [3], [4], [5], [7], [9], [10], [12], [13], [14], [18], [19], [21], [22], [23], [24], [25], [26], [28], [29], [32], [33], [34], [35], [37], [38], [41], [44], [46], [47], [48], [49], [50], [51], [53], [54], [56], [59], [60], [62], [64], [67], [70], [72], [75], [77], [79], [80], [84], [85], [87], [88], [91], [94], [95], [99], [103], [105], [106], [108]	[6], [8], [15], [16], [17], [27], [30], [31], [36], [39], [40], [42], [43], [52], [55], [57], [58], [61], [63], [68], [69], [71], [73], [74], [76], [78], [81], [82], [83], [89], [90], [92], [93], [97], [100], [101], [102], [104], [107]

O outro tipo de detecção é a multiclasse. Neste caso as abordagens têm como foco a classificação do evento em categorias específicas ou em comportamento normal. Dentre os trabalhos encontrados no estado da arte, 39 apresentaram como foco a detecção multiclasse. A detecção multiclasse possui um papel importante na detecção de intrusão, pois, é interessante classificar os ataques em categorias para que sejam executadas contramedidas especificas para determinados tipos de ameaças. Por exemplo, um ataque da categoria de ataque de sondagem geralmente é executado antes de ataques mais poderosos, como *Denial of Service* (DoS), *Distributed DoS* (DDoS), ataques de acesso remoto, etc (Nguyen et al., 2019). Assim, ao detectar um ataque da categoria de ataque de sondagem, é interessante executar mecanismos de detecção adicionais para reforçar a segurança. Além disso, a classificação do tipo ou categoria do ataque é importante para a tomada de decisão do responsável pela rede, pois, determinado tipo de ataque pode indicar uma vulnerabilidade no ambiente. No entanto, observa-se no estado da arte que estas abordagens apresentam taxas de acurácia inferiores aos métodos de detecção binários (Xu; Qian; Hu, 2019; Nguyen et al., 2019; Samy; Yu; Zhang, 2020; Sarwar et al., 2022; Zhao et al., 2022). Além disso, algumas apresentam dificuldades relacionadas a problemas de falsos positivos e a baixa detecção de alguns tipos de ataques (Diro; Chilamkurti, 2018a; Almiani et al., 2020; Du et al., 2020; Abdel-Basset et al., 2021).

Portanto, uma das principais lacunas no estado da arte, é a falta de uma abordagem de detecção de anomalias multiclasse, capaz de identificar o tipo ou categoria do ataque, alcançando taxas de acerto semelhantes ou melhores que a detecção binária, sem sobrecarregar o

nevoeiro. Atualmente os métodos híbridos e ensemble vem ganhando mais destaque por conseguirem melhores resultados, no entanto, possuem um custo maior. Desse modo é necessário aliar o desempenho de detecção com o custo computacional de abordagens mais robustas ao se projetar a solução de detecção.

B.5.6 QPERS5 - Quais técnicas de aprendizado de máquina são usadas na detecção de intrusão?

Nesta seção, diversas técnicas de *Machine Learning* (ML), *Deep Learning* (DL) e *Ensemble Learning* (EL) usadas na detecção de intrusões em ambientes de computação em neblina e IoT são apresentadas e discutidas. Eles são frequentemente empregados em estratégias de detecção baseadas em comportamento.

As técnicas são organizadas de acordo com o tipo de aprendizagem. São destacadas 4 categorias: aprendizado supervisionado (*supervised learning*), aprendizado não-supervisionado (*unsupervised learning*), aprendizado por reforço (*reinforcement learning*) e aprendizado semi-supervisionado (*semi-supervised learning*). A categoria com maior número de trabalhos é a supervisionada. A principal característica desse tipo de abordagem é a existência de rótulos no subconjunto de dados de treinamento. Este tipo de aprendizado reflete a capacidade de um algoritmo de generalizar o conhecimento a partir dos dados disponíveis com casos alvo ou rotulados, de modo que o algoritmo possa ser usado para prever novos casos não rotulados (Berry; Mohamed; Yap, 2019). Desse modo, o processo de treinamento dos métodos utiliza este conhecimento prévio para treinar e gerar os modelos de classificação. Após o treinamento os métodos estão aptos para classificar novos dados. A dificuldade dessa abordagem está na necessidade de dados rotulados para o treinamento dos modelos (Russell; Norvig, 2009).

Inicialmente a Tabela 50 apresenta as técnicas baseadas em aprendizado supervisionado, as quais são discutidas a seguir.

- ***K-Nearest Neighbour (KNN):*** o objetivo principal do algoritmo kNN é determinar a classe de uma nova instância através do calculo da sua semelhança com os exemplos existentes no banco de dados. O algoritmo identifica os K vizinhos mais próximos ao novo ponto de dados e o classificar de acordo com a classe majoritária entre estes vizinhos mais próximos (Mitchell, 1997). Lawal, Shaikh e Hassan (2021) avaliaram as métricas de distância Euclidean, Manhattan e Chebychev para calcular a similaridade, os melhores resultados foram alcançados pelas duas primeiras. O algoritmo KNN é comumente usado em abordagens de detecção de intrusão devido à sua baixa taxa de erro (Illy et al., 2019; Kumar; Gupta; Tripathi, 2020). Em Lawal, Shaikh e Hassan (2021), é utilizado um banco de dados que armazena assinaturas de ataques detectados anteriormente e um esquema de detecção baseado em anomalias que usa o algoritmo de classificação KNN para detectar ataques DDoS. No entanto, o trabalho se concentrou apenas na detecção de DDoS e não é capaz de identificar outros ataques. Uma desvantagem existente no KNN

é o custo computacional, que pode se tornar alto porque é necessário comparar as novas instâncias com todas as instâncias armazenadas na base do exemplo. Souza et al. (2020) propôs uma abordagem que utiliza o algoritmo KNN como segundo nível de detecção, considerando $k = 1$ e a distância Euclidiana como medida de similaridade. Primeiro, na abordagem, uma rede neural é usada. Assim, foi possível atingir uma redução no custo computacional do KNN de aproximadamente 89%.

- ***Decision Tree (DT):*** é um dos algoritmos de classificação mais amplamente usados em mineração de dados (Rokach, 2016; Kumar; Gupta; Tripathi, 2021b). As DTs têm uma estrutura semelhante a uma árvore e são compostas por nós (Breiman et al., 1984). As DTs classificam os dados usando um conjunto de decisões de recursos hierárquicos. As decisões tomadas em nós internos são os critérios de divisão (Dev; Eden, 2019). Peng et al. (2018) investigou o algoritmo de árvore de decisão denominado CART, usando o critério de Gini (Breiman et al., 1984), para detectar intrusão de *big data* no nevoeiro, a abordagem obteve resultados superiores aos modelos KNN e bayesianos em todas as classes de ataques do conjunto de dados KDDCUP99. No entanto, este conjunto de dados não tem tráfego IoT característico e é bastante antigo, então a abordagem deve ser avaliada com uma análise mais complexa com um conjunto de dados mais atual. Os autores (Maharaja; Iyer; Ye, 2019) investigou DT, usando critérios de entropia, para detecção binária de ataques DDoS. Além disso, em alguns trabalhos foram utilizadas DTs com técnicas ensemble para combinação de várias DTs e outros algoritmos (Illy et al., 2019; Kumar; Gupta; Tripathi, 2021b).

- ***Random Forest (RF):*** é um método *ensemble* que cria um grande número de DTs não correlacionados acumulados dentro de uma "floresta". Fornece uma camada adicional de aleatoriedade sobre a estratégia *ensemble Bagging*. Além de construir cada árvore usando uma amostra de *bootstrap* diferente dos dados, a RF muda como as árvores de classificação são construídas. Ao contrário do DT padrão, onde cada nó é dividido usando a melhor divisão entre todas as variáveis, em RF cada nó é dividido usando o melhor entre um subconjunto de preditores escolhidos aleatoriamente nesse nó (Breiman, 2001; Liaw; Wiener et al., 2002). Uma vantagem em relação a DT, é a robustez contra *overfitting* (Breiman, 2001). Vários trabalhos recentes pesquisaram o uso de RF em abordagens de detecção de intrusão (Illy et al., 2019; Kumar; Gupta; Tripathi, 2020; Kumar; Gupta; Tripathi, 2021b). Os autores Farukee et al. (2020), entretanto, utilizaram RF para selecionar as características principais do tráfego a ser submetido a outro classificador. Um aspecto importante da RF é o cálculo da importância do recurso. O índice de critério de impureza de Gini (Breiman et al., 1984) é usado. Assim, eles usaram a propriedade RF para classificar os recursos de acordo com sua importância. O RF também foi utilizado em novos métodos *ensemble* (Kumar; Gupta; Tripathi, 2021b). Kumar et al. (2020) avaliou RF no contexto de detecção de ataque DDOS e descobriu que o fato de RF ser insensível a *outliers*, valores ausentes, *overfitting* e ter a capacidade de lidar com um grande número

de tráfego de entrada o torna adequado na anomalia do processo ferramenta de detecção para o ambiente blockchain-IoT.

Tabela 50 – Métodos de aprendizado de máquina supervisionados.

Método	Trabalhos	Comentários
KNN	[27], [53], [57] [72], [83], [100]	Dificuldade em determinar o valor ideal de k. O custo computacional da previsão pode se tornar alto.
DT	[19], [27], [29] [70], [83]	Problemas de *overfitting* são um dos desafios em DT (T.K.; Annavarapu; Bablani, 2021).
RF	[57], [58], [62] [70], [89], [92] [101], [102], [104]	Robusto para *overfitting* e ruído. Neste contexto, o custo pode ficar alto de acordo com o número de árvores e sua profundidade.
ET	[38], [102]	Robusto para *overfitting* e ruído. Etapa extra de aleatoriedade sobre RF, mas com a mesma dificuldade.
XGBoost	[57], [58], [63], [75] [82], [91], [94], [104]	Resiliente contra *overfitting*. Pode ter dificuldades com *outliers*.
ADT	[19]	Executa *boosting* em uma única árvore de decisão facilitando a compreensibilidade (Freund; Mason, 1999).
SVM	[30], [36], [44], [56], [61], [67]	SVMs são sensíveis ao ruído perto do hiperplano (Liu; Lang, 2019b). A seleção de kernel ideal é difícil (Al-Garadi et al., 2020).
ANN	[15], [19], [28], [46], [48] [50], [52], [56], [59], [67] [69], [78], [85], [88], [93] [97], [102], [103], [105] [106], [107]	O treinamento do modelo consome muito tempo. Propenso a ficar preso em um ótimo local (Liu; Lang, 2019b). Pode apresentar instabilidade com poucos dados de treinamento (NG; Selvakumar, 2020).
RNN	[19], [40], [42]	Pode sofrer de memória de curto prazo (Ahmad et al., 2021). Sujeito a desaparecimento/explosão de gradientes (Samy; Yu; Zhang, 2020).
LSTM	[3], [8], [13], [34], [35] [39], [52], [79], [90]	O treinamento é caro em termos de recursos e tempo. Projetado para lidar com problemas de gradiente que desaparecem ou explodem (Samy; Yu; Zhang, 2020).
GRU	[40], [52]	Estrutura mais simples que o LSTM (Liu; Lang, 2019b). Capaz de resistir ao problema do gradiente de desaparecimento e treinar mais rápido devido ao seu pequeno número de cálculos (Samy; Yu; Zhang, 2020).
CNN	[35], [43], [52], [62] [68], [77], [90]	Tem um alto custo computacional, então implementá-los em um contexto com recursos limitados é um desafio (Al-Garadi et al., 2020).
GNN	[47]	Capacidade de modelar dependências entre nós em um gráfico.
ELM	[10], [17], [18] [25], [64], [73]	Eles são menos precisos do que as redes tradicionais, mas são interessantes para um retreinamento em tempo real. Velocidade de aprendizado extremamente rápida (Huang; Zhu; Siew, 2006).
Naive Bayes	[70], [74], [81]	São necessários poucos dados de treinamento. Apresentou dificuldades de detecção em comparação com outros métodos.

- ***Extra Tree (ET):*** é um método *ensemble* similar a RF, varia principalmente deste na forma de construir os DTs dentro da floresta (Kaur; Mittal, 2020). Em ET, a aleatoriedade vai um passo além na forma como as divisões são calculadas, em vez de procurar os pontos de corte mais discriminantes, eles são sorteados aleatoriamente para cada atributo candidato e o melhor desses pontos de corte gerados é escolhido aleatoriamente como o regra de divisão. Portanto, a estratégia concentra-se em randomizar fortemente a escolha dos atributos e o ponto de corte dos atributos enquanto divide um nó na árvore. No caso extremo, ele constrói árvores totalmente aleatórias, cujas estruturas são independentes dos valores de saída da amostra de aprendizagem (Geurts; Ernst; Wehenkel, 2006). As previsões em árvore são agregadas para produzir a previsão final, por maioria de votos em problemas de classificação e média aritmética em problemas de regressão (Geurts; Ernst; Wehenkel, 2006). Além da precisão, o principal ponto forte do algoritmo resultante é a eficiência computacional, pois, dada a simplicidade do procedimento de divisão do nó, o fator constante pode ser muito menor do que em outros métodos de ensemble que otimizam localmente os pontos de corte (Geurts; Ernst; Wehenkel, 2006). Albdour, Manaseer e Sharieh (2020) propuseram uma abordagem de detecção de intrusão baseada em ET para a camada do nevoeiro.

- ***Extreme Gradient Boosting (XGBoost):*** os autores Lawal, Shaikh e Hassan (2020) apresentaram uma abordagem focada em XGBoost para detecção de anomalias em um *framework* IoT (Kumar et al., 2022). Este algoritmo é uma variante aprimorada do algoritmo de aumento de gradiente (Chen; Guestrin, 2016). No XGBoost, as previsões de métodos fracos são combinadas para desenvolver um método forte, empregando técnicas aditivas. Ele cultiva interativamente milhares de árvores usando informações de uma árvore cultivada anteriormente. Este método aprende com o erro das árvores anteriores para melhorar a precisão nas iterações subsequentes (Kumar; Gupta; Tripathi, 2020). É chamado de aumento de gradiente porque usa um algoritmo de queda de gradiente para minimizar a perda ao adicionar novos modelos. Além dos benefícios de velocidade e desempenho do XGBoost, as vantagens adicionais são evitar *overfitting* e utilização total dos recursos computacionais (Lawal; Shaikh; Hassan, 2020). Para remover vieses e *overfitting*, ele usa o algoritmo L1 (*Least Absolute Shrinkage and Selection Operator*), e L2 (*Ridge Regression*) (Kumar; Gupta; Tripathi, 2020).

- ***Altenate Decision Tree (ADT):*** fornece um mecanismo para combinar as hipóteses fracas geradas durante o *boost* em uma única representação interpretável. Um ADT consiste em uma estrutura com nós de decisão alternados, que especificam uma condição de predicado, e nós de predição contendo um único número. Um ADT classifica uma instância seguindo todos os caminhos para os quais todos os nós de decisão são verdadeiros e somando todos os nós de predição percorridos. Gera regras que geralmente são menores em tamanho e portanto mais fáceis de interpretar (Freund; Mason, 1999). Shafi et al. (2018)

propuseram uma abordagem de detecção de *ensemble* com vários métodos de aprendizado de máquina, incluindo ADT.

- ***Support Vector Machine (SVM):*** nesse método é criado um hiperplano de divisão nos atributos de dados entre duas ou mais classes, de forma que a distância entre o hiperplano e os pontos de amostra mais adjacentes de cada classe seja maximizada (Tong; Koller, 2002). Basicamente utiliza uma função kernel para mapear o conjunto de dados separáveis não lineares para o espaço de alta dimensão, de modo que tenha separabilidade para obter a classificação (Du et al., 2020). A abordagem proposta por Du et al. (2020) conta com uma SVM com parâmetros otimizados pelo algoritmo *Particle Swarm Optimization* (PSO) para obter o classificador de detecção de intrusão SVM ideal. Miranda et al. (2020) apresentaram uma abordagem com 3 camadas de segurança, neste caso implantou um mecanismo baseado em SVM no topo do *framework* para segurança em *Wireless Sensor Networks* (WSN). O IDS distribuído e baseado em nuvem proposto em (Ghazi; Rachid, 2020) tem como objetivo minimizar falsos positivos e maximizar a detecção de ataques do dia zero em um IDS híbrido baseado em SVM incremental online instalado em uma arquitetura de nevoeiro.

- ***Artificial Neural Networks (ANN):*** as redes neurais *MultiLayer Perceptron* (MLP) são compostas por um grande número de neurônios artificiais organizados em camada de entrada, camada oculta e camada de saída (Haykin, 2001). Dessa forma, os neurônios de uma camada são conectados aos neurônios da camada seguinte através de links ponderados (Kaviani; Sohn, 2021). Não há presença de links recursivos nessa topologia de rede. O modelo neural é gerado através do processo de treinamento supervisionado, onde os pesos dos links são atualizados em várias iterações com base no erro estimado. Destaca-se que ANNs são capazes de resolver problemas não lineares, desse modo, são bastante empregadas em tarefas de classificação e de detecção de intrusão (Khater et al., 2019; Thi-Nga et al., 2020; Rahman et al., 2020; Aliyu et al., 2022; Kumar; Tripathi; Gupta, 2021). Alguns estudos trabalharam com ANN para identificar intrusões nos próprios dispositivos da camada de *fog computing*, através da analise de dados de chamadas do sistema dos próprios dispositivos (Khater et al., 2019; Pacheco et al., 2020). Ghazi e Rachid (2020) trabalharam com redes neurais como um segundo nível de detecção, atuando na nuvem, para reavaliar o tráfego previamente classificado como normal por uma SVM. O objetivo é garantir que nenhum novo ataque tenha passado despercebido. As redes neurais profundas (*Deep Neural Networks* - DFFN) são avanços sobre as redes tradicionais simples. Adicionando mais camadas e mais unidades dentro de uma camada, uma rede profunda pode representar funções de complexidade crescente (Goodfellow et al., 2016). Diversos trabalhos investigaram redes profundas para detectar intrusões (Lalouani; Younis, 2021; Kumar; Tripathi, 2021; Razaque et al., 2022; Souza; Westphall; Machado, 2022; Rey et al., 2022). Os autores Diro e Chilamkurti (2018c) propuseram uma nova abordagem distribuída baseada em DNN para detectar intrusões no ambiente

IoT. Em Sugi e Ratna (2020), também foi proposta uma abordagem baseada em DNN com mecanismo de treinamento distribuído nos nós de nevoeiro. A abordagem SDRK (Ravi; Shalinie, 2020) também funciona com DNN. Souza et al. (2020) propuseram um método híbrido composto por uma DNN de duas camadas ocultas com função de ativação tangente hiperbólica e dois neurônios na camada de saída com função softmax. Essa saídas são utilizadas para discriminar quais instâncias são enviadas para um segundo método de classificação, no caso o algoritmo KNN. A abordagem alcançou melhoria nas taxas de detecção e reduziu o tempo de processamento do KNN em aproximadamente 89%. Observa-se também que várias abordagens trabalharam com quantidades reduzidas de neurônios nas camadas intermediárias de modo a ser uma solução leve para o ambiente de computação em nevoeiro. Uma das principais dificuldades das abordagens neurais *feedforward* é o custo para o treinamento do modelo. De acordo com Huang, Zhu e Siew (2006), isso pode ocorrer devido a certa lentidão os algoritmos de aprendizagem baseados em gradiente amplamente usados para treinar redes neurais e ao processo iterativo de ajuste dos parâmetros das redes.

- ***Extreme learning machine (ELM):*** surgiu para superar as limitações da ANN (Huang; Zhu; Siew, 2006). ELM é uma rede neural *feedforward* de camada oculta única que escolhe aleatoriamente os pesos de entrada e o viés da camada oculta sem ajuste e determina os pesos de saída de forma analítica. Esse algoritmo tende a fornecer bom desempenho de generalização em velocidade de aprendizado extremamente rápida. Por esse motivo, diversas pesquisas foram realizadas com este métodos (Pirozmand et al., 2020; Gavel; Raghuvanshi; Tiwari, 2021). An et al. (2018c) apresentaram uma abordagem IDS leve para dispositivos da camada da fog e edge, baseada em ELM. Conforme citado, essas redes possuem características de alta velocidade de treinamento e boa capacidade de generalização. Em Rathore e Park (2018) é proposto uma abordagem com técnicas não supervisionadas e um modelo ELM com função de ativação softmax na camada de saída. Prabavathy, Sundarakantham e Shalinie (2018) propuseram uma abordagem baseada em *Online Sequential ELM* (OS-ELM) (Huang et al., 2005). OS-ELM é uma abordagem usada para problemas de classificação em tempo real podendo ser adotada para detectar ataques cibernéticos em ambiente IoT que geram dados em alta velocidade. OS-ELM fornece um modelo de aprendizado rápido que pode se adaptar a novos dados de dispositivos IoT rapidamente, juntamente com um bom poder de generalização. Em (Alrashdi et al., 2019) também foi proposta uma estrutura de detecção de ataque baseada em OS-ELM. No entanto, nesse caso criou-se uma abordagem ensemble para detectar com eficiência atividades maliciosas.

- ***Recurrent Neural Network (RNN):*** é um tipo de rede neural profunda que estende os recursos da rede neural tradicional *feed-forward* e é projetada para modelar dados de sequência. Elas podem usar seu estado interno de memória para processar sequências de comprimento variável de entradas (Ahmad et al., 2021). Shafi et al. (2018) propuseram

uma abordagem ensemble com RNN, MLP e ADT, eles destacam que RNN detecta com alta precisão ataques de alta intensidade, mas não tem um bom desempenho em baixa intensidade. Em Almiani et al. (2020) é apresentada uma abordagem em cascata com RNN para detecção multiclasse de intrusões. As RNNs possuem algumas dificuldades. A RNN básica normalmente é capaz de lidar com sequências de comprimento limitado e sofrerá de memória de curto prazo se o comprimento da sequência for longo (Ahmad et al., 2021). Além disso, está sujeita a problemas de gradiente que desaparecem. Desse modo, variações foram propostas para superar esses problemas (Samy; Yu; Zhang, 2020).

- ***Long Short Term Memory (LSTM):*** consiste em uma variação de RNN projetada para lidar com problemas de gradiente em expansão e desaparecimento (Samy; Yu; Zhang, 2020). Ela é capaz de lembrar informações por um longo período de tempo. Essa propriedade de lembrar padrões dá ao LSTM uma vantagem sobre os RNNs tradicionais (Hochreiter; Schmidhuber, 1997). Diro e Chilamkurti (2018a) propuseram uma arquitetura distribuída baseada em redes LSTM para detecção de ataques, na qual os nós de nevoeiro hospedam funções de detecção. Ela distribui cálculos, controles e dados armazenados em nós locais para que cada nó detecte a intrusão de dispositivos IoT próximos enquanto troca experiências aprendidas com nós vizinhos por meio de um nó coordenador. Em Priyadarshini e Barik (2022) uma série de investigações foram realizadas com arquiteturas de redes LSTM para a detecção de ataques DDoS em ambientes baseados em nevoeiro. Roopak, Tian e Chambers (2019b) realizaram uma série de experimentos, o primeiro modelo é baseado em MLP, o segundo em CNN e o terceiro em LSTM. Por fim, o quarto modelo consiste em uma abordagem hibrida com CNN e LSTM. A avaliação de desempenho foi realizada usando o conjunto de dados CICIDS2017. Os resultados obtidos demonstram que CNN+LSTM alcançaram melhor desempenho. Os autores Samy, Yu e Zhang (2020) também investigaram redes LSTM e propuseram uma estrutura de detecção de ataques em tráfego IoT. No framework proposto, o tráfego da camada de borda é coletado e enviado para a camada de nuvem para treinar o modelo LSTM. Em seguida, o modelo treinado instalado nos nós da camada de nevoeiro como um mecanismo de detecção para detectar ataques.

- ***Gated Recurrent Unit (GRU):*** consiste em um método baseado no modelo de rede LSTM, mas com poucos parâmetros. Ela possui uma estrutura mais simples e menos componentes de célula do que LSTM. O modelo GRU mescla a porta *forget* e a porta *input* em uma única porta de atualização, que é mais simples do que o LSTM (Liu; Lang, 2019b). Ela é capaz de resistir ao problema do gradiente de desaparecimento e treina mais rápido devido ao seu pequeno número de cálculos (Samy; Yu; Zhang, 2020). Os autores Abdel-Basset et al. (2020) trabalharam com RNN usando a unidade recorrente local com portas (LocalGRU) para aprender representações locais e introduziram uma camada de *Multi-head Attention* (MHA) para capturar e aprender a representação global de longo prazo. Uma camada *feed-forward* é proposta para realizar uma transformação não linear de re-

cursos. Os recursos compactados são alimentados em camadas totalmente conectadas para calcular a decisão de classificação final.

- ***Graph Neural Network (GNN):*** são métodos baseados em aprendizado profundo que operam no domínio de grafos (Zhou et al., 2020). Desse modo, têm a capacidade de extrair recursos espaciais localizados em várias escalas e compô-los para construir representações altamente expressivas. Essencialmente, cada nó no gráfico está associado a um rótulo, e o objetivo é prever o rótulo dos nós sem base de dados. Os autores Protogerou et al. (2020) propuseram um sistema multiagente baseado em GNN, a fim de explorar a natureza colaborativa e cooperativa de agentes inteligentes para detecção de anomalias, de modo a monitorizar de forma eficiente toda a infraestrutura da rede. A GNN é capaz de reconhecer a complexidade das relações entre os nós de forma otimizada, representando suas interdependências de uma forma poderosa (Protogerou et al., 2020).

- ***Convolutional Neural Network (CNN):*** é um tipo de rede neural que usa convolução no lugar da multiplicação geral da matriz, em pelo menos uma de suas camadas (Goodfellow et al., 2016). Essas redes tem alcançado ótimos resultados em problemas de processamento de dados bidimensionais com topologia em grade, como imagens e vídeos (Liu et al., 2017). A arquitetura da CNN consiste em três camadas empilhadas: camada convolucional, camada de *pooling* e camada totalmente conectada. As camadas convolucionais e de *pooling* são conectadas alternadamente e formam a parte intermediária da rede. As camadas convolucionais aplicam vários filtros (*kernels*) de tamanho igual para produzir mapas de recursos. As camadas de *pooling* substituem as saídas da rede em um determinado local por estatísticas resumidas das saídas próximas. A objetivo da camada de *pooling* é diminuir o tamanho da imagem para encontrar possíveis padrões que possam ficar mais evidentes. A dimensão dos dados de entrada é reduzida usando estas camadas de *pooling*. Ele combina um cluster em um único ponto. Por fim, a camada totalmente conectada combina as entradas de todas as posições para realizar a classificação (Farukee et al., 2020). Em NG e Selvakumar (2020) foi proposto um algoritmo para detectar anomalias no tráfego de IoT usando *Vector Convolutional Deep Learning* (VCDL) (Amma; Subramanian, 2018) com estrutura escalonável de aprendizado em um ambiente de nevoeiro distribuído. Farukee et al. (2020) propôs uma abordagem baseada em RF e CNN para detecção de DDoS. RF foi utilizado para a seleção dos recursos analisados e a CNN para a classificação. Uma vez que CNN trabalha com dados na forma 3D e os dados do trafego são 2D, eles foram remodelados para 3D antes de ajustá-los ao modelo. No entanto o modelo não foi projetado para detecção multiclasse. Motivados pelo sucesso do *Temporal CNN* (TCN) em uma variedade de aplicações, Abdel-Basset et al. (2021) propuseram redesenhar e melhorar a arquitetura TCN para detecção de intrusão e classificação de ataques a partir de dados de tráfego de IoT.

- ***Naïve Bayes (NB):*** tem como conceito adquirir, por meio do treinamento, a probabilidade

condicional. Basicamente, é a probabilidade de um evento ou resultado ocorrer com base em ocorrências anteriores. Ele estima a média e a variância para cada recurso usando a abordagem de máxima verossimilhança. Além disso, ele assume que todos os vetores de recursos são independentes. Ele tem uma base simples e confiável para modelar dados e é bastante resiliente a valores discrepantes e ausentes (Iwendi et al., 2020; Manimurugan, 2021; Kumar; Gupta; Tripathi, 2021b). *Naïve Bayes* não foi aplicada diretamente, pois, ela apresenta acurácia menor que outros métodos (Prabavathy; Sundarakantham; Shalinie, 2018; Souza et al., 2020). Foi utilizado como método complementar em abordagens ensemble (Kumar; Gupta; Tripathi, 2020; Kumar; Gupta; Tripathi, 2021b) e abordagens hibridas onde se buscou propor uma abordagem melhorada de *Naïve Bayes* (Manimurugan, 2021).

O aprendizado não-supervisionado também foi utilizado em vários trabalhos. A aprendizagem não supervisionada refere-se ao processo de agrupar dados usando métodos automatizados ou algoritmos, em dados não rotulados. Nessa situação, os algoritmos precisam compreender os relacionamentos ou recursos subjacentes dos dados disponíveis e agrupar casos com recursos ou características semelhantes (Berry; Mohamed; Yap, 2019). Muitas abordagens agrupam os dados com base em relações entre as variáveis (Russell; Norvig, 2009).

Na Tabela 51 são apresentadas as técnicas baseadas em aprendizado não supervisionado, as quais são discutidas a seguir.

- ***AutoEncoder (AE):*** é um tipo de rede neural com múltiplas camadas cuja saída alvo é igual à entrada com alguma quantidade de erro de reconstrução (Thi-Nga et al., 2020). Os AE se baseiam em aprendizado não supervisionado para compactar através de um codificador, a entrada em uma representação de espaço latente com recursos mínimos, conhecida como *bottleneck*. O decodificador ajuda o modelo a reconstruir o saída da representação codificada (Ng et al., 2011). Por fim, a avaliação do desempenho do decodificador e a medição da similaridade entre a saída obtida e a entrada original é chamada de *Reconstruction Loss*. Espera-se que, treinando o AE para copiar a entrada para a saída, a representação latente tenha propriedades úteis. Os *Sparse AutoEncoder* (Sparse AE) possuem critério de treinamento envolvendo uma penalidade de dispersão (Ng et al., 2011). Essa restrição de dispersão força o modelo a responder aos recursos estatísticos exclusivos dos dados de treinamento (Makhzani; Frey, 2014). Sadaf e Sultana (2020) propuseram uma abordagem para detecção de intrusão por anomalia em ambientes de nevoeiro baseado em Sparse AE que alimenta uma IF para identificar outliers. Em Kumar, Tripathi e Gupta (2021) é proposta uma abordagem IDS com um mecanismo de preservação de privacidade baseado em Sparse AE que transforma os dados originais em uma nova forma codificada que evita ataques de inferência. Posteriormente, uma ANN é utilizada para a etapa de análise e classificação dos dados. Diro e Chilamkurti (2018b) utilizou *Stacked AutoEncoder* (Stacked AE) como um meio de rede autodidata para extrair recursos ocultos, em seguida, os recursos obtidos foram aplicados aos dados de teste para

extrair os recursos finais para uma classificação softmax. Os Stacked AE são construídos através de vários Sparse AE empilhados uns com os outros de forma que as saídas de cada camada sejam alimentadas nas entradas da próxima camada. Em Moussa e Alazzawi (2020) foi proposto Stacked AE modificado que usa um processo gradual de codificação-decodificação para detecção de ataques em aplicações IoT automotivas. A abordagem usa várias camadas ocultas nos lados do codificador e do decodificador, simetricamente. São implementados vários AEs internos dentro do processo codificador-decodificador que permitem um tratamento gradual dos dados nas diferentes fases do modelo.

- ***Self Organizing Map (SOM):*** é um tipo de rede neural que é treinada usando aprendizagem não supervisionada para produzir uma rede de baixa dimensão. Possui a propriedade especial de criar "representações internas" espacialmente organizadas de vários recursos de sinais de entrada e suas abstrações (Kohonen, 1990). Os mapas auto-organizáveis diferem de outras redes neurais artificiais, pois aplicam aprendizagem competitiva em oposição à aprendizagem de correção de erros e no sentido de que usam uma função de vizinhança para preservar as propriedades topológicas do espaço de entrada (Kohonen, 1990). A abordagem supracitada proposta por Nguyen et al. (2019), utilizou SOM para atuar em um segundo nível de detecção no nevoeiro.

- ***Isolation Forest (IF):*** é um método baseado na premissa de que as anomalias tendem serem isoladas precocemente se o espaço de dados for dividido aleatoriamente (Liu; Ting; Zhou, 2008). Desse modo, a estrutura utilizada como base é similar a arvores binárias, elas são chamadas de iTree. O IF constrói um conjunto de iTrees, elas isolam as anomalias mais perto da raiz da árvore, permitindo a construção de modelos parciais e eficazes empregando apenas uma pequena proporção de dados de treinamento (Liu; Ting; Zhou, 2008). Sadaf e Sultana (2020) propuseram uma abordagem para detecção onde a saída gerada por um AE alimenta uma IF para identificar outliers. AE identifica o ataque e separa o ataque e os dados de tráfego de rede normal em dois conjuntos. No entanto, os conjuntos resultantes podem conter pontos de dados que não pertencem a eles. A IF no estágio 2, tenta identificar esses pontos de dados desajustados (*outlier*), o que melhora a precisão geral. O IF converge rapidamente com um número muito pequeno de árvores e requer apenas um pequeno tamanho de subamostragem para atingir um alto desempenho de detecção com alta eficiência. Além disso, não faz uso de medidas de distância, daí a redução encontrada no custo necessário para computação (Liu; Ting; Zhou, 2008).

- ***Abordagens de clusterização:*** A clusterização consiste em criação de *cluster* de agrupamento de dados que se assemelham de alguma forma (MacQueen et al., 1967). Um dos algoritmos mais tradicionais de clusterização é o K-means. Ele forma *K clusters* diferentes no conjunto de dados. Geralmente, por meio de alguma métrica de distância é encontrado o centro de *cluster* mais próximo, e se atribui a este *cluster* o dado que se quer agrupar. Posteriormente, os centróides são atualizados sempre tomando o valor mé-

dio de todos os dados naquele *cluster*. Ravi e Shalinie (2020) propuseram uma variante do K-means com *Repeated Random Sampling*, denominada algoritmo RRS-K-means, juntamente com uma DNN, para detectar intrusões em IoT. O método *Fuzzy C-Means* (FCM) (Pal et al., 2005) particiona os dados em *clusters* de forma que, de acordo com alguma medida, os pontos de dados mais semelhantes pertençam ao mesmo *cluster* e os que são diferentes pertençam a *clusters* diferentes. Além disso, permite a atribuição de um ponto de dados a diferentes classes com vários graus de associação e oferece melhorias conceituais adicionais no agrupamento. Assim, ele fornece uma maneira mais razoável de tratar padrões, e também pode identificar eventuais *outliers* (Pal et al., 2005). Em Zahra e Chishti (2020) é proposta uma arquitetura segura para IoT (FLFSIoT) baseada em nevoeiro e na técnica FCM. A lógica difusa foi usada para aliviar a incerteza de pertencer a um *cluster* nítido de um nó de borda e para detectar vários ataques clássicos. A arquitetura foi avaliada contra taques DDoS e de conluio alcançando desempenho superiores a trabalhos existentes. Para detectar intrusões sem usar nenhum conhecimento prévio, em Hosseinpour et al. (2016) é introduzido um mecanismo de agrupamento como uma resposta imune inata. O mecanismo de *clustering* emprega a técnica *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) (Ester et al., 1996) para classificar o tráfego de rede real em *clusters* e contá-los como self, enquanto comportamentos fora dos *clusters* serão considerados como ruído ou não-self. Basicamente, no DBSCAN, busca-se que para cada ponto de um *cluster*, uma vizinhança para um dado raio contém, no mínimo, certo número de pontos, ou seja, a densidade na vizinhança tem que exceder um limiar.

- ***Abordagens estatísticas:*** Os métodos probabilísticos são particularmente úteis na compreensão dos padrões presentes em dados (Bouguila; Ziou; Vaillancourt, 2004). Os modelos estatísticos de mistura, por exemplo, podem ser uma importante técnica para projetar um perfil totalmente legítimo do tráfego de rede. Os autores Moustafa et al. (2019) propuseram um mecanismo de aprendizado estatístico adversarial para detecção de anomalias no nevoeiro. A abordagem é capaz de se adaptar contra ataques de envenenamento de dados que injetam instâncias maliciosas na fase de treinamento para interromper o processo de aprendizagem. Em segundo lugar, ele estabelece um perfil estatístico legítimo e considera variações da linha de base do perfil como anomalias usando *Outlier Dirichlet Mixture* (ODM). A distribuição *Dirichlet* é a generalização multivariada da distribuição Beta, que oferece flexibilidade e facilidade de uso consideráveis. Em contraste com outras distribuições, a distribuição de *Dirichlet* permite múltiplos modos simétricos e assimétricos (Bouguila; Ziou; Vaillancourt, 2004). Os autores Moustafa et al. (2021) propuseram uma abordagem baseada em *Gaussian Mixture-based Correntropy*. Os *Gaussian Mixture Models* (GMM), podem ajustar com eficiência os limites legítimos de dados em distribuições multivariadas que consideram todas as variações dos dados legítimos, em vez de usar um conjunto de observações de dados estáticos. O *Principal Component Analysis*

Tabela 51 – Métodos de aprendizado de máquina não supervisionados.

Método	Trabalhos	Comentários
AE	[14], [30], [69] [89], [97], [107]	AEs consomem um tempo computacional considerável. Se o conjunto de dados de treinamento não for representativo do conjunto de dados de teste, AEs podem apenas complicar o processo de aprendizagem em vez de representar as características do conjunto de dados (Al-Garadi et al., 2020).
Isolation Forest	[51]	Requer apenas um pequeno tamanho de subamostragem. Não usa medidas de distância (Liu; Ting; Zhou, 2008).
K-means	[49]	Menos eficazes do que os métodos de aprendizagem supervisionados, especificamente na detecção de ataques conhecidos (Al-Garadi et al., 2020).
Fuzzy C-means	[18], [54]	O FCM pode levar a um mínimo local (PIMENTEL, 2017).
DBSCAN	[1], [83]	Pode não ser capaz de identificar corretamente um *cluster* quando as densidades variam amplamente de grupo para grupo (Tan; Steinbach; Kumar, 2006).
SOM	[30], [95]	Requer dados necessários e suficientes para desenvolver clusters significativos (Pang, 2003).
GMM	[71]	Quando tem um número insuficiente de pontos por mistura, estimar as matrizes de covariância torna-se difícil, e o algoritmo é conhecido por divergir e encontrar soluções com probabilidade infinita, a menos que se regularize as covariâncias artificialmente [2].
DMM	[38]	Para garantir o melhor desempenho, uma grande quantidade de instâncias puras legítimas é necessária para produzir as taxas de detecção mais altas (Moustafa et al., 2019).
PCA	[38], [56], [61], [71], [74], [76]	É um método de redução de atributos que precisa ser usado em conjunto com outros métodos de previsão para detectar intrusões.

(PCA) é outra técnica estatística presente no contexto de detecção de intrusões. Ela vem sendo bastante utilizada para determinar as características potenciais do tráfego de rede e remover os recursos irrelevantes ou ruidosos, reduzindo a dimensionalidade dos dados, eliminando a correlação entre atributos e reduzindo o tempo de treinamento (Moustafa et al., 2019; Ghazi; Rachid, 2020; Du et al., 2020; Manimurugan, 2021; Moustafa; Ahmed; Ahmed, 2020; Moustafa et al., 2021). O PCA é uma abordagem que seleciona um pequeno número de recursos não correlacionados, chamados de componentes principais, a partir de um grande número de recursos, uma vez que seu alvo determina a maior variância com o menor número de componentes principais (Abdi; Williams, 2010).

Outro tipo de aprendizagem é a por reforço, onde o método aprende como atingir uma meta em um ambiente incerto e potencialmente complexo. Tu et al. (Tu et al., 2018) propuseram uma abordagem que explora as propriedades do canal de comunicação entre os dispositivos e o nevoeiro para detectar ataques *impersonation*. Ele utiliza o algoritmo *Q-learning* para obter os valores de limite ideais para detecção. No algoritmo *Q-learning*, cada agente apreende a atingir a estratégia ótima. Os receptores constroem o teste de hipótese para avaliar o transmissor para cada pacote T reconhecido no intervalo de tempo para detecção de personificação com *Q-learning* (Tu et al., 2018).

Por fim, alguns trabalhos propuseram abordagens baseada em aprendizagem semi-supervisionadas. Neste tipo de aprendizado os métodos buscam apreender utilizando dados rotulados e não rotulados. Devido à enorme quantidade de registros de tráfego de IoT gerados

diariamente e ao esforço e tempo necessários para rotular cada registro, o aprendizado semi-supervisionado torna-se muito importante para treinar modelos usando uma combinação de um grande número de registros não rotulados e um pequeno número de registros rotulados (Berry; Mohamed; Yap, 2019; Abdel-Basset et al., 2021). A seguir são apresentadas algumas abordagens semi-supervisionadas, elas treinam um classificador específico usando dados rotulados e não rotulados. Algumas trabalhos apresentaram abordagens baseadas em métodos consolidados de aprendizado semi supervisionado.

Em Xu, Qian e Hu (2019), foi proposta uma abordagem de detecção com *One Class SVM* (OCSVM), uma variação das SVMs tradicionais. OCSVM operou com dados positivos e não rotulados para a detecção de anomalias. Esta técnica se baseia no conceito de *One Class Classification* (OCC) e tem como objetivo encontrar um hiperplano, ou seja, um limite de decisão, que separa a maioria dos dados da origem (Xu; Qian; Hu, 2019). Os autores Yahyaoui et al. (2021) também apresentaram uma abordagem de detecção de anomalia baseada em OCSVM, para compor um *framework* para IoT.

Os autores Rathore e Park (2018) propuseram um modelo que é uma combinação de ELM e *Semi-supervised Fuzzy C-Means* (SFCM). Inicialmente, o ELM é aplicado ao conjunto de dados rotulado. O conjunto de dados rotulado e o não rotulado são fornecidos como entrada para o FCM. A confiança na previsão de dados não rotulados é calculada. Se a confiança for maior do que um limite, ela é colocada no conjunto de dados rotulado. Caso contrário, ele é agrupado novamente e o processo de treinamento é repetido. O modelo ELM treinado é colocado nos nós de nevoeiro para detectar intrusões neste ambiente.

Além disso, alguns trabalhos propuseram abordagens completas de detecção de intrusão que aplicam técnicas para realizar o treinamento com a combinação de métodos supervisionados juntamente com técnicas não supervisionadas.

Ravi e Shalinie (2020) propuseram uma variante do algoritmo não supervisionado K-means com *Repeated Random Sampling*, juntamente com uma DNN supervisionada, para detectar intrusões em IoT. A abordagem realiza escolha de amostras aleatórias dos pontos de dados iniciais. Em seguida, avalia a distância euclidiana da instância de dados de entrada em relação às amostras aleatórias. O *cluster* da amostra que possui distância mínima para a instância de entrada é escolhido como saída. Este processo é repetido R vezes, cada vez pegando diferentes amostras aleatoriamente e a saída é agregada como uma lista. O resultado final é o valor do *cluster* que apareceu pelo menos $\lceil \frac{R}{2} \rceil$ vezes, se não houver nenhum *cluster*, então "?" é a saída. O *clustering* tem o poder de classificar dados desconhecidos. Dos vários tipos de agrupamento, o K-means pode convergir em uma taxa rápida para o ótimo local quando os pontos de dados iniciais refletem fortemente o agrupamento, e neste caso, quando o DNN é executado em cima dele, o modelo fornece pontos de dados iniciais quase perfeitos (Ravi; Shalinie, 2020). A abordagem, no entanto, foca apenas em detecção binária.

Além disso, os autores Abdel-Basset et al. (2021) trabalharam com treinamento de modelo usando uma combinação de um grande número de registros não rotulados e um pequeno número de registros rotulados. Uma abordagem inovadora de treinamento hierárquico

semi-supervisionado é introduzida, na qual os registros de tráfego não rotulados são divididos em várias partes em ordem consecutiva. Em seguida, os procedimentos de treinamento são realizados em cada parte separada em uma estratégia gradual que permite preservar os inter-relacionamentos sequenciais durante o treinamento. Neste caso, utilizam 75% dos dados de treinamento como não rotulados e o restante como amostras rotuladas. No entanto, ainda assim a abordagem apresentou dificuldades na detecção de alguns ataques.

Por fim, algumas abordagens trabalharam com métodos *ensemble*. *Ensemble learning* consiste em combinar vários modelos de classificadores básicos para melhorar o desempenho. Um classificador *ensemble* tende a ser mais preciso, por reduzir o risco de selecionar o classificador errado. Pois, através da combinação das suas predições é capaz de estabilizar o resultado (Dietterich, 2000). Várias técnicas de conjunto podem ser usadas, incluindo *Bagging*, *Boosting* e *Stacking*. *Bagging* (Breiman, 1996) é um método que gera diversas versões de classificadores básicos e os utiliza para obter um classificador agregado. As várias versões são formadas fazendo réplicas de *bootstrap* do conjunto de aprendizagem e usando-as como novos conjuntos de aprendizagem para cada versão. O elemento vital é a instabilidade do classificador base. Se perturbar o conjunto de aprendizagem pode causar mudanças significativas no classificador construído, então o *Bagging* pode melhorar a precisão (Breiman, 1996). *Boosting* é uma abordagem de aprendizagem em conjunto que reduz principalmente o preconceito e a variação na aprendizagem supervisionada. O objetivo é converter classificadores básicos em um classificador agregado. Porém, a combinação é realizada de forma diferente do *Bagging*, neste caso, cada classificador trabalha dentro das limitações dos classificadores anteriores. Desta forma, os classificadores base posteriores só são treinados ou testados nos casos em que os classificadores anteriores não conseguiram atingir uma precisão satisfatória (Bartlett et al., 1998). Finalmente, a estratégia *Stacking* considera vários modelos heterogêneos simples, treina-os em paralelo e combina-os para gerar um metamodelo para produzir uma previsão baseada nas previsões dos diferentes modelos fracos. Com base nessas estratégias, vários algoritmos de ensemble foram criados para classificação (Shafi et al., 2018; Illy et al., 2019; Alrashdi et al., 2019; Kumar; Gupta; Tripathi, 2020; Al-Khafajiy et al., 2021; Souza; Westphall; Machado, 2022; Alhowaide; Alsmadi; Tang, 2021).

O bom desempenho de detecção das abordagens depende da qualidade do treinamento realizado e da capacidade de aprendizagem do método. A grande maioria dos trabalhos encontrados no estado da arte tiveram como proposta abordagens baseadas em aprendizado supervisionado. Para realizar o treinamento dessas abordagens geralmente é necessário ter um grande número de dados rotulados (Ravi; Shalinie, 2020). Isso se torna um problema, pois é necessário capturar o tráfego de rede e rotulá-lo para serem usados no processo de treinamento do modelo de detecção. Portanto, devido a grande quantidade de dados necessários essa tarefa pode se tornar extremamente custosa. Alguns trabalhos propuseram abordagens promissoras, para lidar com o problema de rotulagem de dados de treinamento, usando métodos de amostragem e agrupamento de dados (Rathore; Park, 2018; An et al., 2018c; Ravi; Shalinie, 2020). Como pontos de estudos futuros, destacamos a necessidade de pesquisa em novas abordagens híbridas

baseados na aprendizagem supervisionada e na não supervisionada para contornar o problema da rotulagem de dados. Pois, a combinação de técnicas de aprendizado supervisionadas com outras técnicas capazes de trabalhar com dados não rotulados ou semi-rotulados pode ser uma ótima solução.

B.5.7 QPERS6 - Quais são as estratégias de colaboração empregadas nas abordagens de IDS distribuídos?

Nesta seção, discutimos abordagens distribuídas e colaborativas. Tradicionalmente, os IDS são implantados em um único dispositivo, porém, existem abordagens de detecção distribuídas e colaborativas, que permitem que um conjunto de detectores troque as informações necessárias buscando melhorar o desempenho da detecção (Li; Au; Wang, 2021). Diversos trabalhos propuseram abordagens com implantação distribuída e estratégias colaborativas buscando melhorias de desempenho (Rathore; Park, 2018; Alrashdi et al., 2019; Rahman et al., 2020; Ravi; Shalinie, 2020; Krishnan; Duttagupta; Achuthan, 2020).

Essas abordagens distribuídas que operam com compartilhamento de informações são conhecidas como Collaborative Intrusion Detection System (CIDS). Neste contexto, a computação em nevoeiro, como paradigma de computação distribuída, apresenta-se como um excelente framework para a implementação de abordagens CIDS, buscando melhor desempenho de detecção e eficiência no tempo de resposta (Li; Au; Wang, 2021).

Uma implementação de IDS distribuído pode ser realizada simplesmente compartilhando alertas entre dispositivos de borda (Lalouani; Younis, 2021). Li, Au e Wang (2021) apresentou uma estrutura CIDS baseada em fog para aumentar a eficiência da detecção de intrusões em redes inteligentes. A camada CIDS fornece monitoramento, detecção e diagnóstico de ameaças, trocando informações com outros nós, como alertas. Na arquitetura proposta por Azarkasb, Kashi e Khasteh (2021), cada dispositivo fog, além de realizar operações de detecção de intrusão, observa os nós ao seu redor, encontra o nó comprometido e informar os nós próximos a ele para se desconectarem desse nó. Em Aliyu et al. (2022), a lista negra e a distribuição de nós são usadas para bloquear dispositivos IoT comprometidos. Quando um dispositivo é detectado como malicioso, o nó do nevoeiro responsável informa outros para colocarem o nó malicioso na lista negra.

Algumas abordagens concentraram-se em enviar as intrusões detectadas para um servidor centralizado para gerar uma visão global do estado de segurança da aplicação IoT (Prabavathy; Sundarakantham; Shalinie, 2018; Samy; Yu; Zhang, 2020; Kumar; Gupta; Tripathi, 2020). Os resultados dos nós de nevoeiro são resumidos no servidor em nuvem para analisar e visualizar o estado atual de segurança do aplicativo IoT. Ele pode ser usado para prever a próxima ação do invasor usando abordagens de reconhecimento do plano do invasor e é útil para identificar ataques em vários estágios e ataques distribuídos com base nos resultados de nós de neblina distribuídos geograficamente.

Outras abordagens distribuíram tarefas de detecção e processamento pela estrutura IoT-

Fog-Cloud. Wang et al. (2018) propuseram uma estrutura colaborativa baseada em dispositivos edge e fog com foco na detecção baseada em regras e preservação da privacidade. A camada CIDN permite que vários nós IDS melhorem seu desempenho de detecção, trocando as informações necessárias entre si. A estrutura colaborativa de detecção de intrusão (COLIDE) (Arshad et al., 2019) permite a colaboração entre nós de borda e sensores com recursos limitados para detecção de intrusão eficaz e oportuna. Os eventos detectados pelos nós finais são reportados ao sistema centralizado, que opera na camada de computação em nuvem. Essa camada executa a correlação de alertas de vários hosts finais e aplica detecção baseada em anomalias nesses eventos correlacionados. O sistema de detecção proposto em Kumar et al. (2020) possui mecanismos distribuídos pelos dispositivos de neblina. Os dispositivos IoT enviam seus dados para o nó do cluster mais próximo que fornece serviço de nó de inteligência de borda. A inteligência de borda fornece armazenamento distribuído fora da cadeia e se conecta aos nós de neblina de maneira distribuída. Isso permite balanceamento de carga, gerenciamento de recursos, escalabilidade de redes e remove dados redundantes do tráfego de rede. Assim, proporciona uma forma distribuída e descentralizada de processamento dos dados, que são pré-processados e submetidos ao modelo de detecção. Além disso, em Kumar et al. (2022) os autores trabalham com integração de IDS com pool de mineração juntamente com trabalho de IDS distribuído, para detectar ataques DDoS em computação em nevoeiro habilitado para IA. Mourad et al. (2021) trabalhou com esquema de federações e propôs um esquema de Vehicular Edge Computing (VEC) habilitado para nevoeiro, permitindo descarregar tarefas de detecção de intrusão para nós de veículos federados próximos localizados dentro de nevoeiro veicular ad hoc para serem executados cooperativamente com latência mínima (Mourad et al., 2020). Gavel, Raghuvanshi e Tiwari (2021) propôs uma abordagem baseada em nós de nevoeiro distribuídos em uma infraestrutura IoT para detectar e relatar invasões. Depois que os dados são transmitidos dos dispositivos IoT para os dispositivos de neblina, eles são reportados a uma estação central por meio de um modelo de filtro. O modelo de filtro consiste em um modelo de redução de dimensionalidade de eixo duplo. Os dados agregados obtidos a partir da técnica de redução de dimensionalidade são utilizados para treinar o classificador, que é utilizado para detecção eficiente de intrusão na rede.

Outros trabalhos utilizaram a característica distribuída da computação em neblina para propor abordagens de treinamento distribuído entre dispositivos de neblina por meio do compartilhamento de informações (Sugi; Ratna, 2020; Diro; Chilamkurti, 2018c; Labiod; Korba; Ghoualmi, 2022; Kumar et al., 2020). Os autores Diro e Chilamkurti (2018c) propôs uma nova abordagem distribuída baseada em redes neurais para detectar invasões no ambiente IoT. A abordagem é implantada distribuída em dispositivos de neblina e possui dois níveis. Ele utiliza um dispositivo de neblina como mestre responsável por treinar e colocar o modelo nos demais nós de neblina. Os nós de segundo nível enviam atualizações de modelo para o nó mestre. O nó mestre atualiza o modelo global e espalha as atualizações para outros nós (Diro; Chilamkurti, 2018b). A abordagem proposta em Abdel-Basset et al. (2020) também possui estratégia similar No entanto, este nó primário pode ser considerado um ponto único de falha (SPOF), que é

mais fácil de comprometer do que uma abordagem de atualização de parâmetros baseada em nuvem. NG e Selvakumar (2020) também propõem uma estrutura com nó mestre de neblina para aprendizagem global e muitos nós de neblina funcionais para aprendizagem local. Cada nó de nevoeiro processa dados obtidos de dispositivos IoT conectados a ele. Após a conclusão do aprendizado, os resultados são armazenados no nó mestre de nevoeiro e usados para detecção de anomalias de tráfego IoT desconhecido. Além disso, esse processo faz com que o modelo compartilhe os melhores parâmetros obtidos com o aprendizado para evitar *overfitting* do framework proposto. Sahi, Soni e Auluck (2021) propôs uma abordagem de detecção de neblina com nó mestre e nós de trabalho. Os trabalhadores do nos fog fazem todo o treinamento e classificação, eles enviam os resultados de volta para o nó mestre. A função do nó mestre do nevoeiro é distribuir o modelo treinado e os dados para todos os nós de nevoeiro. No entanto, não são fornecidos detalhes sobre como ocorre a agregação dos modelos.

Neste contexto de treinamento distribuído de modelos de detecção, é importante discutir a Aprendizagem Federada (FL), que consiste em uma estratégia para formar um modelo global de ML a partir de vários modelos locais baseados em dados (Lalouani; Younis, 2021). Essa estratégia descentraliza o aprendizado de máquina, eliminando a necessidade de coletar dados em um único dispositivo. Em vez disso, o modelo é treinado por meio de múltiplas iterações em diferentes dispositivos. FL é uma estrutura ideal para agregar modelos distribuídos, preservando a privacidade e permitindo a convergência para um mecanismo de aprendizagem distribuído com precisão próxima à de uma implementação centralizada (McMahan et al., 2017). A arquitetura do framework proposta por Rey et al. (2022) é composta por clientes que monitoram dispositivos IoT e um servidor que coordena um processo de Aprendizagem Federada. A abordagem prevê detecção de intrusões por meio de modelos de redes neurais. Considerando que os dispositivos IoT geralmente possuem recursos limitados e confiabilidade modesta, os clientes responsáveis pelo treinamento dos modelos não são os dispositivos a serem protegidos, mas sim outras entidades capazes de coletar tráfego de dispositivos IoT presentes na mesma rede, como nós de nevoeiro (Rey et al., 2022). O servidor coordena esforços de treinamento para clientes federados e pode ser implantado na nuvem. Ele inicializa os pesos iniciais do modelo e os compartilha com todos os clientes, para que cada cliente comece com o mesmo modelo. Os parâmetros do modelo cliente k são referidos como w_k e os parâmetros do modelo global como w. Após receber os parâmetros de template atualizados de cada cliente, ou seja, $w_k : \forall_k \in [K]$, o servidor deve agregá-los para formar os novos parâmetros de template globais w. No trabalho, os autores utilizaram a função de agregação baseada na média, assim, a fórmula da função é dada por $w := \sum_{k=1}^{K} \frac{1}{K} w_k$. Com base nesta função, o trabalho também considera dois algoritmos de agregação, o Mini-lote e o Multi-época. Na agregação de minilote, o modelo é treinado com um único minilote de dados antes de enviá-lo ao servidor para agregação. Após a agregação feita pelo servidor, o cliente recebe então o novo modelo global agregado, com o qual o treinamento pode continuar. Este processo é repetido até que um número de épocas do conjunto de treinamento completo seja concluído. Por outro lado, na agregação Multi-Epoch, o modelo é treinado para todas as épocas de uma vez antes de ser enviado ao servidor para

agregação. Uma desvantagem potencial é que os modelos de média podem ter resultados arbitrariamente ruins devido à não convexidade do objetivo. Este problema é muito mais provável para agregação de múltiplas épocas, pois os modelos são treinados separadamente por muito mais tempo antes de serem agregados (Rey et al., 2022). A abordagem FLIDS (Lalouani; Younis, 2021) também emprega aprendizagem federada para permitir agregação distribuída que preserva a privacidade. Dispositivos de borda $\Psi_i = (\psi_1, \psi_2, ..., \psi_n)$ treinam seus modelos locais para determinar pesos locais e valores de perda. Esta informação é então enviada para os nós $\Phi_i = (\phi_1, \phi_2, ..., \phi_n)$ fog responsáveis pela agregação. O modelo agregado é então enviado de volta aos dispositivos para melhorar o modelo usando alertas de segurança coletados. O objetivo da otimização é minimizar a perda geral $F(w^f)$, que pode ser expressa como a minimização de $[F(w^f) - F(w^*)]$, onde w^f representa os pesos finais após muitas etapas de agregação iterativas. Neste processo, o servidor central passa inicialmente o vetor de parâmetro inicial global w_0 para todos os dispositivos para sincronização. Os pesos no nó ψ_i são inicialmente $w_i = w^0$. Então o dispositivo local $\psi_i \in \Psi$ atualiza sua função de perda $F_i(w)$ e atualiza seus pesos locais. O servidor então agrega todas as atualizações do dispositivo para melhorar o modelo global calculando a média dos pesos locais. A agregação dos modelos individuais formará o modelo global e permitirá o cálculo da função de perda global. O servidor então atualiza o modelo global e envia de volta os novos pesos aos dispositivos e assim por diante. Um ponto de atenção é que a precisão dos modelos locais pode ser degradada maliciosamente na esperança de produzir um IDS agregado impreciso. Este comportamento malicioso pode ser impulsionado por um ou vários participantes do conluio (Lalouani; Younis, 2021). Este tipo de ameaça levanta preocupações, é necessário ser capaz de determinar o subconjunto de nós coniventes que estão diminuindo a taxa de detecção de IDS. O problema é combinatório e acaba sendo NP-difícil (Lalouani; Younis, 2021). A FLIDS busca identificar tentativas individuais de envenenamento de dados e conluio por meio de um mecanismo de defesa robusto.

B.5.8 QPERS7 - Quais são as principais abordagens para executar contramedidas no contexto da IoT e da computação em neblina?

Nesta seção são apresentados os mecanismos de contramedidas propostos nas abordagens do estado da arte, para bloquear os ataques detectados. É importante ter mecanismos para realizar contramedidas para bloquear e evitar que a intrusão seja bem-sucedida. Entre as ações existentes está a emissão de alertas para o gerente da rede, para apenas informar o gestor da ocorrência de uma intrusão. Esta é considerada uma ação de pós-detecção passiva. Outra classe de abordagens de pós-detecção é a ativa, onde as ações realizadas têm como objetivo interromper um ataque em andamento e, em seguida, bloquear o acesso do invasor (Bace; Mell, 2001). Os IDSs que executam contra-medidas ativas são conhecidos como Sistemas de Prevenção de Intrusão (*Intrusion Prevention Systems* - IPS) (Birkinshaw; Rouka; Vassilakis, 2019). A seguir são apresentadas diversas abordagens de contramedidas ativas propostas por trabalhos do estado da arte, conforme pode ser observado na Tabela 52.

Tabela 52 – Tipos de mecanismos de contramedidas encontrados nas abordagens do estado da arte.

Tipo de mecanismo de contramedida	**Trabalhos**
SDN	[19], [30], [34], [44], [65], [98]
Bloqueio de conexões e descarte de pacotes	[6], [22], [63], [106]
Honeypot	[7], [21]
Teoria dos jogos	[11], [64], [20]
Leituras ponderadas	[5], [23]
Bloqueio dos pacotes por *firewall*	[29], [37]
Blacklist	[54], [93], [106]
Reinicialização do sistema e mudança de configurações	[31], [46]
Medidas preventivas de restrições	[32], [33]
Elliptic Curve Cryptography (ECC)	[33]
Advanced Encryption System (AES)	[9]
Filtro MAC-IP	[49]

Foram encontrados trabalhos no estado da arte que abordaram contramedidas para um contexto, protocolo ou ataque específico. Em Potrino, Rango e Santamaria (2019), Potrino, Rango e Fazio (2019), foram propostas abordagens para mitigar ataques específicos contra o protocolo MQTT. São aplicadas um conjunto de restrições preventivas e políticas de descarte de pacotes de acordo com as especificações definidas pelos autores.

Alguns trabalhos abordaram o uso de criptografia para mitigar certos tipos e ataques. Aliyu, Sheltami e Shakshuki (2018) utilizaram o *Advanced Encryption Syste*m (AES), técnica de criptografia simétrica, enquanto a chave de criptografia é trocada usando a troca de chaves *Diffie-Hellman*, para evitar que as mensagens sejam modificadas por um ataque *Man-in-the-Middle* (MITM). No entanto, não são fornecidos detalhes sobre o processo de bloqueio do nó detectado como malicioso. Além disso, Potrino, Rango e Fazio (2019) destaca que a utilização do protocolo *Secure Socket Layer/Transport Layer Security* (SSL/TLS) pode consumir uma quantidade significativa de energia em um dispositivo IoT. Já os criptossistemas de curva elíptica sobre campo finito têm alguns benefícios como o tamanho da chave, que pode ser consideravelmente menor em comparação com criptossistemas adicionais como RSA e *Diffie-Hellman*. Portanto, nesse caso, os autores propuseram criptografar os *payloads* MQTT usando *Elliptic Curve Cryptography* (ECC) para mitigar data *tampering* and *eavesdropping*. ECC é um tipo de cifra assimétrica baseado na aritmética de pontos em uma curva elíptica (Potrino; Rango; Fazio, 2019).

Yaseen et al. (2017), Yaseen et al. (2018) propuseram mecanismos para ponderar as leituras de dispositivos sensores e mitigar ataques de conluio. A abordagem calcula os valores de confiabilidade e atribui pesos às leituras de cada dispositivo IoT que pertencem ao *cluster* de nó de nevoeiro correspondente. Os valores de confiabilidade são usados para calcular a média ponderada das leituras para todos os dispositivos no *cluster*.

Alguns trabalhos propuseram abordagens que realizam o descarte de pacotes e o blo-

queio de conexões quando um ataque é detectado. A abordagem SDRK (Ravi; Shalinie, 2020) bloqueia pacotes identificados como maliciosos usando um filtro IP-MAC. Quando um ataque é detectado o módulo de mitigação é acionado e uma regra de descarte é definida no nevoeiro para o par de endereços MAC-IP do dispositivo que enviou os pacotes. Além disso, um temporizador aleatório é definido. Depois que o cronômetro expira, a regra de descarte é revogada. O cronômetro deve ser aleatório para que o invasor não possa deduzir a estratégia de tempo de mitigação (Ravi; Shalinie, 2020). Se o pacote for detectado como "?", a regra de descarte é definida para um tempo aleatório relativamente pequeno porque o modelo não tem certeza da classe de tráfego. No entanto, não são fornecidos maiores detalhes de como o filtro é implementado. A abordagem proposta por Tu et al. (2018) rejeita pacotes de um determinado canal de comunicação entre o dispositivo e o nevoeiro quando detecta um ataque de personificação. Em Pan, Pacheco e Hariri (2017) é proposto uma abordagem com medidas de mitigação de ataques no contexto de automação de prédios e computação em nevoeiro. Assim que um alerta de anomalia é recebido, o módulo extrai o mecanismos de ataque, alvo e peso da importância de cada ativo. Eles utilizaram Tabelas de Decisão para classificar os fluxos de pacotes anômalos com base nas características de mecanismo de ataque e ativo de destino. O algoritmo gera uma tabela de decisão candidata agrupando e contando tuplas no conjunto de dados de treinamento e, então a poda, comparando a confiança de cada regra no grupo com o limiar de referência (Pan; Pacheco; Hariri, 2017). Durante a fase de treinamento, as duas características de ataque usadas na classificação foram adicionadas ao conjunto de matrizes contextuais de anomalias e, em seguida, treinados com os algoritmos de tabelas de decisão. Ações de proteção são acionadas com o objetivo de manter a disponibilidade e o sigilo da rede. Essas ações são determinadas usando um conjunto de políticas predefinidas. Dentro das políticas de ação, o mecanismo de ataque é o fator chave para escolher a ação de proteção apropriada. As ações de proteção incluem descartar pacotes, suspender a conexão usada pelo fluxo de tráfego e descartar o tráfego invocado dos ativos de anomalia. A fase final da operação do módulo é traduzir o alerta, bem como a decisão de ação tomada na fase anterior, em uma mensagem de alarme compreensível. Os autores Lawal, Shaikh e Hassan (2020) também apresentam um *framework* que conta com um módulo de bloqueio e descarte de pacotes identificados como intrusivos, além do envio de mensagem de alerta. No entanto, não são apresentados detalhes de como as ações dessas abordagens são implementadas.

Outros trabalhos utilizaram *firewalls* para bloquear e filtrar pacotes e acessos. Na abordagem FOCUS (Maharaja; Iyer; Ye, 2019), o *firewall* é usado para bloquear o acesso ao servidor de Rede Privada Virtual (*Virtual Private Network* - VPN) que protege a comunicação entre o nevoeiro e os dispositivos IoT. Ele permite que solicitações de tráfego confiável acessem o servidor VPN se não houver objeções à lista de controle de acesso do *firewall*, ao mesmo tempo em que rejeita as não confiáveis de contatar o servidor VPN. Zhou, Guo e Deng (2019) propôs uma abordagem para mitigação de DDoS em ambiente IIoT. Os *firewalls* empregados executam as funções de filtragem de pacotes com base em regras, enquanto os servidores locais e em nuvem conduzem a funcionalidade de detecção de anomalias. Uma vez que um ataque DDoS é

detectado, as assinaturas de ataque podem ser extraídas, transmitidas e atualizadas através dos *firewalls* para filtrar pacotes de ataque perto de suas fontes de forma eficaz. O *firewall* também executa filtragem reversa que detecta e bloqueia o tráfego de ataque de dispositivos de campo comprometidos para camadas superiores. Além disso, o *firewall* bloqueia os pacotes de ataque de acordo com um conjunto de regras pré-configuradas. Eles podem ser construídos com base em endereços IP, protocolos ou portas permitidas. Qualquer pacote que viole as regras predefinidas, ou não seja relevante, será filtrado.

A abordagem proposta pelos autores Sandhu, Sohal e Sood (2017), Sohal et al. (2018) utiliza *Honeypot*. Ela consiste em uma armadilha criada pelo administrador do sistema, uma réplica do ambiente real para atrair intrusos ao sistema. Portanto, quando um dispositivo é detectado como malicioso, ele é enviado para a *honeypot*. Na *honeypot*, é possível simular o comportamento real do sistema e registrar a rota de ataque do atacante. Além disso, quando um sensor é movido para a *honeypot* por engano, é possível calcular a probabilidade de que o dispositivo tenha sido movido por engano. Se necessário, restaure-o à operação normal. Como tal, a abordagem é resiliente a falsos positivos.

Zahra e Chishti (2020) propuseram uma abordagem para responder às intrusões detectadas com base em quão prejudiciais elas podem ser. Essa decisão é tomada por uma base de conhecimento difusa. Dependendo das prioridades de uma organização, ela atribuirá peso às regras e dará uma decisão sobre a resposta do sistema. Nesse contexto, foi proposto como regra para a mitigação de ataques de conluio e DDoS, a inserção do nó detectado como intrusivo na *blacklist* mantida na nuvem e a geração de alertas à toda a rede para bloqueio de comunicação com o nó detectado.

Abordagens dos autores (Pacheco; Benitez; Félix-Herrán, 2019; Pacheco et al., 2020) apresenta como contra-medidas o reinício de conexões e sistema e até mesmo alterações de configuração. A abordagem depende de níveis de ação. Inicialmente, a ação de reinicialização da conexão é usada. Se o problema persistir sempre que a conexão for renovada, outras ações serão necessárias. O manipulador de ações emprega um arquivo de *log* para manter um registro de cada erro, incluindo seu carimbo de data/hora. Antes de aplicar qualquer política de proteção, o arquivo de *log* é revisado, procurando a frequência de um determinado erro. Se uma alta frequência for encontrada, ações são realizadas para alterar a configuração do sistema e solicitar autenticação.

Alguns trabalhos propuseram abordagens baseadas na Teoria dos Jogos. Os autores An et al. (2018a) propuseram uma estratégia, baseada no esquema FC-IDS proposto em An et al. (2018c), para responder à intrusão. O processo de mitigação de uma intrusão detectada e descrita como um modelo matemático baseado na teoria do jogo diferencial. De acordo com este modelo, a estratégia de resposta ideal é obtida correspondendo à estratégia de intrusão ideal. Shen et al. (2018b) propuseram uma infraestrutura de detecção de malware que usa um jogo de correspondência para divulgar as interações entre objetos inteligentes e ou não os novos correspondentes devido à incerteza de malware para objetos inteligentes. O sistema de resposta toma decisões de contra-ação: Conceder ou Prevenir. Uma primeira decisão permite

que o objeto inteligente trabalhe continuamente com a rede IoT à qual pertence. Outra decisão atua gradativamente com contra-ações para o objeto inteligente, como remoção de privilégios, redução do número de recursos, inserção em uma lista de malware e interrupção da sessão.

Por fim, alguns trabalhos encontrados apresentaram resultados promissores na pesquisa de abordagens baseadas em Redes Definidas por Software (*Software Defined Networks* - SDN) para a mitigação de intrusão em ambientes de computação em nevoeiro e IoT. A SDN se baseia na separação de planos de controle e planos de dados. Essa separação permite que o controlador seja facilmente modificado e atualizado para alterar todo o comportamento da rede. O controlador é programável e elimina a necessidade de configuração prévia do dispositivo físico da rede, tornando-o reutilizável (McKeown et al., 2008). Nas abordagens encontradas no estado da arte, geralmente, quando o controlador SDN recebe um pacote da rede IoT, ele o envia para o módulo de análise. Se o pacote for identificado como não intrusivo, o controlador SDN libera o tráfego. Se o pacote for identificado como intrusivo, o controlador SDN implementa contramedidas por meio de regras na tabela de fluxo (Shafi et al., 2018; Nguyen et al., 2019; Priyadarshini; Barik, 2022; Miranda et al., 2020). Os autores Shafi et al. (2018) propuseram uma estrutura de mitigação de intrusão através de SDN para redes IoT. Quando são detectadas ameaças o controlador SDN é invocado para instalar dinamicamente medidas preventivas. O controlador SDN instala regras de fluxo apropriadas nos *switches* da borda da rede para controlar a interrupção do tráfego de ataque. O sistema proposto satisfaz os requisitos de escalabilidade, é possível adicionar mais recursos computacionais no caso do grande aumento no número de dispositivos de IoT. Priyadarshini e Barik (2022) também propuseram abordagem com SDN. Neste caso, para implantar um módulo defensor de DDoS no nevoeiro. A responsabilidade da SDN é gerenciar toda a rede, mantendo a tabela de encaminhamento. Além disso, quando um pacote é detectado como intrusivo todas as suas informações são enviadas ao controlador SDN. O controlador, executa ações para impedir que esse pacote entre na rede, como bloquear o endereço IP na tabela de fluxo. Do mesmo modo, Miranda et al. (2020) apresentaram estrutura de prevenção baseada em rede definida por software. A abordagem *SeArch* (Nguyen et al., 2019) apresenta uma arquitetura composta por um arranjo hierárquico de três camadas. Os controladores SDN, responsáveis pelo gerenciamento da rede, são implantados na camada de nevoeiro. Estas camadas trabalham em colaboração para detectar anomalias e formular políticas nos dispositivos de *gateway* IoT baseados em SDN, para interromper o tráfego malicioso o mais rápido possível. Se um ataque for detectado, são instaladas regras de fluxo no *gateway* IoT baseado em SDN afetado, para uma reação rápida aos ataques. No caso de ataques de negação de serviço, a política padrão é a implementação de uma regra e a definição de um valor de *hard_timeout* para descartar todos os pacotes da fonte de ataque. Além disso, a solução proposta oferece soluções baseadas em balanceamento de carga para suportar o grande volume de trafego gerado por ataques. A abordagem apresenta grande esforço na definição de contramedidas para ataques DoS.

Finalmente, Rangisetti, Dwivedi e Singh (2021) propôs uma abordagem baseada em SDN para evitar falsificação de protocolo de resolução de endereço (ARP) em plataformas

Cloud-Fog-Edge. Usuários mal-intencionados internos podem realizar ataques de falsificação de ARP explorando ambientes de rede compartilhados. Principalmente, os ataques de falsificação de ARP podem levar a ataques DoS e DDoS, MITM e sequestro de sessão de rede. A abordagem evita ataques de falsificação de ARP, reduzindo a sobrecarga em controladores centralizados e switches OpenFlow. Krishnan, Duttagupta e Achuthan (2020) também propôs arquitetura com SDN. Uma vez identificado um ataque, o plano de controle elabora um conjunto de estratégias para reagir, habilitando inicialmente os atuadores de defesa no plano de dados. Quando os tipos de ataque e os locais de origem de um ataque DDoS são rastreados, o plano de controle reage carregando atuadores de defesa específicos em dispositivos de plano de dados direcionados. Quando o tipo de ataque gira e os switches afetados no caminho do ataque e na rede de origem são determinados, instruções são enviadas a esses switches com ações de limpeza, que bloquearam o tráfego de ataque upstream e eliminaram fluxos maliciosos do switch.

B.5.9 QPERS8 - Quais estratégias de avaliação são usadas para validar abordagens de detecção?

Nesta seção, são apresentados detalhes sobre as estratégias de avaliação utilizadas pelos trabalhos. Além disso, são discutidos os principais conjuntos de dados utilizados na avaliação de abordagens de detecção. O objetivo foi realizar um levantamento das estratégias de validação utilizadas no estado da arte e dos conjuntos de dados atuais que possam servir de base para futuros pesquisadores, fornecendo indicações que possam auxiliar na decisão sobre quais conjuntos de dados são mais adequados para o contexto de suas respectivas pesquisas.

Apenas o trabalho dos autores Wang et al. (2018) fez uma avaliação em ambiente real. Eles colaboraram com um provedor de serviços de Tecnologia da Informação para validar o desempenho da abordagem proposta em um ambiente de rede real, onde várias métricas de confiabilidade foram obtidas a partir dos experimentos.

A maioria dos trabalhos incluídos no mapeamento realizaram avaliação através de simulação, até mesmo a abordagem de Wang et al. (2018) que conforme supracitado, realizou também a avaliação em ambiente real. Dentre as ferramentas utilizadas para simulação destacam-se o *Matlab* [3], o qual foi utilizado em cinco estudos para os experimentos de simulação (An et al., 2018a; An et al., 2018b; Shen et al., 2018b; Miranda et al., 2020; Pirozmand et al., 2020). O *Omnet++* [4] were performed in three studies (Aliyu; Sheltami; Shakshuki, 2018; Potrino; Rango; Santamaria, 2019; Aliyu et al., 2022; Paranjothi; Atiquzzaman, 2021). A ferramenta *Cloudsim* [5] foi utilizada em dois trabalhos (Yaseen et al., 2017; Yaseen et al., 2018). O simulador COOJA [6] foi utilizado em três trabalhos (Arshad et al., 2019; Zahra; Chishti, 2020;

[3] mathworks.com/products/matlab.html
[4] omnetpp.org/
[5] cloudbus.org/cloudsim/
[6] contiki-os.org/

Zahra; Chishti, 2022). A ferramenta *CloudExp* foi utilizada em dois trabalhos (Jararweh et al., 2014; Yaseen et al., 2017). Os autores Yaseen et al. (2016) utilizaram o simulador NS-2 [7]. As ferramentas iCanCloud [8] e Castalia [9] foram utilizadas em um trabalho cada. Além disso, alguns trabalhos realizaram experimentos em estruturas construídas em ambiente laboratorial (Pan; Pacheco; Hariri, 2017; Sandhu; Sohal; Sood, 2017; Sohal et al., 2018; Shafi et al., 2018; Tu et al., 2018; Maharaja; Iyer; Ye, 2019; Nguyen et al., 2019; Pacheco; Benitez; Félix-Herrán, 2019; Potrino; Rango; Fazio, 2019; Priyadarshini; Barik, 2022; Zhou; Guo; Deng, 2019; Mourad et al., 2020; Pacheco et al., 2020; Ravi; Shalinie, 2020; Samy; Yu; Zhang, 2020).

Outra estratégia bastante utilizada para avaliar abordagens de detecção de intrusão, principalmente aquelas baseadas em aprendizado de máquina, é o uso de conjuntos de dados compostos por exemplos de tráfego normal e intrusivo. A Tabela 53 apresenta os principais bancos de dados usados nos problemas de detecção de intrusão. Além disso, as principais características dos conjuntos de dados são apresentadas a seguir.

O conjunto de dados KDD Cup 99 (KDD Cup 1999 Data, 1999) foi utilizado para avaliar 6 trabalhos. Ele é composto por ocorrências de ataques e atividades normais capturadas em redes de computadores (Tavallaee et al., 2009). Cada exemplo da base KDD Cup 99 possui 42 atributos, dos quais 41 são características comportamentais da rede e são extraídos dos pacotes capturados durante sua atividade. Esse conjunto de dados cobre as características básicas obtidas diretamente dos pacotes, como protocolo, portas e sinalizadores usados, bem como dados de conteúdo do pacote, medidas estatísticas e atributos baseados em tempo. Os ataques presentes na base podem ser agrupados em DoS, acesso não autorizado de uma máquina remota, acesso não autorizado ao root e sondagem. Apesar de ser uma base amplamente utilizada em pesquisas, estudos observaram que ela possui algumas deficiências, como amostras redundantes (Tavallaee et al., 2009). Além disso, é muito antiga para representar o ambiente de rede atual (Liu; Lang, 2019b).

O banco de dados CAIDA (Center for Applied Internet Data Analysis, 2007) foi usado em (Nguyen et al., 2019). Consiste em aproximadamente uma hora de tráfego de dados, correspondendo a ataques DDoS. A base de dados é composta por um grande volume de dados, o que ilustra a força de ataques desta natureza. Todo esse tráfego foi gerado distribuído, com foco em um único alvo para ocupar todos os recursos disponíveis no servidor vítima. Pode-se citar como desvantagem desta base a ausência de uma variedade de ataques ou pacotes referentes a atividades consideradas normais (Khraisat et al., 2019).

Para resolver os problemas mencionados acima da base de dados KDD CUP 99, os pesquisadores Tavallaee et al. (2009) propuseram um novo banco de dados chamado NSL-KDD. Ele foi construído com registros selecionados do conjunto de dados KDD CUP 99 completo, mantendo os atributos e otimizando a redundância da amostra. Portanto, o banco de dados possui a mesma estrutura em termos de atributos em comparação ao KDD Cup 99. No entanto,

[7] ns2simulator.com/
[8] icancloud.org
[9] github.com/ boulis/Castalia

Tabela 53 – Conjunto de dados utilizados para avaliação de abordagens de detecção de intrusão.

Dataset	Atr.	Rót.	IoT	Comentários	Trabalhos
KDD Cup 99	42	✓	x	Problemas de redundância.	[1], [3], [8], [10] [16], [30], [36], [61]
CAIDA		x	no	Sem diversidade de ataques.	[30]
NSL-KDD	42	✓	x	Reduz os problemas de da KDD Cup 99, mas é muito antigo para representar os padrões de rede atuais.	[14], [15], [17], [18] [25], [27], [38], [39], [42], [49], [50], [51] [52], [53], [55], [67] [71], [73], [77], [78] [79], [80], [81], [88] [91], [100], [102], [106]
ISCX IDS 2012	19	✓	x	O tráfego não é representativo atualmente.	[1], [13], [34], [39]
ADFA	80	✓	x	Focado apenas em dados de host.	[28], [85], [94]
CTU-13 Botnet	33	✓	x	Focado apenas em ataques de botnet.	[34], [47]
UNSW-NB15	49	✓	✓	Não inclui alguns recursos específicos de IoT.	[19], [30], [38], [40] [41], [52], [57], [59] [71], [74], [75], [80] [88], [91], [93], [100]
AWID	155	✓	x	Possui baixa variedade de ataques.	[13], [48]
RPL-NIDS17	21	✓	✓	Tem foco em ataques de roteamento, no entanto, não aborda outros ataques comuns em IoT.	[52]
CICIDS-2017	80	✓	x	Não inclui alguns recursos específicos de IoT.	[35], [52], [53], [62], [68] [79], [89], [91], [102]
CIDDS-001	10	✓	x	Não é específico IoT.	[90]
CICIDS-2018	80	✓	x	Não inclui alguns recursos específicos de IoT.	[68], [73]
N-BaIoT	115	x	✓	Focado apenas em ataques de *botnet*.	[52, 105]
DS2OS	13	✓	✓	Dataset IoT recente	[57], [92]
ToN-IoT	7	✓	✓	Conjunto de dados recente focado em IoT.	[69], [70], [76] [84], [97]
Bot-IoT	46	✓	✓	Dataset IoT recente. Baixa variedade de ataques.	[40], [43], [58], [63] [76], [87], [102], [104]
CICDDoS-2019	80	✓	x	Focado apenas em DDoS.	[72]
IoT-23	21	✓	✓	Tráfego real IoT. Baixa variedade de ataques.	[107]
IoTID20	12	✓	✓	Dataset IoT recente.	[82], [102]
MQTT-IoT-IDS2020	44	✓	✓	Dataset IoT recente. Apenas protocolo MQTT.	
MQTTset	33	✓	✓	Dataset IoT recente. Apenas protocolo MQTT.	
NetFlow Datasets	43	✓	✓	Dataset recente. Possui tráfego no contexto IoT. Padronização de atributos.	[84], [107]

este conjunto de dados possui apenas 125,973 registros. Esta base de dados é extremamente relevante em detecção de intrusão e foi a mais utilizada pelos trabalhos levantados na revisão.

No entanto, conforme mencionado acima, esta base de dados também é muito antiga e apresenta dificuldades para representar o ambiente de rede atual, ainda mais quando se considera o contexto da IoT.

O conjunto de dados ISCX IDS 2012 (Shiravi et al., 2012) foi utilizado em 4 trabalhos. Ele foi construído a partir de dados capturados em 2010 ao longo de uma semana, incluindo atividade normal e ataques de infiltração, DoS, DDoS e força bruta contra o protocolo *Secure Socket Shell* (SSH). Os dados que compõem a base foram capturados em ambientes de rede com tráfego real e são compostos por 19 atributos relacionados às características do tráfego. Como nas outras bases antigas, os tráfegos presentes nesta base não são mais representativos do tráfego atual.

Os conjuntos de dados ADFA (Creech, 2014; Creech; Hu, 2013) são conjuntos de dados que cobrem os dados capturados dos sistemas operacionais Linux e Windows. Eles são projetados para avaliação HIDS com base em chamadas de sistema. Os bancos de dados contêm milhares de rastreamentos de chamadas do sistema coletados em vários cenários para simular circunstâncias da vida real. Eles consistem em uma coleção abrangente de rastreamentos de chamadas do sistema que representam vulnerabilidades e ataques recentes no nível do sistema. Estes conjuntos foram utilizados em 3 trabalhos. Um dos principais motivos para isso é que ela não considera as características do tráfego de rede.

O conjunto de dados CTU-13 (García et al., 2014) foi utilizado por 2 trabalhos. Esse conjunto consiste em treze capturas, chamadas de cenários, de diferentes amostras de *botnets*. Em cada cenário, foi executado um *malware* específico, que usou vários protocolos e executou ações diferentes. Apesar dessa variedade, a base se concentra apenas em ataques de botnet.

A base de dados UNSW-NB15 (Moustafa; Slay, 2015) foi utilizada por 16 trabalhos. O conjunto de dados foi compilado pela *University of South Wales*, onde os pesquisadores configuraram três servidores virtuais para capturar o tráfego da rede. A base de dados possui 2.540.044 registros. Cada registro possui 49 atributos, incluindo o representante da classe. Esses atributos podem ser categorizados em seis grupos: recursos básicos, recursos de fluxo, recursos de tempo, recursos de conteúdo, recursos adicionais gerados e recursos rotulados. A base possui eventos de 9 categorias de ataques: *Fuzzers*, *Analysis*, *Backdoors*, DoS, *Exploits*, *Generic*, *Reconnaissance*, *Shellcode* e *Worms*. De acordo com Moustafa (2019) este conjunto de dados não inclui recursos IoT específicos.

A base de dados AWID2 (Kolias et al., 2016) foi utilizada por 2 trabalhos. Esta base é baseada em eventos intrusivos e não intrusivos presentes em redes *wireless*. A arquitetura da rede monitorada foi baseada na família de redes IEEE 802.11 contando com aparelhos celulares e notebook de diversas marcas. A base AWID possui 4 classes principais, eventos normais, ataques *Flooding*, *Injection* e *Impersonation*. Sua principal fraqueza é a falta de variedade de ataques.

O conjunto de dados RPL-NIDS17 (Verma; Ranga, 2019) foi desenvolvido com a coleta do tráfego de simulações de diferentes ataques de roteamento contra o protocolo de roteamento IPv6 para redes de baixa potência e perdas (*IPv6 Routing Protocol for Low Power*

and Lossy Networks - RPL). A base tem 21 atributos e inclui ataques como encaminhamento seletivo, ataque de ID de clone, ataques *Sybil*, *Sinkhole*, *Hello Flood*, *Local repair* e *Black hole*. Portanto, a base tem o diferencial de trazer diversos ataques de roteamento extremamente comuns em redes IoT (Verma; Ranga, 2019).

O banco de dados CICIDS2017 é outra base importante para avaliar abordagens de detecção de intrusão. Foi gerado em 2017 pelo *Canadian Institute for Cybersecurity* (CIC). A base CICIDS2017 corresponde a registros extraídos de pacotes de rede capturados durante a comunicação do computador, onde diferentes tipos de ataques foram realizados. Ele possui dados sobre o comportamento abstrato de 25 usuários com base nos protocolos *Hyper Text Transfer Protocol* (HTTP), *Hyper Text Transfer Protocol Secure* (HTTPS), *File Transfer Protocol* (FTP), SSH e *email*. Além do tráfego normal, vários ataques são realizados, incluindo FTP de força bruta, SSH de força bruta, DoS, *Heartbleed*, ataque da web, infiltração, *botnet* e DDoS (Sharafaldin; Lashkari; Ghorbani, 2018). No entanto, não é um conjunto de dados focado na representação do tráfego de IoT.

CIDDS-001 (Ring et al., 2017) é um conjunto de dados rotulado baseado em fluxo para avaliar sistemas de detecção de intrusão de rede baseados em anomalias. O tráfego contém ataques na forma de DDoS, *Port-Scan* e *Brute Force*. O conjunto de dados foi construído a partir da coleta do tráfego de rede. O conjunto de dados consiste em 10 atributos e 5 classes. O número total de ocorrências tomadas é de 180.387, onde os registros normais são 147.073 e os ataques são 33.313.

A base CICIDS2018 foi concebida através de um projeto colaborativo entre o *Communications Security Establishment* (CSE) e o CIC (Sharafaldin; Lashkari; Ghorbani, 2018). É composto por 16233002 registros, com 80 atributos relacionados às características do tráfego capturado durante dez dias em uma extensa rede de computadores. A base tem um conjunto semelhante de ataques em comparação com a versão de 2017, com 14 ataques.

O conjunto de dados N-BaIoT foi desenvolvido para validar abordagens de detecção de ataques de*botnet*usando técnicas de detecção de anomalias. O conjunto de dados contém o tráfego real coletado de nove dispositivos IoT comerciais, incluindo a campainha *Danmini*, o termostato *Ecobee* e a câmera de segurança *Provision PT-737E*. Os autores comprometeram os dispositivos IoT usados em sua mesa de teste com ataques de *botnet Mirai* e BASHLITE. N-BaIoT contém cinco ataques BASHLITE (varredura, lixo, inundação *User Datagram Protocol* (UDP), inundação *Transmission Control Protocol* (TCP) e COMBO) e cinco ataques Mirai (varredura, inundação Ack, inundação Syn, inundação UDP, UDPplain) (Meidan et al., 2018). O conjunto de dados é recente e extremamente integrado ao contexto da IoT. No entanto, ele se concentra apenas em ataques de *botnet*.

O conjunto de dados DS2OS contém rastreamentos capturados no ambiente DS2OS IoT. Eles são da camada de aplicativo, portanto, bastante diferentes dos rastreamentos de rede convencionais. O conjunto de dados principal foi coletado por um dia. Os dados da IoT foram capturados de controladores de luz, termômetros, sensores de movimento, máquinas de lavar, baterias, termostatos, portas inteligentes e *smartphones*. A base consiste em apenas 12 atributos

de tráfego (Aubet, 2018).

ToN-IoT é um conjunto de dados realista coletado de uma rede heterogênea de grande escala (Kumar; Gupta; Tripathi, 2020; Alsaedi et al., 2020). Diferentes técnicas de *hacking*, como DoS, DDoS e *ransomware*, foram lançadas contra aplicativos da web, *gateways* IoT e sistemas de computador em uma rede IoT. Este conjunto de dados apresenta dados capturados de uma estrutura baseada em computação de nevoeiro e IoT. O banco de dados funciona com menos de 10 atributos e tem ataques como sondagem, DoS, DDoS, *ransomware*, *backdoor*, injeção de dados, *Cross-site Scripting* (XSS), ataque de quebra de senha, ataques MITM (Moustafa, 2019).

O conjunto de dados BoT-IoT foi criado projetando um ambiente de rede realista no *Cyber Range Lab* do *UNSW Canberra Cyber center*. O ambiente incorpora tráfego normal e ataques como DoS, DDoS, sondagem e ataques de roubo. A base possui 46 atributos referentes às características do tráfego. Este conjunto é extremamente recente e realista para o tráfego atual. No entanto, ele não possui outros tipos de ataques presentes no contexto de IoT (Koroniotis et al., 2019).

A base CICDDoS2019 apresenta tráfego normal e ataques DDoS atualizados que se assemelham a dados reais do mundo real. Apresenta 80 atributos relacionados às características do tráfego capturado durante dois dias (Sharafaldin et al., 2019). O principal ponto fraco dessa base é a falta de outras categorias de ataques.

O conjunto de dados IoT-23 é um dos mais recentes e consiste em um grande conjunto de dados relacionado ao tráfego de ataques de *botnets* em um ambiente IoT. Uma das vantagens deste conjunto de dados é que ele foi capturado considerando dispositivos do mundo real, em vez de dispositivos simulados(Garcia; Parmisano; Erquiaga, 2020).

O conjunto de dados IoTID20 possui dados referentes ao tráfego de rede de dispositivos IoT e estruturas interligadas típicas de uma *Smart Home*. Um dos dispositivos utilizados é uma câmera de segurança, por exemplo. Uma variedade interessante de ataques é registrada, envolvendo *DoS*, *Botnet Mirai*, MITM e atividades de sondagem (Ullah; Mahmoud, 2020).

O conjunto de dados MQTT-IoT-IDS2020 concentra-se em um dos protocolos de comunicação máquina a máquina da IoT, o *Message Queuing Telemetry Transport* (MQTT). A estrutura monitorada conta com 12 sensores e um corretor. Ataques como sondagem, força bruta SSH e força bruta MQTT estão presentes neste conjunto de dados (Hindy et al., 2021).

O conjunto de dados MQTTset também apresenta tráfego capturado de uma rede IoT baseada no protocolo MQTT. Foram monitorados diferentes sensores IoT conectados a 1 *broker*. A partir disso, vários tráfegos normais e intrusivos relacionados a ataques DoS, *MQTT Publish Flood*, dados malformados, *Brute Force Authenticate*, *Slow DoS* foram capturados (Vaccari et al., 2020).

Os Datasets Netflow (Sarhan et al., 2021) foram construídos para preencher a lacuna da falta de padronização dos conjuntos de dados e da variação das informações representadas. Quatro conjuntos de dados recentes foram escolhidos, UNSW-NB15, BoT-IoT, ToN-IoT e CICIDS2018 usando seus arquivos de captura de pacotes disponíveis publicamente, através desses

conjuntos de dados, os autores converteram para um formato baseado em recursos do NetFlow. Pcaps publicamente disponíveis do conjunto de dados ToN-IoT foram usados para gerar seus registros NetFlow, levando a um conjunto de dados de rede IoT baseado em NetFlow chamado NF-ToN-IoT. O número total de fluxos de dados é de 1.379.274, dos quais 1.108.995 (80,4%) são amostras de ataque e 270.279 (19,6%) são benignas. Este conjunto de dados consiste em 14 recursos e inclui todos os ataques que existem no conjunto de dados ToN-IoT. Na versão 2 os conjuntos de dados Netflow possuem 43 características comuns entre os conjuntos de dados e foram construídos de forma semelhante. A versão 2 do NetFlow do conjunto de dados UNSW-NB15, denominado NF-UNSW-NB15-v2, possui 43 recursos e 2.390.275 fluxos de dados, dos quais 95.053 (3,98%) são amostras de ataque e 2.295.222 (96,02%) são benignos. As amostras de ataque continuam a ser classificadas em nove subcategorias.

B.5.10 QPURS1 - Quais os principais locais de publicação?

Nesta seção são apresentados os locais de publicação dos trabalhos encontrados no estado da arte. A Tabela 54 apresenta os principais. É possível observar que o *IEEE Access* e o *IEEE Internet of Things Journal* foram os periódicos com maior número de publicações, sete publicações cada.

Tabela 54 – Principais locais de publicação dos trabalhos do estado da arte.

Locais de publicação	**Fator de impacto**	**Trabalhos**
IEEE Access	3,9	[19], [22], [30], [46], [51], [52], [69]
IEEE Internet of Things Journal	10,238	[20], [45], [49], [66], [68], [79], [80]
Future Generation Computer Systems	7,307	[15], [43], [71], [103]
Wireless Communications and Mobile Computing	2,146	[10], [16], [60], [64]
Security and Communication Networks	1,968	[11], [89], [90]
Journal of Ambient Intelligence and Humanized Computing	3,662	[54], [57], [74]
Transactions on Emerging Telecommunications Technologies	3,31	[4], [58], [97]
Computers & Security	5,6	[21], [37]
Electronics	2,69	[63], [85]
Cluster Computing	2,303	[29], [98]
IEEE Wireless Communications and Networking Conference (WCNC)	1,744	[27], [32]

Além disso, também houve destaque para os periódicos *Future Generation Computer Systems* e o *Wireless Communications and Mobile Computing*, dos trabalhos incluídos no mapeamento, eles publicaram quatro trabalhos cada. Os conceituados periódicos *Computers & Security*, *Electronics* e *Cluster Computing* também obtiveram destaque com dois artigos publicados cada.

Dentre as conferencias, destaca-se a *IEEE Wireless Communications and Networking Conference* (WCNC), que apresentou 2 estudos publicados.

B.5.11 QPURS2 - Quantos trabalhos foram publicados por ano?

Nesta seção, são abordados os anos de publicação dos estudos incluídos neste trabalho. A Figura 30 apresenta uma relação entre os anos e os trabalhos publicados.

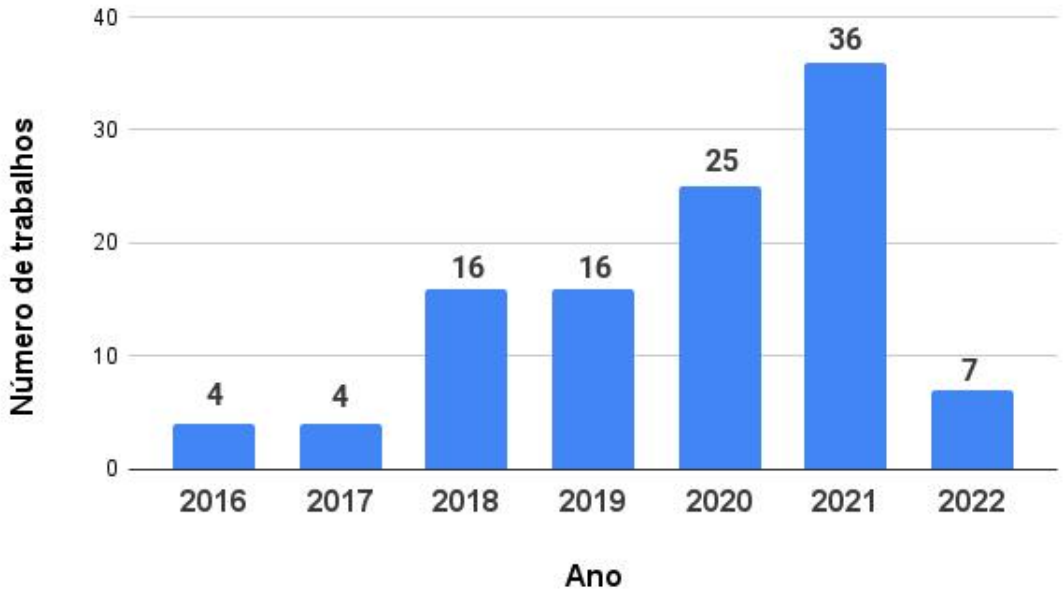


Figura 30 – Número de trabalhos publicados por ano.

Em ordem cronológica, os primeiros artigos incluídos no mapeamento foram publicados em 2016. Em 2017, foram publicados 4 artigos (Pan; Pacheco; Hariri, 2017; Sandhu; Sohal; Sood, 2017; Yaseen et al., 2017; Yaseen et al., 2017).

Posteriormente, em 2018 foram publicados 16 estudos, assim como em 2019, onde mais 16 artigos foram publicados. Em 2020 foram publicados 25 trabalhos. O ano com maior número de artigos foi 2021, quando foram publicados 36 artigos. Por fim, no ano de 2022 foram publicados mais 7 artigos incluídos no mapeamento. Considerando que a busca por estudos abrangeu apenas os primeiros meses de 2022, o número de obras publicadas nesse ano é maior.