

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
CENTRO TECNOLÓGICO  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA  
CURSO SISTEMAS DE INFORMAÇÃO

PEDRO SILVEIRA DALENOGARE

**ABORDAGEM PARA DETECÇÃO E IDENTIFICAÇÃO DE INTRUSÃO EM  
AMBIENTES DE FOG COMPUTING E IOT**

FLORIANÓPOLIS

2020

PEDRO SILVEIRA DALENOGARE

**ABORDAGEM PARA DETECÇÃO E IDENTIFICAÇÃO DE INTRUSÃO EM  
AMBIENTES DE FOG COMPUTING E IOT**

Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do Título de Bacharel em Sistemas de Informação.

Orientador: Carlos Becker Westphall

Coorientador: Cristiano Antonio de Souza

FLORIANÓPOLIS

2020

#### Ficha de identificação da obra

Dalenogare, Pedro Silveira

Abordagem para detecção e identificação de intrusão em ambientes de fog computing e IoT / Pedro Silveira Dalenogare ; orientador, Carlos Becker Westphall, coorientador, Cristiano Antonio de Souza, 2020.

84 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistema de Informação, Florianópolis, 2020.

Inclui referências.

1. Sistema de Informação. 2. Intrusion Detection Systems. 3. Internet of Things. 4. Fog computing. 5. Artificial Neural Networks. I. Westphall, Carlos Becker . II. de Souza, Cristiano Antonio. III. Universidade Federal de Santa Catarina. Graduação em Sistema de Informação. IV. Título.

Pedro Silveira Dalenogare

**Abordagem para detecção e identificação de intrusão em ambientes de fog  
computing e IoT**

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel e aprovado em sua forma final pelo Curso de Sistemas de Informação

Local, 02 de dezembro de 2020.

---

Prof. Dr. Cristian Koliver  
Coordenador do Curso

**Banca Examinadora:**

---

Prof. Dr. Carlos Becker Westphall  
Orientador  
Universidade Federal de Santa Catarina

---

Me. Cristiano Antonio de Souza  
Coorientador  
Universidade Federal de Santa Catarina

---

Prof.<sup>a</sup> Dra. Carla Merkle Westphall  
Universidade Federal de Santa Catarina

---

Prof Dr. Jorge Werner  
Universidade Federal de Santa Catarina

## **AGRADECIMENTOS**

Agradeço primeiramente aos meus pais, Silvia Maria Silveira e Neri Samuel Dalenogare, por toda educação, atenção e condições que me foram dadas ao longo da vida. Por isto, serei eternamente grato.

Agradeço à minha namorada Nathalia Gomes dos Santos pelo apoio e incentivo desde o início do desenvolvimento deste trabalho, sempre me motivando, sendo fundamental ao longo deste período.

Agradeço à Universidade Federal de Santa Catarina, por permitir a realização deste curso, assim como a todos os professores que fizeram parte deste processo.

Agradeço aos meus colegas de curso e amigos para vida Vinícius Eduardo, Pedro Steinheiser, Lucas Rocha e Cesar Bess, por todas as risadas e momentos vividos dentro desta universidade.

Agradeço ao meu coorientador Cristiano Antonio de Souza, por todo o conhecimento passado e pela atenção durante todo o período de elaboração deste trabalho.

## RESUMO

A segurança é um dos pontos-chave para o sucesso das tecnologias existentes em geral. Na Internet das Coisas (*Internet of Things* - IoT), onde há uma grande quantidade de dispositivos gerando e compartilhando um grande volume de dados, não é diferente. Nesta recente tecnologia, a segurança é, sem dúvidas, uma das áreas que merece mais atenção, pois é fundamental para a garantia da integridade e privacidade dos dados dos usuários. Um dos métodos existentes para a manutenção da segurança em IoT são os Sistemas de Detecção de Intrusão, que possuem como objetivo detectar tentativas de ataque e intrusão nos dispositivos inteligentes. Uma prática recorrente no desenvolvimento destes sistemas é a utilização de Redes Neurais Artificiais, uma estrutura que permite a criação de algoritmos de aprendizado de máquina para detecção de intrusão. A maior parte das abordagens existentes focam na detecção binária, ou seja, tratam um evento como normal ou como ataque, sem especificar o tipo do ataque. Outro modelo existente é o de classificação multiclasse, que é capaz de detectar a intrusão e identificar qual tipo de ataque se trata, como por exemplo, de negação de serviço. As abordagens de classificação múltipla servem para auxiliar na execução de contramedidas, pois identificando o tipo de ataque que está sendo executado, torna-se mais fácil contê-lo, facilitando para o gerenciador da rede identificar os ataques mais recorrentes e proteger de forma mais eficiente o sistema em questão. Este trabalho de conclusão de curso tem como objetivo propor uma abordagem para detecção e identificação de intrusão em *fog computing* e IoT utilizando Redes Neurais Artificiais treinadas para identificar possíveis ataques aos dispositivos.

**Palavras-chave:** Internet das Coisas. Detecção de Intrusão. Redes Neurais Artificiais.

## **ABSTRACT**

*Security is one of the key points for the success of existing technologies in general. In the Internet of Things (IoT), where there is a large amount of devices generating and sharing a large volume of data, it is no different. In this recent technology, security is, without a doubt, one of the fields that deserves more attention, since it is fundamental to guarantee the integrity and privacy of users' data. One of the existing methods for maintaining security in IoT are the Intrusion Detection Systems, which aim to detect attack attempts and intrusion in intelligent devices. A recurring practice in the development of these systems is the use of Artificial Neural Networks, a structure that allows the creation of machine learning algorithms for intrusion detection. Most of the existing approaches focus on binary detection, that is, they treat an event as normal or as an attack, without specifying the type of attack. Another existing model is the multiclass classification, which is able to detect an intrusion and identify what type of attack is, for example, denial of service. The multiple classification approaches serve to assist in the execution of countermeasures, once it is known which type of attack is being executed, it becomes easier to contain, making it easier for a network manager to identify the most recurring attacks and protect the system in question more efficiently. This course conclusion paper aims to propose an approach to detect and identify intrusion in fog computing and IoT using Artificial Neural Networks trained to identify possible attacks to devices.*

*Keywords: Internet of Things. Intrusion Detection. Artificial Neural Networks.*

## LISTA DE FIGURAS

Figura 1 - Ecossistema baseado em cloud computing para dispositivos IoT. ....	22
Figura 2 - Exemplo de rede MLP.....	27
Figura 3 - Representação de um Neurônio Artificial.....	28
Figura 4 - Modelo proposto. ....	33
Figura 5 - Ambiente com redes de dispositivos IoT.....	34
Figura 6 – Ilustração da abordagem proposta.....	36
Figura 7 - Ilustração do método <i>5-fold cross-validation</i> .....	43
Figura 8 - Arquiteturas com duas camadas ocultas. ....	44
Figura 9 - Arquiteturas com três camadas ocultas. ....	45
Figura 10 - Arquiteturas com quatro camadas ocultas. ....	45
Figura 11 - Arquiteturas com cinco camadas ocultas.....	46
Figura 12 - Ilustração dos experimentos realizados.....	47
Figura 13 - Especificações do processador.....	48
Figura 14 - Especificações da memória .....	49



## LISTA DE GRÁFICOS

Gráfico 1 - Resultados obtidos para classe Normal; .....	55
Gráfico 2 - Resultados obtidos para classe Dos.....	55
Gráfico 3 - Resultados obtidos para classe Probe. ....	56
Gráfico 4 - Resultados obtidos para classe R2L. ....	57
Gráfico 5 - Resultados obtidos para classe U2R.....	57
Gráfico 6 - Comparação entre arquiteturas com diferentes números de camadas. ..	58
Gráfico 7 - Comparação entre arquiteturas com diferentes números de neurônios. .	59

## LISTA DE QUADROS

Quadro 1 - Atributos da base NSL-KDD.....	39
Quadro 2 - Nomenclatura das arquiteturas. ....	46

## LISTA DE TABELAS

Tabela 1 - Amostras do conjunto de treino da NSL-KDD. ....	38
Tabela 2 - Resultados dos experimentos com a arquitetura 2C41N. ....	50
Tabela 3 - Resultados dos experimentos com a arquitetura 2C82N. ....	51
Tabela 4 - Resultados dos experimentos com a arquitetura 3C41N. ....	51
Tabela 5 - Resultados dos experimentos com a arquitetura 3C82N. ....	52
Tabela 6 - Resultados dos experimentos com a arquitetura 4C41N. ....	52
Tabela 7 - Resultados dos experimentos com a arquitetura 4C82N. ....	53
Tabela 8 - Resultados dos experimentos com a arquitetura 5C41N. ....	53
Tabela 9 - Resultados dos experimentos com a arquitetura 5C82N. ....	54

## LISTA DE ABREVIATURAS E SIGLAS

IoT *Internet of Things*

IDS *Intrusion Detection Systems*

RNA *Redes Neurais Artificiais*

U2R *User to Root*

R2L *Remote to Local*

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b> .....	<b>15</b>
1.1	MOTIVAÇÕES.....	17
1.2	OBJETIVO GERAL.....	17
1.3	OBJETIVOS ESPECÍFICOS .....	17
1.4	ORGANIZAÇÃO DO TRABALHO.....	18
<b>2</b>	<b>CONCEITOS BÁSICOS</b> .....	<b>19</b>
2.1	IOT.....	19
2.1.1	<b>Dispositivos</b> .....	<b>19</b>
2.1.2	<b>Tecnologias de comunicação</b> .....	<b>20</b>
2.1.3	<b>Visão geral de segurança em IoT</b> .....	<b>20</b>
2.2	CLOUD COMPUTING .....	21
2.3	FOG COMPUTING .....	21
2.4	EVOLUÇÃO DA CLOUD COMPUTING PARA A FOG COMPUTING.....	23
2.5	IDS.....	23
2.5.1	<b>Tipos de Detecção</b> .....	<b>23</b>
2.5.2	<b>Tipos de Arquitetura de Acordo com o Alvo</b> .....	<b>24</b>
2.5.3	<b>Tipos de Arquitetura de Acordo com o Local</b> .....	<b>25</b>
2.5.4	<b>Comportamento Pós Detecção</b> .....	<b>25</b>
2.5.5	<b>Frequência de Uso</b> .....	<b>26</b>
2.6	MACHINE LEARNING .....	26
2.6.1	<b>Redes Neurais Artificiais</b> .....	<b>27</b>
<b>3</b>	<b>ESTADO DA ARTE</b> .....	<b>30</b>
<b>4</b>	<b>PROPOSTA DE DESENVOLVIMENTO DO TRABALHO</b> .....	<b>34</b>
<b>5</b>	<b>AVALIAÇÃO</b> .....	<b>38</b>
5.1	EXPERIMENTOS .....	38
5.1.1	<b>Base de Dados</b> .....	<b>38</b>
5.1.2	<b>Métricas de Avaliação</b> .....	<b>41</b>
5.1.3	<b>Cross-validation</b> .....	<b>43</b>
5.1.4	<b>Arquiteturas Utilizadas</b> .....	<b>44</b>
5.1.5	<b>Ferramentas Utilizadas</b> .....	<b>48</b>
5.2	RESULTADOS .....	50

5.3	DISCUSSÕES.....	59
6	CONCLUSÃO.....	61
	REFERÊNCIAS.....	62
	APÊNDICE A – ARTIGO DA MONOGRAFIA .....	65

## 1 INTRODUÇÃO

Com a popularização da Internet das Coisas (*Internet of Things* - IoT), tecnologia em alta nos tempos atuais e utilizada em diversas áreas como agricultura, saúde, casas inteligentes, entre outras aplicações (U.FAROOQ *et al.*, 2015), algumas preocupações acerca deste tema vêm se destacando.

Aliada à inovação, esta recente tecnologia é seguida de muitos desafios, sendo o principal deles, sem dúvidas, a segurança. Para o sucesso do conceito da IoT, é imprescindível que haja a garantia da segurança e privacidade dos dados dos usuários. A detecção de intrusão é uma das técnicas utilizadas para aumentar segurança dos dispositivos, tendo como objetivo detectar e identificar tentativas de intrusão em ambientes computacionais por parte de invasores internos e externos (MUKHERJEE; HEBERLEIN; LEVITT, 1994). Os Sistemas de Detecção de Intrusão (*Intrusion Detection Systems* - IDS) se diferenciam em diversos aspectos, como o tipo de detecção realizada, a arquitetura do sistema de acordo com o alvo das detecções, o local onde está disposto, entre outras características.

Com o aumento da quantidade de dados geradas pelos dispositivos IoT, se fez necessária a utilização de outras tecnologias para auxiliar no processamento e armazenamento dos dados. Inicialmente foi utilizada a *cloud computing*, tanto no processamento quanto no armazenamento de dados. Esta tecnologia, porém, apresentou alta latência e baixa eficiência no processamento dos dados, por atuar distante do dispositivo IoT (YANNUZZI *et al.*, 2014). Visto que os dispositivos necessitam de uma resposta imediata, a *fog computing* foi adotada para atuar no processamento dos dados, já que este paradigma atua na borda da rede, mais próxima dos dispositivos (BONOMI *et al.*, 2012). Como o processamento dos dados dos dispositivos IoT são feitos no nível da *fog*, é possível propor uma abordagem de detecção e identificação de intrusão atuando neste nível, através da análise dos tráfegos enviados pelos dispositivos.

Outra tecnologia bastante utilizada nos dispositivos IoT, especialmente em métodos de detecção de intrusão, são as Redes Neurais Artificiais (RNA), que podem ser utilizadas no processo de aprendizado de máquina, através da utilização

de um conjunto de treinamento (HAYKIN, 2001), ou seja, permite o desenvolvimento de abordagens eficientes com os mais diversos objetivos.

Neste trabalho é proposta uma abordagem que utiliza RNA com o objetivo de detectar e identificar pacotes maliciosos. Esta abordagem atua no nível da *fog*, que como dito anteriormente, é responsável pelo processamento dos dados da tecnologia IoT, ou seja, permite a captura e análise dos tráfegos entre os dispositivos.

Atualmente, existem diversas abordagens de detecção de intrusão que utilizam RNA. Porém, grande parte destes trabalhos focam exclusivamente na detecção binária, ou seja, apenas detectam se um evento é intrusivo ou não, sem especificar o tipo de ataque. Um outro modelo existente é o de classificação multiclasse, que é capaz de detectar a intrusão e identificar a qual tipo de ataque ela pertence. As abordagens de classificação múltipla podem auxiliar na execução de contramedidas, uma vez que é conhecido o tipo de ataque que está sendo executado, torna-se mais fácil contê-lo. Este tipo de classificação também facilita para o gerenciador da rede identificar os tipos de ataques realizados e garantir uma melhor segurança para o sistema em questão. Por exemplo, se a abordagem multiclasse detecta um ataque de acesso remoto (U2R), o gerente da rede pode tratar este evento intrusivo e já exercer uma contramedida para um possível ataque de escalonamento de privilégio (R2L), tendo em vista que estes dois ataques geralmente acontecem nesta ordem. O número inferior de trabalhos com este tipo de abordagem em relação aos de detecção binária é uma das motivações para a realização deste trabalho.

Com o objetivo de encontrar a abordagem mais eficiente utilizando RNA, serão apresentadas diferentes arquiteturas, variando os números de camadas ocultas e de neurônios. Após a realização do treinamento das arquiteturas, é realizada uma avaliação individual por arquitetura, levando em conta as métricas selecionadas. Posteriormente, uma avaliação geral dos resultados é realizada observando o desempenho das arquiteturas considerando as duas principais métricas desta abordagem: acurácia e precisão.



## 1.1 MOTIVAÇÕES

Assim como em outras tecnologias, a segurança é um ponto crucial para o sucesso dos dispositivos IoT. Com o aumento do número de usuários e a popularidade dos dispositivos inteligentes, surgem também novos ataques que colocam em cheque a confiabilidade desta tecnologia.

Um recente caso envolvendo as vulnerabilidades da segurança em IoT ocorreu em 2016, através do *Mirai botnet*. Esta *botnet* é um *malware* que utiliza computadores remotamente controlados, também conhecido como *bots*, para formarem uma rede, que possui por objetivo realizar ataques de rede em grande escala. Neste caso, o ataque foi realizado no provedor de serviços *Dyn* (KOLIAS et al., 2017). Resumidamente, o ataque infectou e tornou dispositivos IoT em *bots*. Estes dispositivos infectados ajudaram a proliferar o ataque, detectando e infectando outros dispositivos conectados na rede, possibilitando a formação de uma rede de *bots*, posteriormente utilizada para realizar um ataque *DDoS* nos servidores alvos. Este ataque teve grande impacto, pois retirou do ar sites famosos e muito utilizados como *Netflix*, *Twitter*, *GitHub* e *Reddit* por diversas horas. Como lição, serviu para mostrar que a questão de segurança em IoT ainda necessita muita atenção e estudos para que sejam desenvolvidas novas técnicas e aplicações cada vez mais eficientes para impedir ataques similares.

## 1.2 OBJETIVO GERAL

O objetivo geral deste trabalho de conclusão de curso é propor uma abordagem baseada em Redes Neurais Artificiais para detecção e identificação de intrusão, visando a classificação de ataques à *fog computing* e IoT.

## 1.3 OBJETIVOS ESPECÍFICOS

Como objetivos específicos foram listados os dois principais:

1. Realizar uma revisão do estado da arte relacionado a detecção de intrusão em ambiente de *fog computing* e IoT utilizando Redes Neurais Artificiais e *machine learning*, identificando os principais problemas e soluções existentes;
2. Propor uma abordagem baseada em Redes Neurais Artificiais para detecção e identificação de intrusão em ambientes de *fog computing* e IoT, utilizando Redes Neurais Artificiais com método de classificação;
3. Realizar a detecção multiclasse, especificando o tipo de intrusão detectada;
4. Encontrar a quantidade ideal de camadas ocultas e neurônios, analisando a influência da alteração destes valores nos resultados.

#### 1.4 ORGANIZAÇÃO DO TRABALHO

Na Seção 2 são abordados os conceitos dos temas principais deste trabalho, sendo estes IoT, *cloud* e *fog computing*, IDS, *machine learning* e Redes Neurais Artificiais. A Seção 3 apresenta o estado da arte, contextualizando o estado atual das pesquisas e estudos na área de detecção de intrusão em dispositivos IoT, comentando sobre alguns trabalhos realizados. Na Seção 4 é apresentada a abordagem proposta deste trabalho. A Seção 5 abrange toda a metodologia utilizada para a execução da abordagem. Também comenta sobre os resultados e discussões referente aos experimentos realizados. A Seção 6 contém uma conclusão sobre o trabalho realizado, falando a respeito dos temas principais deste trabalho, realizando uma reflexão dos resultados obtidos pela abordagem proposta e comentando sobre possíveis trabalhos futuros.

## 2 CONCEITOS BÁSICOS

Neste capítulo são apresentados alguns conceitos básicos a respeito de temas presentes neste trabalho, como *Internet of Things* - IoT, computação em nuvem (*Cloud Computing*), computação em nevoeiro (*Fog Computing*), *Intrusion Detection Systems* - IDS. A Seção 2.6 apresenta o conceito de *machine learning* e aborda de maneira sucinta sua aplicabilidade em algoritmos de detecção de intrusão. Por fim, a Seção 2.6.1 aborda de maneira mais profunda as Redes Neurais Artificiais (RNA), apresentando seus conceitos e demonstrando como se dá o processo de aprendizado de máquina.

### 2.1 IOT

A Internet das Coisas pode ser vista como uma rede de objetos físicos interligados através de uma infraestrutura de rede que são capazes de interagirem entre si e trocar informações coletadas no ambiente, sendo utilizados em diversas áreas, como casas e cidades inteligentes (WORTMANN; FLÜCHTER, 2015). Esta tecnologia permite que os objetos troquem informações entre si sem a necessidade da interferência humana. Os dispositivos IoT possuem baixa capacidade de processamento e armazenamento dos dados, por este motivo, há a necessidade de utilização de paradigmas como *cloud computing* e *fog computing*, para atuarem no processamento dos dados gerados pelos dispositivos (YANNUZZI et al., 2014).

#### 2.1.1 Dispositivos

Devido ao baixo custo e fácil manipulação, os dispositivos IoT vem sendo cada vez mais utilizados nos dias de hoje. O crescente aumento do número de dispositivos é proporcional a quantidade de dados gerados. Devido ao fato destes dispositivos possuírem recursos limitados, há a necessidade do uso de paradigmas para o processamento e armazenamento dos dados. Inicialmente, o paradigma mais utilizado era a *cloud computing*, porém, atualmente o paradigma que melhor atende

os dispositivos IoT na parte de processamento de dados é a *fog computing*, que é um paradigma que surgiu para atuar mais próximo aos dispositivos IoT (BONOMI et al., 2012).

### 2.1.2 Tecnologias de comunicação

A comunicação é o ponto chave para o sucesso da tecnologia IoT, a capacidade de coletar e trocar informações entre dispositivos faz com que essa tecnologia seja cada vez mais promissora.

Os dispositivos inteligentes podem possuir tanto a conexão a cabo quanto sem fio. Em relação a conexão sem fio, diversos protocolos e tecnologias podem ser utilizados, como *Internet Protocol Version 6 (IPv6)*, *Low power Wireless Personal Area Networks (6LoWPAN)*, *ZigBee*, *Bluetooth Low Energy (BLE)*, *Z-Wave* e *Near Field Communication (NFC)*. Todos os protocolos citados anteriormente são de curto alcance, ao contrário dos protocolos *SigFox* e *Cellular*, que são exemplos de protocolos utilizados em IoT, mas que são *Low Power Wide Area Network (LPWAN)*, ou seja, redes sem fio com grande alcance para dispositivos de baixa potência (ALSARAWI et al., 2017).

### 2.1.3 Visão geral de segurança em IoT

Aliada à inovação, a tecnologia IoT é seguida de muitos desafios, sendo o principal deles a garantia da segurança e privacidade dos usuários, extremamente importante para o seu sucesso. A detecção de intrusão é um dos pontos chaves da segurança, tendo por objetivo detectar tentativas de ataques por parte de usuários e algoritmos mal-intencionados. A visibilidade oferecida por um sistema de detecção no nível do nó é única e pode melhorar a detecção geral de invasões. No entanto, as capacidades restritas de recursos inviabilizam as ferramentas tradicionais de detecção. Desse modo, existe a necessidade de uma abordagem capaz de detectar intrusões no nível da *fog*, onde os pacotes emitidos pelos dispositivos são processados.

## 2.2 CLOUD COMPUTING

Computação em nuvem pode ser vista como um paradigma que permite acesso onipresente a um conjunto de recursos de computação configuráveis, como por exemplo: armazenamento, serviços, aplicativos, entre outros que podem ser rapidamente disponibilizados (MELL; GRANCE, 2011). Dropbox, Gmail e Google Maps são alguns exemplos de aplicações famosas que utilizam *cloud computing*.

Este modelo é composto por três tipos de serviço, sendo eles: *Hardware as a Service*, *Software as a Service* e *Platform as a Service* (MELL; GRANCE, 2011).

- *Infrastructure as a Service (IaaS)*: Terceirização de servidores e data centers para memória, armazenamento, processamento, entre outros tipos de infraestruturas que o usuário contratante necessitar. Neste modelo de serviço, o usuário paga apenas aquilo que ele utiliza.
- *Software as a Service (SaaS)*: Disponibilização de aplicações prontas que permitem que os usuários paguem apenas pelo que utilizam, ao contrário do que ocorre com o modelo tradicional de vendas de licenças e unidades de software. Google Drive é um exemplo muito utilizado de *Software as a Service*.
- *Platform as a Service (PaaS)* : Este modelo de serviço pode ser visto como um meio termo entre *IaaS* e *SaaS*. Neste modelo, os sistemas *cloud* podem oferecer a plataforma de software na qual os usuários podem desenvolver, rodar e gerenciar aplicações de acordo com suas necessidades, sem precisar se preocupar com a infraestrutura necessária para o desenvolvimento e utilização das aplicações.

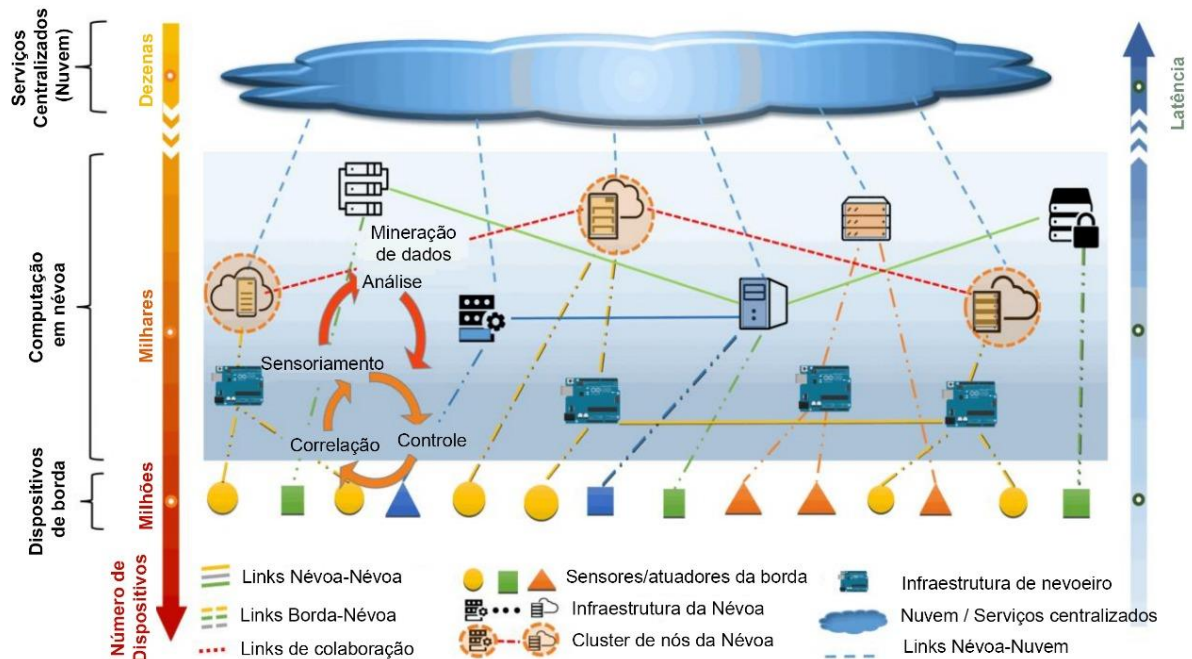
## 2.3 FOG COMPUTING

A *fog computing* pode ser vista como uma plataforma virtualizada que possui como objetivo fornecer serviços de computação, armazenamento, e rede entre dispositivos e *data centers*, atuando grande parte das vezes na borda da rede (BONOMI et al., 2012). Ou seja, em comparação à *cloud*, a *fog* atua de maneira

mais próxima aos dispositivos IoT, permitindo que os dados possam ser processados com mais eficiência antes de serem enviados ou não à *cloud*. O principal componente da arquitetura do fog são os *fog nodes*, que podem ser compreendidos como componentes físicos (gateways, switches, servidores, roteadores) ou virtuais (máquinas virtuais, switches virtuais, entre outros) (IORGA et al., 2018). Assim como a computação em nuvem, a computação em névoa oferece três modelos de serviços, sendo eles: *Software as a Service (SaaS)*, *Platform as a Service (PaaS)* e *Infrastructure as a Service (IaaS)*, comentados na Seção anterior.

A Figura 1 apresenta a utilização da *fog computing* em um ecossistema de computação em nuvem voltado à dispositivos inteligentes. Nesta imagem podemos observar a relação da proximidade da *fog* e da *cloud* em relação aos dispositivos finais, comentado anteriormente. Outra observação que pode ser feita diz respeito a latência. Quanto mais distante dos dispositivos finais, maior a latência. Quanto mais próximo dos dispositivos finais, menor a latência. Esta característica é crucial na hora de decidir entre adotar *fog* ou *cloud* para efetuar o processamento dos dados, tendo em vista que alta latência.

Figura 1 - Ecossistema baseado em cloud computing para dispositivos IoT.



Fonte: Adaptado de IORGA et al., 2018.

## 2.4 EVOLUÇÃO DA CLOUD COMPUTING PARA A FOG COMPUTING

A principal motivação do uso da *fog* no lugar da *cloud*, é que a *cloud computing* se trata de uma estrutura centralizada, ou seja, a separação entre os *data centers* de processamento de dados e os dispositivos IoT acaba por implicar em uma alta latência, não desejada por aplicações que trabalham com respostas em tempo real. Já a *fog computing* é utilizada de maneira mais próxima aos dispositivos, permitindo um eficiente processamento dos dados e garantindo baixa latência, extremamente desejada pelos usuários de IoT.

## 2.5 IDS

A detecção de intrusão é um dos mecanismos utilizados para tentar garantir a segurança na Internet das Coisas. O objetivo da detecção de intrusão é detectar e identificar tentativas de intrusão no ambiente computacional por parte de invasores internos e externos (MUKHERJEE; HEBERLEIN; LEVITT, 1994).

### 2.5.1 Tipos de Detecção

Os *Intrusion Detection Systems* (IDS) podem ser classificados de acordo com os métodos de detecção empregados. As principais abordagens de detecção de ataques são: por anomalia e por abuso (NORTHCUTT et al., 2001).

Na detecção por abuso, há uma espécie de banco de dados de comportamentos maliciosos já conhecidos. Esses padrões são chamados de assinaturas e possuem duas características positivas: permite uma rápida detecção e diminui a ocorrência de alarmes falsos, em contrapartida, possui a limitação de detectar apenas ataques conhecidos (NORTHCUTT et al., 2001).

Quando a detecção é por anomalia, o sistema de detecção considera qualquer ação intrusiva como anômala, ou seja, se a atividade não apresenta comportamentos definidos como normais, ela é automaticamente considerada uma intrusão. A grande vantagem desta técnica é que ela permite que novos ataques, ou

até mesmo variações de ataques já conhecidos sejam detectados, pois não é necessário conhecê-los previamente, como no método de detecção por abuso. A desvantagem, no entanto, é que esta técnica tem maior tendência de classificar eventos erroneamente, pois nem toda atividade anômala é uma intrusão (MACHADO, 2005).

### 2.5.2 Tipos de Arquitetura de Acordo com o Alvo

Um sistema de detecção de intrusão pode ser classificado como HIDS (*Host-based Intrusion Detection Systems*), NIDS (*Network-based Intrusion Detection System*) ou híbrido. Sistemas de detecção de intrusão baseados em rede (NIDS) coletam dados de entrada através do monitoramento do tráfego da rede. Já os sistemas baseados em *host* (HIDS) coletam esses dados a partir do *host* que está sendo monitorado. Um IDS pode ser classificado ainda como híbrido, que é quando a coleta dos dados é feita de ambas as formas, através da rede e do *host* (BUTUN; MORGERA; SANKAR, 2014).

Sistemas de detecção de intrusão baseados em rede se destacam pelo alcance de monitoramento. Quando bem posicionados, poucos IDS baseados em rede são capazes de monitorar uma rede bastante extensa. Em contrapartida, este tipo de IDS pode encontrar dificuldades para processar os dados de uma rede muito grande, abrindo portas para que ataques sejam realizados em horários de tráfego intenso, sem que o IDS consiga reconhecer um ataque lançado neste período (BACE; MELL, 2001).

Uma das principais vantagens dos IDS baseados em *host* é a capacidade de detectar eventos locais para um *host*, o que não é possível de se fazer com um sistema baseado em rede. Porém, sistemas de detecção de intrusão que possuem esta arquitetura são mais difíceis de lidar, pois a informação deve ser configurada e gerenciada individualmente para cada *host* monitorado (BACE; MELL, 2001).



### 2.5.3 Tipos de Arquitetura de Acordo com o Local

Os IDS podem ainda ser classificados de acordo com a disposição de seus componentes. Podem ser classificados como centralizados, parcialmente distribuídos e distribuídos (CAMPELLO; WEBER, 2001).

Um IDS é chamado de centralizado quando a análise de dados é realizada de maneira centralizada, ou seja, concentrada em um único ponto, independentemente do número de *hosts* que estão sendo monitorados, mantendo assim todos os seus componentes funcionais centralizados. Uma das principais vantagens deste tipo de IDS é a fácil configuração. Em contrapartida, a escalabilidade é um ponto em que os sistemas centralizados pecam, já que o aumento no número de *hosts* monitorados pode sobrecarregar o centro de processamento (SPAFFORD; ZAMBONI, 2000).

Para um sistema de detecção de intrusão ser considerado parcialmente distribuído, seus componentes devem estar espalhados por diversos pontos da rede, porém, gerenciados por módulos centralizados. Apesar da disposição dos componentes ser característica de sistemas distribuídos, a tomada de decisão é centralizada num único ponto, o que faz com que este sistema seja parcialmente distribuído.

Os IDS que possuem seus componentes espalhados por diversos pontos da rede, utilizando mensagens para se comunicar, são chamados IDS distribuídos. Nestes sistemas, a análise de dados é realizada individualmente pelos *hosts*. A principal vantagem deste tipo de arquitetura é a abrangência de detecção, tendo em vista que os componentes estão espalhados pelo sistema se comunicando através de trocas de mensagens (SPAFFORD; ZAMBONI, 2000).

### 2.5.4 Comportamento Pós Detecção

O comportamento pós detecção diz respeito ao *feedback* que o IDS emite após a detecção de um evento intrusivo. O comportamento pode ser caracterizado como passivo ou ativo (BACE; MELL, 2001).

No comportamento passivo, o IDS apenas informa o usuário do sistema sobre o evento detectado, confiando que ele tomará alguma ação para impedir o ataque de acordo com as informações que o sistema forneceu. No comportamento ativo, há diferentes opções de ações automáticas que o sistema pode executar de acordo com o tipo de intrusão detectada, são elas: coletar informações adicionais sobre o ataque, pausar o ataque e bloquear o atacante, e por último, lançar ataques ao atacante e/ou obter informações a respeito do mesmo (BACE; MELL, 2001).

### 2.5.5 Frequência de Uso

A frequência de uso do IDS diz respeito ao intervalo de tempo entre os eventos que são monitorados e a análise dos mesmos.

Nos IDS baseados em intervalos, também conhecidos como IDS de análise periódica, não há um fluxo contínuo entre os pontos de monitoramento e os mecanismos de análise. A análise dos dados coletados pelos pontos de monitoramento é realizada em períodos predefinidos. Desse modo, o uso de processamento pode ser reduzido e melhor gerenciado (ILGUN, 1993).

Nos IDS de monitoramento contínuo, também conhecidos como IDS de tempo real, realizam a análise no momento em que os eventos são coletados pelos pontos de monitoramento utilizado por soluções baseadas em rede, pois os dados analisados provêm de tráfego constante de pacotes na rede, exigindo uma análise imediata (BACE; MELL, 2001).

## 2.6 MACHINE LEARNING

Uma das tecnologias ascendentes na atualidade é o *machine learning*. Trata-se de uma das técnicas de Inteligência Artificial que permite um sistema adquirir conhecimento através do reconhecimento de padrões, utilizando conjuntos de dados de treinamento (XIE et al., 2019).

De acordo com Naqa e Murphy (2015), é uma área em evolução dos algoritmos computacionais que simulam a inteligência humana, capazes de adquirir conhecimentos com o ambiente.

Na área de segurança computacional, esta técnica vem sendo bastante utilizada devido à sua eficiência no desenvolvimento de algoritmos de detecção de intrusão. Um dos modelos que podem ser citados, que inclusive será abordado neste trabalho, se trata das Redes Neurais Artificiais (RNA).

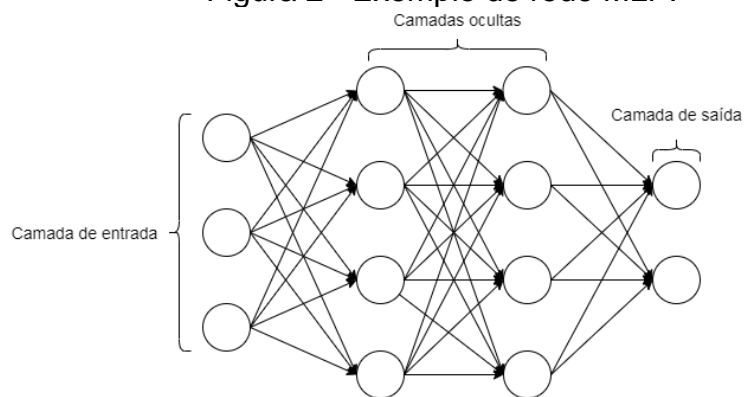
### 2.6.1 Redes Neurais Artificiais

Uma RNA pode ser vista como uma estrutura paralelamente distribuída que possui por função processar informações de maneira similar ao cérebro, através da sinapse entre neurônios interligados, com o objetivo de realizar uma função específica (HAYKIN, 2001).

As RNA podem ser utilizadas no processo de aprendizagem de máquina, através da utilização do algoritmo de aprendizagem, responsável por modificar os pesos sinápticos da rede, pesos estes que são responsáveis por armazenar o conhecimento da rede (HAYKIN, 2001).

Existem diversos modelos de RNA. Neste trabalho, o modelo abordado é a *multilayer perceptron (MLP) feedforward*, que se trata de um modelo similar ao *perceptron*, utilizando conexões com pesos entre os neurônios, mas que não se limita a uma classificação binária, conforme ilustrado na Figura 2.

Figura 2 - Exemplo de rede MLP.



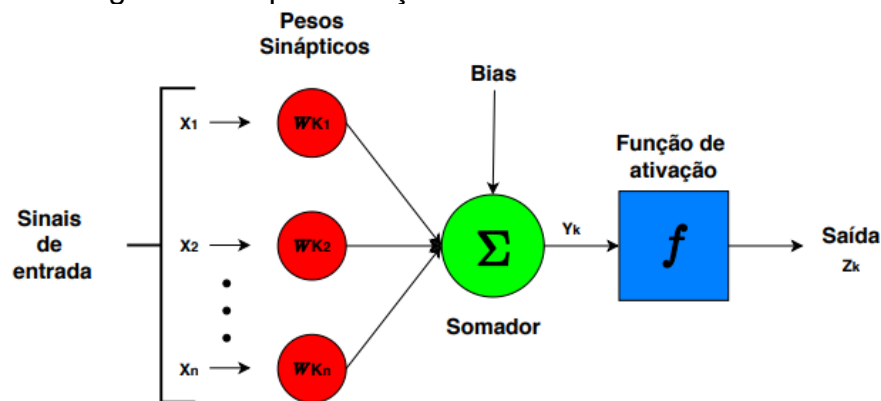
Fonte: PRÓPRIO (2020).

Com a *multilayer perceptron* é possível utilizar múltiplas classificações na camada de saída (GARDNER; DORLING, 1998). Por se tratar de uma rede *feedforward*, o fluxo do processo sináptico dos neurônios ocorre a partir da camada de entrada, passando pelas camadas ocultas, chegando então na camada de saída, sempre respeitando esta ordem.

A principal característica de uma RNA é a capacidade de aprender de acordo com o ambiente inserido e melhorar seu desempenho através do processo de treinamento (HAYKIN, 2001). O método mais utilizado no processo de treinamento de redes neurais é o algoritmo de *backpropagation*. Com este algoritmo, os pesos sinápticos de cada neurônio são recalculados com base no erro obtido pela comparação entre a saída gerada de cada neurônio com o resultado esperado (aprendizado supervisionado). Com os novos pesos, o treino é realizado novamente, recalculando novamente os pesos se necessário, até que a taxa de erro seja inferior a um número pré-estabelecido.

A Figura 3 apresenta de maneira simplificada um modelo de Neurônio Artificial, em que o neurônio  $k$  recebe uma entrada  $x$  que entra pela sinapse  $j$

Figura 3 - Representação de um Neurônio Artificial.



Fonte: Adaptado de HAYKIN, 2001.

Cada sinal de entrada representada pela variável  $X_j$  é multiplicada por um peso  $W_{kj}$ , que na criação do neurônio é gerado com um valor randomizado. O resultado desta operação passa por uma função que soma os sinais de entrada ponderados pelos pesos sinápticos com um *bias* externo ( $Y_k$ ). O *bias* possui a responsabilidade de aumentar ou diminuir o valor resultante do somador, antes que

este valor seja repassado à função de ativação. Esta função recebe o valor passado pelo somador e transforma em um intervalo fechado geralmente entre  $[0,1]$  ou  $[-1,1]$ , que conseqüentemente é repassado aos outros neurônios (HAYKIN, 2001). A função de ativação é responsável por definir a saída do neurônio, dado uma entrada. Esta saída é definida de acordo com o resultado do somatório dos sinais de entrada ponderados.

### 3 ESTADO DA ARTE

Esta Seção possui como objetivo contextualizar o estado atual das pesquisas e estudos voltados à detecção de intrusão em dispositivos IoT. Para seleção do material, foi realizada uma breve revisão bibliográfica, selecionando artigos escritos em língua inglesa e portuguesa, publicados nas fontes: *IEEE*, *ACM*, *Elsevier* e *Springer*. Além disso, foram considerados também trabalhos de dissertação na área em questão.

No trabalho da autora Jafier (2018) foi realizada uma breve contextualização do panorama atual a respeito da tecnologia IoT e dos sistemas de detecção de intrusão existentes. Como objetivo principal, apresenta um sistema de detecção de intrusão para dispositivos IoT baseado em *Data Mining*, para ser utilizado em ambientes de rede sem fio. Segundo a autora, alguns algoritmos de *Data Mining* como *clustering* e classificação já são utilizados em sistemas de detecção de intrusão. Na abordagem proposta, ela adota a utilização do algoritmo ID3 como classificador em conjunto com outros algoritmos buscando a otimização do sistema de detecção de intrusão. As técnicas de detecção de intrusão baseadas em *Data Mining* se dividem em dois grupos: detecção por anomalia e por abuso. Na detecção por abuso, o sistema seleciona um método suspeito e realiza a comparação com métodos intrusivos já conhecidos, a fim de realizar a detecção e identificar o método suspeito como intrusivo. O modelo de sistema proposto pela autora recebe como entrada conexões de qualquer ambiente de rede (com fio, sem fio e objetos inteligentes), e como saída classifica a conexão como normal ou intrusiva, para esta última, sendo especificado o tipo de intrusão. Como resultado, o sistema se mostrou capaz de descobrir informações sobre os ataques e as utiliza para construir o conjunto de dados final para treinar e testar o classificador ID3, tornando-o mais eficiente na detecção de intrusão.

Na dissertação de mestrado de Souza (2018), o autor inicia comentando a respeito do rápido crescimento tecnológico que vivenciamos no momento atual, citando a grande quantidade de dados gerados por novas tecnologia e seus principais desafios, sendo um deles, a segurança. Com essa crescente quantidade

de dados gerados, é possível observar também o desenvolvimento de novas e sofisticadas técnicas de ataques à sistemas. Com o aumento da preocupação em relação a segurança e privacidade dos dados dos usuários, o estudo na área de detecção de intrusão vem ganhando destaque, com o intuito de desenvolver novas e melhores técnicas a fim de evitar ataques nos meios computacionais. O resultado da utilização de métodos híbridos de detecção de intrusão, com o auxílio de técnicas de inteligência artificial vêm se provando cada vez mais satisfatórios do que a utilização de apenas uma destas abordagens, individualmente. Nesta dissertação, o autor apresenta um método híbrido de detecção de intrusão utilizando o método de classificação *K-Nearest Neighbors* (KNN), muito conhecido pela acurácia na detecção de eventos intrusivos e não intrusivos, porém, com um tempo de execução elevado na etapa de classificação. Outra técnica utilizada neste trabalho foram Redes Neurais Artificiais (RNA), que possuem um tempo de execução melhor, quando comparado ao KNN, porém, com acurácia de detecção inferior. O objetivo desta proposta é melhorar a taxa de detecção do método RNA, que possui menor necessidade de processamento do que o KNN na etapa de classificação. Em contrapartida, a função do KNN é determinar a classe de um elemento a partir de uma correlação com exemplos pré-estabelecidos. Com a melhoria da acurácia do método RNA é possível obter um método de detecção de intrusão mais eficiente, utilizando estas duas técnicas em conjunto, extraindo o melhor de cada uma. Para comprovação da eficiência desta abordagem, foram realizados diversos testes de eficiência com os métodos RNA e KNN em conjunto, e com cada uma destas técnicas sendo utilizadas individualmente, a fim de gerar uma base de resultados permitindo a comparação entre os diferentes modelos testados. Como conclusão, o autor cita que o método híbrido proposto e apresentado neste trabalho obteve desempenho superior ao da técnica RNA em termos de acurácia em todas as bases de dados utilizadas nos testes. Outra conclusão que se obteve, foi de que as taxas de acurácia do método híbrido foram equivalentes ao do método KNN, porém, levou vantagem em relação ao tempo de processamento, apresentando melhor eficiência. Por fim, esta tese comprova que o uso híbrido destas técnicas na detecção de intrusão é superior ao uso individual de cada uma delas.

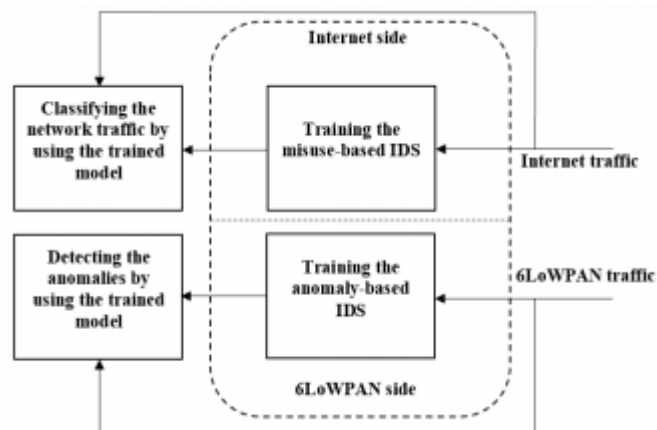
No trabalho de Al-hawawreh, Moustafa e Sitnikova (2018), os autores propõe uma técnica de detecção por anomalia para sistemas de controle industrial. Devido a deficiência dos sistemas de detecção existentes na parte de coleta de informação e utilização desta para aperfeiçoamento próprio, a técnica apresentada pelos autores é baseada em modelos de *deep learning*, capazes de aprender e validar a partir de informações obtidas em pacotes TCP/IP. Para confecção desta técnica, são utilizados processos de treinamento de redes neurais com auxílio de dois *datasets* famosos, o NSL-KDD e o UNSW-NB15. Os métodos de *deep learning* utilizados neste trabalho foram o *deep auto-encoder*, utilizado na fase de treinamento do sistema de detecção e responsável por criar os parâmetros de inicialização utilizados no treinamento da *deep feedforward neural network*, outra técnica de *deep learning* utilizada no trabalho. Esta técnica tem como objetivo descobrir novos ataques, além dos pré-existentes. Consiste em uma rede neural com uma camada de entrada, mais de uma oculta e uma de saída. As camadas escondidas neste modelo são responsáveis por analisar e descobrir informações a partir dos dados inseridos na camada de entrada, gerando um resultado apresentado na camada de saída. Os autores utilizam ainda um algoritmo de *backpropagation*, utilizado no cálculo da performance do resultado obtido e na propagação deste valor para as camadas ocultas, onde os pesos inicialmente fornecidos pelo *deep auto-encoder* são atualizados, visando sempre a melhora da acurácia do modelo. Como resultado, os autores conseguiram uma acurácia de 98.6 e 92.4, taxa de detecção de 99 e 93, e taxa de falso positivo de 1.8 e 8.2, para os *datasets* NSL-KDD e UNSW-NB15, respectivamente, comprovando a eficiência da técnica desenvolvida quando comparada com outros métodos existentes, de acordo com as tabelas comparativas apresentadas pelos autores.

No trabalho de Sheikhan e Bostani (2016), os autores propõe uma abordagem de detecção de intrusão por anomalia, capaz de detectar ataques conhecidos ou não, e detecção por abuso, conhecida por reconhecer os ataques pela sua assinatura. O modelo proposto é baseado no modelo de programação *MapReduce*, que processa grandes quantidades de dados em paralelo, possibilitando a detecção distribuída. O método de classificação utilizada foi o



*Optimum-Path Forest*, e o *dataset* utilizado para treinamento e teste deste classificador foi o NSL-KDD. O modelo proposto consiste em dois módulos, o da *Internet*, utilizado na detecção por abuso, e do *6LoWPAN*, responsável pela detecção por anomalia, conforme demonstrado na Figura 4.

Figura 4 - Modelo proposto.



Fonte: SHEIKHAN; BOSTANI, 2016.

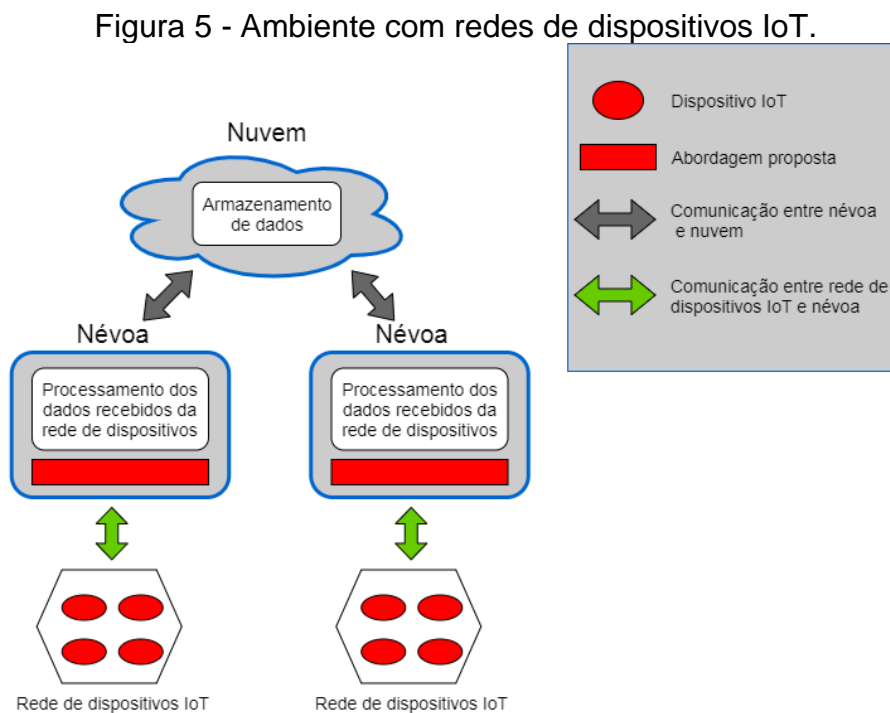
Devido a estrutura do modelo proposto, foi possível realizar a detecção de intrusão paralela. Após realização de testes com o *dataset* escolhido, os autores obtiveram para o módulo de detecção por anomalia uma taxa de detecção de 80.95 e taxa de alarme falso de 5.92. No módulo de detecção por abuso, os valores encontrados foram de 96.20 e 1.44. Os resultados obtidos demonstraram uma performance superior na detecção simultânea de ataque internos e cibernéticos no ambiente da Internet das Coisas.

#### 4 PROPOSTA DE DESENVOLVIMENTO DO TRABALHO

Neste trabalho é proposta uma abordagem de detecção e identificação de intrusão para dispositivos IoT utilizando redes neurais artificiais.

Como já mencionado neste trabalho, os dispositivos inteligentes possuem recursos limitados, com pouca capacidade de processamento (YANNUZZI et al., 2014). Por este motivo, não é possível utilizar um método de detecção de intrusão tradicional por anomalia, assim como a abordagem não pode atuar diretamente nos dispositivos IoT. Como a *fog* é utilizada no processamento dos dados destes dispositivos, atuando mais próximo da rede, obtendo assim menor latência em comparação a *cloud*, a abordagem proposta está situada no nível da *fog* e é responsável por monitorar a rede de dispositivos IoT.

A proposta consiste em uma abordagem de detecção e identificação de intrusão baseado em Redes Neurais Artificiais, treinada com uma base de dados contendo diversos tipos de ataques, possibilitando a detecção e identificação dos pacotes enviados pelos dispositivos IoT à *fog*. A Figura 5 ilustra um ambiente com dispositivos IoT com a abordagem proposta inserida no nível da *fog*.

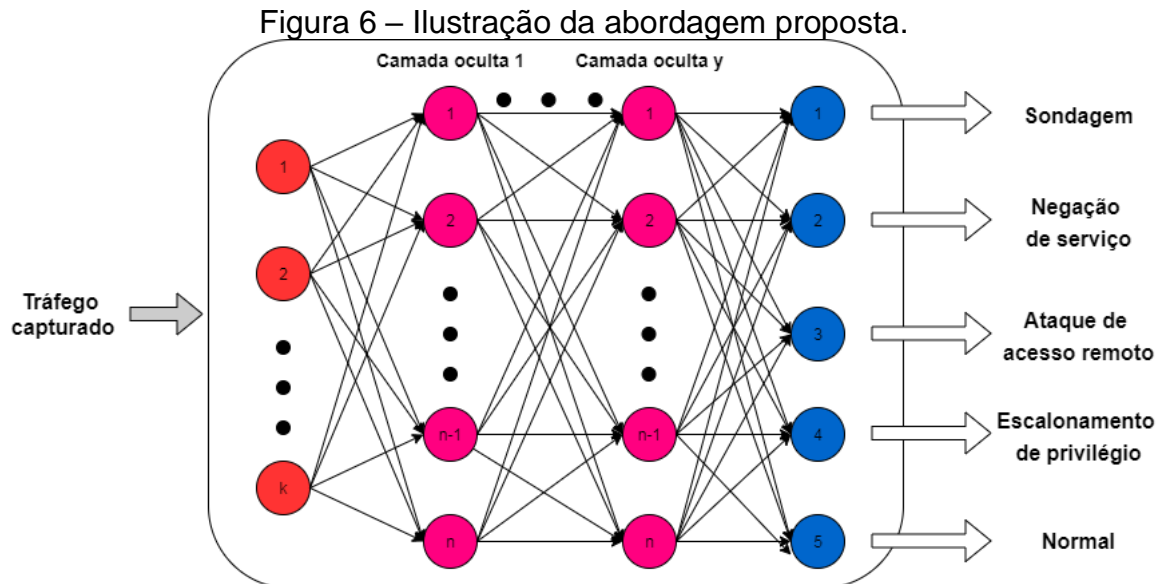


Fonte: PRÓPRIO (2020).

A abordagem proposta atua na névoa capturando os pacotes enviados pelos dispositivos IoT, analisando cada pacote e classificando-os entre 5 classes diferentes, sendo uma classe para eventos não intrusivos, chamada de *normal*, e quatro classes de ataques, sendo estas:

- Sondagem (*Probe*): ataque que visa obter informações a respeito do sistema atacado, como endereço IP, sistema operacional utilizado entre outras informações confidenciais. É um tipo de ataque que como o próprio nome diz, sonda o alvo, capturando informações a respeito do mesmo, informações estas que futuramente podem ser utilizadas em ataques mais intrusivos, como ataques negação de serviço (KHAMPHAKDEE; BENJAMAS; SAIYOD, 2014).
- Negação de serviço (*Denial of Service - DoS*): possui como objetivo tornar os serviços do sistema alvo indisponíveis, através do envio de pacotes maliciosos. Este ataque pode fazer com que os pacotes maliciosos consumam todos os recursos disponíveis do sistema, tornando-o inútil, ou então o ataque pode ainda obstruir a comunicação entre o sistema alvo e os utilizadores dele (JHAVERI; PATEL; JINWALA, 2012).
- Ataque de acesso remoto (*Remote to Local - R2L*): o atacante tenta obter acesso ao usuário local do sistema remoto através de alguma vulnerabilidade do mesmo. Este ataque precede ao ataque de escalonamento de privilégio (ALHARBI; ALHAIDARI; ZOHDY, 2018).
- Escalonamento de privilégio (*User to Root - U2R*): neste tipo de ataque, o usuário malicioso que já conquistou acesso ao sistema remoto busca obter acesso a recursos privilegiados, ou seja, recursos que aquele não deveria ter acesso, permitindo que ele possa executar *malwares*, tarefas de administrador, entre outras funções que podem comprometer o sistema (JEYA; MURALIDHARAN; RAVICHANDRAN, 2012).

A Figura 6 ilustra a abordagem proposta, responsável por analisar o tráfego obtido na entrada e fornecer uma classificação na camada de saída.



Fonte: PRÓPRIO (2020).

Conforme observado na Figura 6, a RNA possui uma camada de entrada, com  $k$  neurônios, valor este que deve ser equivalente ao número de atributos que cada amostra da base de dados possui. O número de camadas ocultas  $y$  é variável e será testado com diferentes valores, assim como o número de neurônios  $n$  em cada camada oculta. A alteração destes valores tem como objetivo investigar a influência do aumento do número de camadas ocultas e de neurônios na detecção das classes estabelecidas. Para tal, os processos de treinamento e teste da RNA serão realizados com diferentes arquiteturas. Os neurônios das camadas ocultas, responsáveis pelo aprendizado da rede, utilizam a função de ativação *ReLU*, que trabalha como uma função linear quando os valores de entrada repassados pelo somador são maiores que 0, dando como saída da função o mesmo valor recebido na entrada. Caso o valor do *input* seja negativo, ela se comporta como uma função não linear, pois transforma estas entradas em 0, e este valor é considerado no *output* do neurônio. Esta função de ativação se tornou a função padrão utilizada para diversos tipos de Redes Neurais, pois os modelos que a utilizam são mais fáceis de treinar e normalmente atingem melhores resultados. A camada de saída conta com 5

neurônios, um para cada classe: sondagem, negação de serviço, ataque de acesso remoto, escalonamento de privilégio e normal. Estas classes foram definidas de acordo com os tipos de ataques presentes na base de dados utilizada. Estes neurônios utilizam a função de ativação *softmax*, que gera um valor de classificação entre 0 e 1 para cada neurônio. Com esta função, as saídas para cada uma das classes são transformadas em valores entre 0 e 1, dando a probabilidade de uma determinada entrada pertencer a cada uma das classes.

## 5 AVALIAÇÃO

Nesta Seção é descrita a metodologia utilizada na realização dos experimentos. Além disso, são apresentados os resultados obtidos pela abordagem proposta. Por fim, uma discussão sobre os resultados é apresentada.

### 5.1 EXPERIMENTOS

Nesta Seção são apresentados os experimentos realizados para avaliar a abordagem proposta. São apresentados detalhes sobre a base de dados utilizada, a técnica de *cross-validation* e as métricas de avaliação consideradas.

#### 5.1.1 Base de Dados

A base de dados utilizada no treinamento da abordagem proposta foi a NSL-KDD. Este conjunto de dados é uma versão melhorada do KDDCUP1999, outro conjunto de dados de referência para detecção de intrusão (TAVALLAEE et al., 2009). O conjunto de treino do NSL-KDD consiste em 125973 amostras, enquanto o conjunto de teste possui 22544 amostras. Este *data set* abrange a maioria das categorias de ataques, entre elas: *Probe*, *DoS*, *U2R* e *R2L*. Nesta abordagem, são trabalhados apenas os dados referentes ao conjunto de treino da NSL-KDD, com o auxílio do método *k-fold cross-validation* com 10 *folds*, ou seja, as 125973 amostras são divididas em 10 subconjuntos, onde cada um destes subconjuntos é utilizado uma vez para teste, enquanto os demais são utilizados na etapa de treinamento.

Na Tabela 1 é possível observar o número de amostras presentes no conjunto de treino da base de dados NSL-KDD, categorizadas em 5 classes, sendo elas: *Normal*, *DoS*, *Probe*, *R2L* e *U2R*.

Tabela 1 - Amostras do conjunto de treino da NSL-KDD.

	<b>Total</b>	<b>Normal</b>	<b>DoS</b>	<b>Probe</b>	<b>R2L</b>	<b>U2R</b>
NSL-KDD <i>Train</i>	125973	67343	45927	11656	995	52

Fonte: (ALJAWARNEH; ALDWAIRI; YASSEIN, 2018).

É importante observar a baixa quantidade de amostras referentes a R2L e U2R quando comparados com as outras classes. Este pode ser um fator que influencie na eficiência da abordagem proposta, onde o menor número de amostras implique em um aprendizado limitado das Redes Neurais, influenciando diretamente no resultado das classificações.

No Quadro 1 são listados os atributos presente na base NSL-KDD. Estes são os atributos utilizados no processo de aprendizagem das Redes Neurais utilizadas na abordagem proposta. Cada atributo tem seu peso calculado e recalculado iterativamente na abordagem, permitindo inferências entre o valor de cada atributo com a classe a qual ele pertence.

Quadro 1 - Atributos da base NSL-KDD.

Número	Atributo	Descrição
1	<i>duration</i>	Duração da conexão
2	<i>protocol_type</i>	Protocolo de comunicação (ex: <i>TCP</i> , <i>UDP</i> , <i>ICMP</i> )
3	<i>service</i>	Serviço de destino
4	<i>flag</i>	<i>Flag</i> de <i>status</i> da conexão
5	<i>src_bytes</i>	<i>Bytes</i> enviados da fonte para destino.
6	<i>dst_bytes</i>	<i>Bytes</i> enviados d destino para fonte
7	<i>land</i>	1 se a conexão é de/para o mesmo <i>host/port</i> ; 0 caso contrário
8	<i>wrong_fragment</i>	Número de fragmentos errados
9	<i>urgent</i>	Número de pacotes urgentes
10	<i>hot</i>	Número de indicadores “ <i>hot</i> ”
11	<i>num_failed_logins</i>	Número de <i>logins</i> com falha
12	<i>logged_in</i>	1 se logado com sucesso; 0 caso contrário
13	<i>num_compromised</i>	Número de condições “comprometidas”
14	<i>root_shell</i>	1 se o <i>root shell</i> é obtido; 0 caso contrário
15	<i>su_attempted</i>	1 se o comando “ <i>su root</i> ” for tentado; 0 caso contrário
16	<i>num_root</i>	Número de acessos no “ <i>root</i> ”
17	<i>num_file_creations</i>	Número de operações de criação de arquivo

18	<i>num_shells</i>	Número de <i>prompts</i> de <i>shell</i>
19	<i>num_access_files</i>	Número de operações em arquivos de controle de acesso
20	<i>num_outbound_cmds</i>	Número de comandos de saída em uma sessão <i>ftp</i>
21	<i>is_host_login</i>	1 se o <i>login</i> pertence à lista “ <i>host</i> ”; 0 caso contrário
22	<i>is_guest_login</i>	1 se o <i>login</i> pertence à lista “ <i>guest</i> ”. 0 caso contrário
23	<i>count</i>	Número de conexões ao mesmo <i>host</i> que a conexão atual nos últimos 2 segundos
24	<i>srv_count</i>	Número de conexões ao mesmo <i>serviço</i> que a conexão atual nos últimos 2 segundos
25	<i>serror_rate</i>	Número de conexões com erros “ <i>SYN</i> ”
26	<i>srv_serror_rate</i>	Número de conexões com erros “ <i>SYN</i> ”
27	<i>rerror_rate</i>	Número de conexões com erros <i>REJ</i>
28	<i>srv_rerror_rate</i>	Número de conexões com erros <i>REJ</i>
29	<i>same_srv_rate</i>	Número de conexões para o mesmo <i>serviço</i>
30	<i>diff_srv_rate</i>	Número de conexões para diferentes <i>serviços</i>
31	<i>srv_diff_host_rate</i>	Número de conexões para diferentes <i>hosts</i>
32	<i>dst_host_count</i>	Contagem de conexões com o mesmo <i>host</i> de destino
33	<i>dst_host_srv_count</i>	Contagem de conexões com o mesmo <i>host</i> de destino e usando o mesmo <i>serviço</i>
34	<i>dst_host_same_srv_rate</i>	Número de conexões com o mesmo <i>host</i> de destino e usando o mesmo <i>serviço</i>
35	<i>dst_host_diff_srv_rate</i>	Número de diferentes <i>serviços</i> no <i>host</i> atual
36	<i>dst_host_same_src_port_rate</i>	Número de conexões com o <i>host</i> atual com a mesma porta <i>src</i>
37	<i>dst_host_srv_diff_host_rate</i>	Número de conexões para o mesmo <i>serviço</i> vindo de <i>hosts</i> diferentes
38	<i>dst_host_serror_rate</i>	Número de conexões com o <i>host</i> atual que têm um erro <i>S0</i>
39	<i>dst_host_srv_serror_rate</i>	Número de conexões com o <i>host</i> atual e <i>serviço</i> especificado que têm um erro <i>S0</i>
40	<i>dst_host_rerror_rate</i>	Número de conexões com o <i>host</i> atual que têm um erro <i>RST</i>
41	<i>dst_host_srv_rerror_rate</i>	Número de conexões com o <i>host</i> atual e <i>serviço</i> especificado que têm um erro <i>RST</i>

Fonte: (JEYA; MURALIDHARAN; RAVICHANDRAN, 2012).



### 5.1.2 Métricas de Avaliação

Uma das principais etapas na validação da abordagem proposta é a utilização de métricas para compreender o desempenho obtido nos experimentos. Para realizar o cálculo das métricas utilizadas, foram geradas matrizes de confusão contendo os seguintes valores:

- **Falsos Positivos (FP):** Eventos normais classificados como intrusivos pela abordagem proposta;
- **Falsos Negativos (FN):** Eventos intrusivos classificados como normais pela abordagem proposta;
- **Positivos Verdadeiros (PV):** Eventos intrusivos classificados corretamente pela abordagem proposta;
- **Negativos Verdadeiros (NV):** Eventos normais classificados corretamente pela abordagem.

As métricas utilizadas na avaliação deste trabalho podem ser observadas a seguir (ALMIANI *et al.*, 2019):

1. **Acurácia (ACC):** métrica que apresenta a proporção de eventos classificados corretamente sobre o número total de classificações. A acurácia é ser calculada através da Equação 5.1, apresentada abaixo:

$$ACC = \frac{PV + NV}{FP + FN + PV + NV} \quad (5.1)$$

2. **Erro (ERR):** apresenta o percentual de eventos classificados incorretamente sobre o número total de classificações. O erro é calculado pela Equação 5.2.

$$ERR = \frac{FP + FN}{FP + FN + PV + NV} \quad (5.2)$$

3. **Precisão (PRE):** métrica bastante utilizada para avaliação de algoritmos de *machine learning*. Apresenta a taxa de eventos classificados corretamente como intrusivos dentre todos os eventos classificados como intrusivos. É obtida através da Equação 5.3:

$$PRE = \frac{PV}{FP + PV} \quad (5.3)$$

4. **Recall:** apresenta a taxa de eventos classificados corretamente como intrusivos dentre todos os eventos intrusivos do conjunto de dados. É calculada pela Equação 5.4:

$$Recall = \frac{PV}{PV + FN} \quad (5.4)$$

5. **True Negative Rate (TNR):** apresenta a taxa de eventos classificados não intrusivos entre todos os eventos não intrusivos. É calculada a partir da Equação 5.5:

$$TNR = \frac{NV}{NV + FP} \quad (5.5)$$

6. **F1-Score:** esta métrica apresenta uma média harmônica de precisão e *recall*, onde o melhor valor é 1, ou seja, boas taxas de precisão e *recall*, e o pior valor é igual a 0. É calculada pela Equação 5.6:

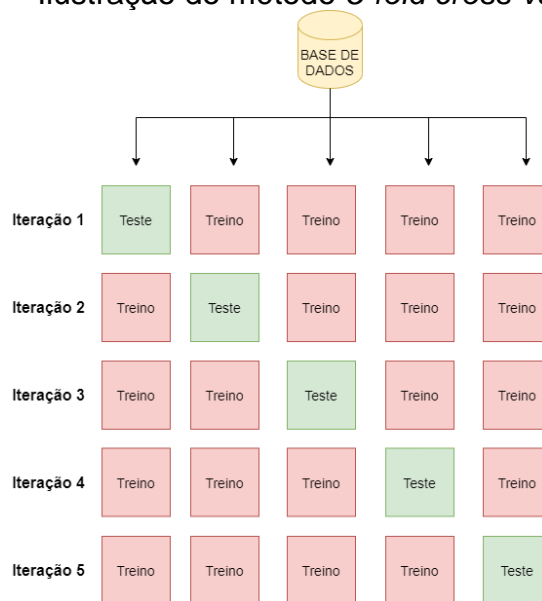
$$F1\text{-SCORE} = 2 * \frac{\text{Precisão} * \text{Recall}}{\text{Precisão} + \text{Recall}} \quad (5.6)$$

### 5.1.3 Cross-validation

O *Cross-validation* é uma técnica utilizada para avaliar modelos de aprendizado de máquina dado uma base de dados finita, com o objetivo de estimar o quão preciso o modelo é na prática. Neste trabalho é utilizado o método *k-fold cross-validation*. O objetivo deste método é particionar a base de dados de treino da NSL-KDD em  $k$  subconjuntos de tamanhos iguais. Em cada iteração do método, um subconjunto é utilizado para testar o modelo proposto, enquanto os demais grupos são utilizados na etapa de treinamento do modelo, onde são estimados os parâmetros. O método realiza  $k$  iterações, alternando em cada uma delas o subconjunto utilizado para teste, até que todos os  $k$  subconjuntos sejam utilizados. Este procedimento permite que o método *k-fold cross-validation* avalie o desempenho do modelo, calculando a acurácia sobre os erros encontrados.

Na Figura 7 é ilustrado de maneira simplificada o funcionamento do método, neste caso, utilizando 5 *folds*, onde a base de dados é dividida em 5 subconjuntos de tamanhos iguais e as iterações acontecem da maneira supracitada.

Figura 7 - Ilustração do método *5-fold cross-validation*.

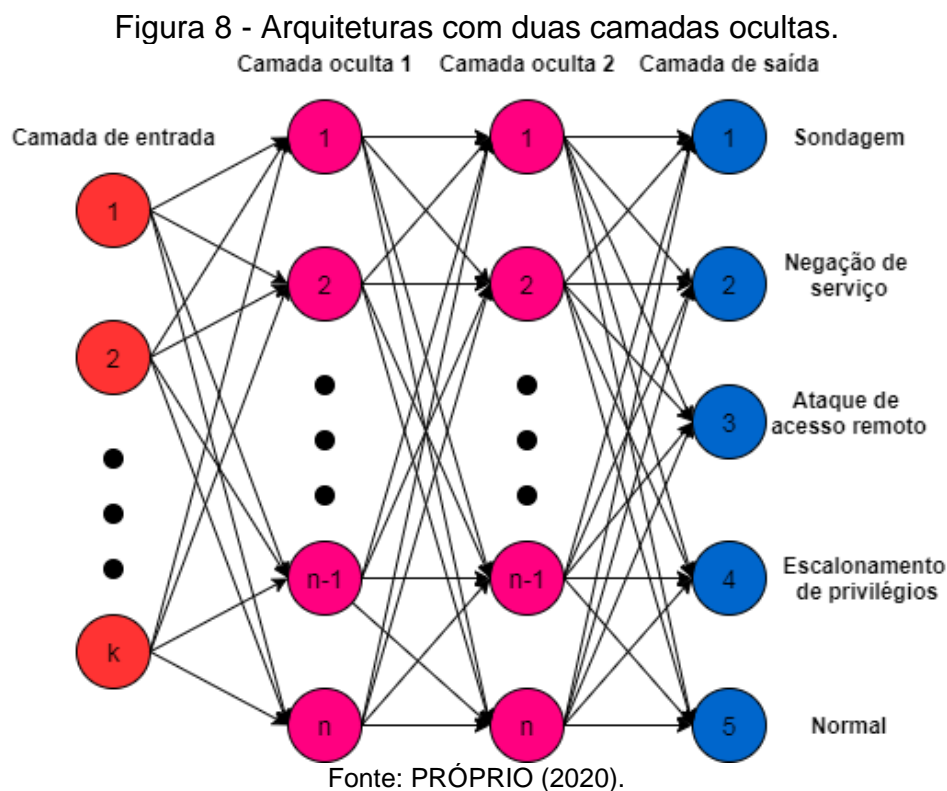


Fonte: PRÓPRIO (2020).

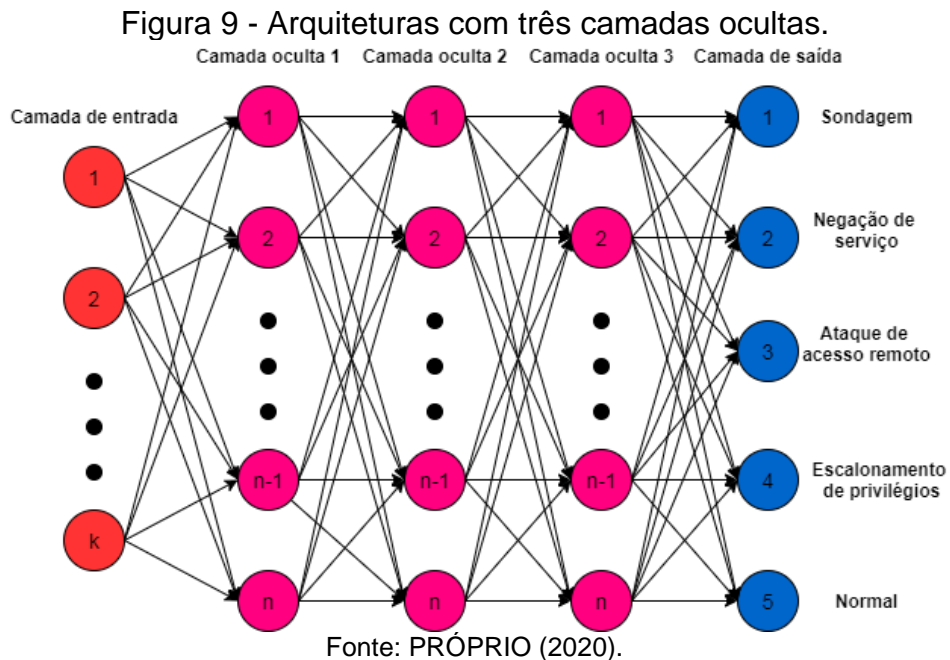
### 5.1.4 Arquiteturas Utilizadas

Para fim de experimentos, foram utilizadas oito diferentes arquiteturas na execução dos testes e avaliação da abordagem proposta. Estas arquiteturas se diferem pela quantidade de camadas ocultas que a Rede Neural Artificial tem, bem como o número de neurônios em cada uma destas camadas. Em todas as arquiteturas, as camadas de entrada e saída possuem uma quantidade fixa de neurônios, 41 e 5, respectivamente, respeitando sempre os 41 atributos da base de dados utilizados na camada de entrada, e os 5 diferentes tipos de classificação, correspondente a camada de saída.

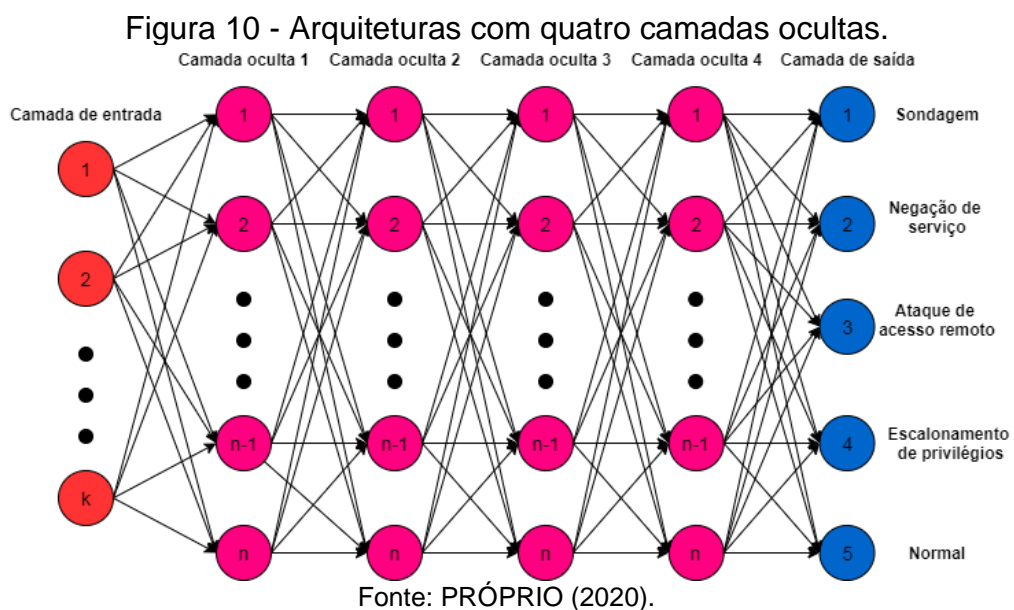
A Figura 8 representa as arquiteturas mais simples utilizadas nos experimentos. Estas possuem apenas duas camadas ocultas, uma com o número de neurônios por camada oculta  $n$  igual a 41, sendo esta a arquitetura 2C41N, e outra com o número de neurônios igual a 82, chamada de 2C82N. Em ambas as arquiteturas, o número de neurônios na camada de entrada  $k$  é igual a 41, de acordo com o número de atributos da base utilizada.



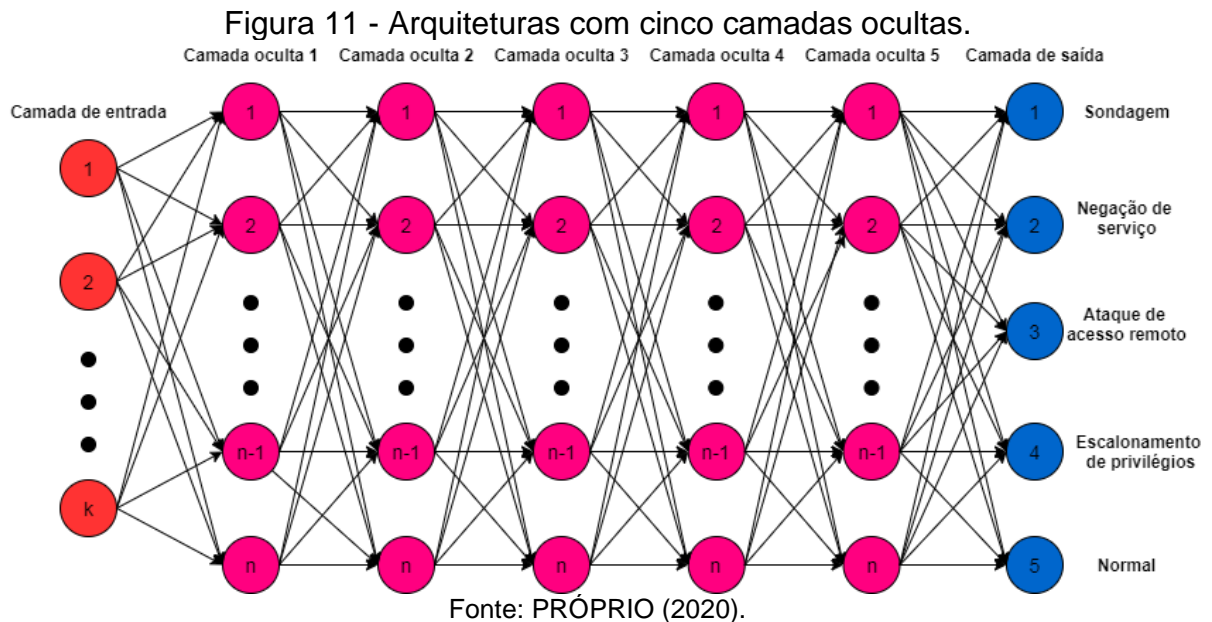
A Figura 9 representa as arquiteturas com duas camadas ocultas, a 3C41N, para  $n=41$ , e a 3C82N, para  $n=82$ . Em ambas as arquiteturas o valor de  $k$  é igual a 41.



A Figura 10 representa as arquiteturas com duas camadas ocultas, a 4C41N, para  $n=41$ , e a 4C82N, para  $n=82$ . Em ambas as arquiteturas o valor de  $k$  é igual a 41.



A Figura 11 representa as arquiteturas com duas camadas ocultas, a 5C41N, para  $n=41$ , e a 5C82N, para  $n=82$ . Em ambas as arquiteturas o valor de  $k$  é igual a 41.



O Quadro 2 apresenta a nomenclatura das arquiteturas, de acordo com a composição das mesmas.

Quadro 2 - Nomenclatura das arquiteturas.

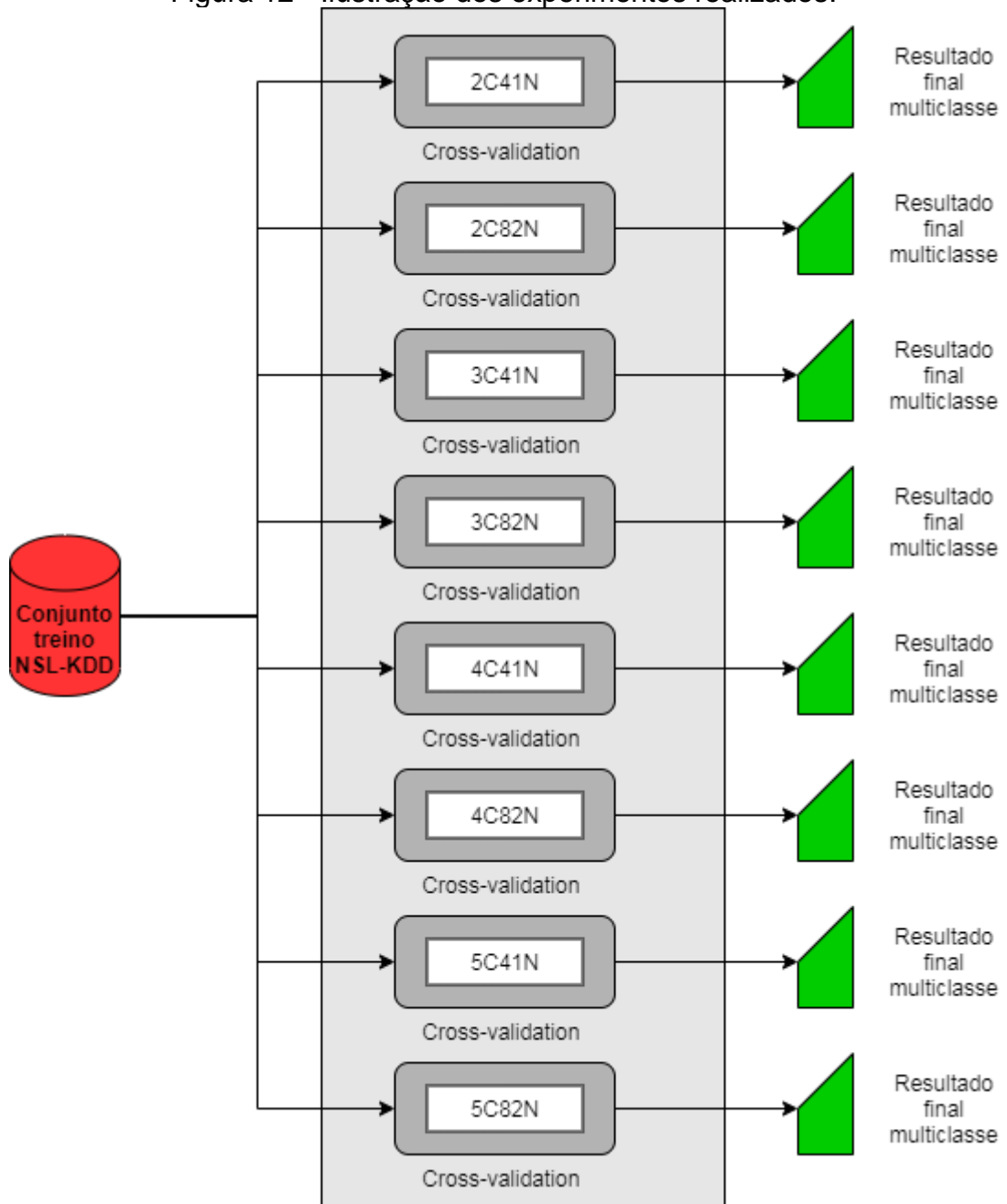
Abreviação	Nomenclatura
2C41N	2 camadas ocultas com 41 neurônios cada
2C82N	2 camadas ocultas com 82 neurônios cada
3C41N	3 camadas ocultas com 41 neurônios cada
3C82N	3 camadas ocultas com 82 neurônios cada
4C41N	4 camadas ocultas com 41 neurônios cada
4C82N	4 camadas ocultas com 82 neurônios cada
5C41N	5 camadas ocultas com 41 neurônios cada
5C82N	5 camadas ocultas com 82 neurônios cada

Fonte: PRÓPRIO (2020).

Na Figura 12, temos uma demonstração de como o experimento foi realizado, submetendo o conjunto de treino da base NSL-KDD às arquiteturas pré-definidas, responsáveis pelo aprendizado e classificação dos eventos. Também temos o auxílio do método *cross-validation* com 10 *folds* para divisão do conjunto de

treino em subgrupos, permitindo o teste e avaliação da abordagem proposta com diferentes subconjuntos de dados.

Figura 12 - Ilustração dos experimentos realizados.



Fonte: PRÓPRIO (2020).

### 5.1.5 Ferramentas Utilizadas

Para configuração do ambiente dos experimentos, foi utilizada a ferramenta *Google Colaboratory*, por onde foi possível implementar e executar a abordagem proposta. A máquina oferecida por este serviço em nuvem já vem pré-configurada, com as seguintes especificações, conforme as Figuras 13 e 14:

Figura 13 - Especificações do processador

```
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 63
model name    : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping      : 0
microcode     : 0x1
cpu MHz       : 2300.000
cache size    : 46080 KB
physical id   : 0
siblings      : 2
core id       : 0
cpu cores     : 1
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception : yes
cpuid level   : 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush
bugs          : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs
bogomips      : 4600.00
clflush size  : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

processor      : 1
vendor_id     : GenuineIntel
cpu family    : 6
model         : 63
model name    : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping      : 0
microcode     : 0x1
cpu MHz       : 2300.000
cache size    : 46080 KB
physical id   : 0
siblings      : 2
core id       : 0
cpu cores     : 1
apicid        : 1
initial apicid : 1
fpu           : yes
fpu_exception : yes
cpuid level   : 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush
bugs          : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs
bogomips      : 4600.00
clflush size  : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

Fonte: PRÓPRIO (2020).



Figura 14 - Especificações da memória

```

MemTotal:      13333552 kB
MemFree:       10835156 kB
MemAvailable:  12539864 kB
Buffers:       73332 kB
Cached:        1779776 kB
SwapCached:    0 kB
Active:        640888 kB
Inactive:      1596464 kB
Active(anon):  352812 kB
Inactive(anon): 340 kB
Active(file):  288076 kB
Inactive(file): 1596124 kB
Unevictable:   0 kB
Mlocked:       0 kB
SwapTotal:     0 kB
SwapFree:      0 kB
Dirty:         136 kB
Writeback:     0 kB
AnonPages:     384388 kB
Mapped:        206152 kB
Shmem:         940 kB
Slab:          160296 kB
SReclaimable: 122676 kB
SUnreclaim:    37620 kB
KernelStack:   3780 kB
PageTables:    5320 kB
NFS_Unstable:  0 kB
Bounce:        0 kB
WritebackTmp:  0 kB
CommitLimit:   6666776 kB
Committed_AS:  2641740 kB
VmallocTotal:  34359738367 kB
VmallocUsed:   0 kB
VmallocChunk:  0 kB
Percpu:        928 kB
AnonHugePages: 0 kB
ShmemHugePages: 0 kB
ShmemPmdMapped: 0 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize:  2048 kB
Hugetlb:       0 kB
DirectMap4k:   87228 kB
DirectMap2M:   6203392 kB
DirectMap1G:   9437184 kB

```

Fonte: PRÓPRIO (2020).

Para criação e execução da abordagem proposta, foram utilizadas algumas bibliotecas *Python*. A biblioteca *panda* foi utilizada no carregamento da base de dados NSL-KDD, armazenando-a em uma variável. A biblioteca *NumPy*, que é utilizada para manipulação de *arrays* e matrizes, serviu para agrupar os ataques da base de dados em 5 classes diferentes. Também foi utilizada a *Scikit-Learn*, uma biblioteca de aprendizado de máquina, através da qual foi possível utilizar o método de *cross-validation*, realizar os cálculos das métricas utilizadas na avaliação, entre outras tarefas. Para a RNA, foi utilizada a biblioteca *Keras*, que permite a manipulação e configuração de algumas características da RNA, como por exemplo, o número de camadas. Esta biblioteca se trata de uma interface de alto nível para a biblioteca *TensorFlow*, que é a responsável pela criação e execução da RNA.

## 5.2 RESULTADOS

Nesta Seção são apresentados os resultados dos experimentos realizados com as arquiteturas propostas, demonstrando percentualmente a taxa obtida pelas arquiteturas em cada uma das métricas de avaliação definidas na Seção 5.1.2. Posteriormente, é realizada uma análise refletindo sobre os resultados obtidos e o motivo das variações observadas, procurando identificar qual das arquiteturas propostas foi a mais eficiente.

A Tabela 2 apresenta os resultados dos experimentos com a arquitetura 2C41N.

Tabela 2 - Resultados dos experimentos com a arquitetura 2C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	95,66	4,34	95,16	96,81	94,34	95,96
DoS	97,94	2,06	96,77	97,63	98,12	97,19
probe	98,11	1,89	95,68	84,08	99,55	89,02
R2L	98,99	1,01	23,94	20,12	99,57	21,24
U2R	99,96	0,04	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Os resultados obtidos na detecção das classes *normal*, *DoS* e *probe* se mostraram satisfatórios. Já os resultados obtidos para as classes *R2L* e *U2R* demonstraram uma deficiência da arquitetura na classificação destes ataques, levando em consideração os resultados ruins obtidos no *recall*. O *recall* de cada uma destas classes apresenta a taxa de eventos classificados corretamente como intrusivos (neste caso *R2L* e *U2R*) dentre todos os eventos intrusivos (*R2L* e *U2R*) do conjunto de dados. Então, mesmo que estas classes tenham apresentado acurácias com valores altos, esta arquitetura apresentou grande deficiência na classificação destes ataques, em especial do *U2R*, que obteve o *recall* igual a 0. O TNR, taxa de eventos classificados como não intrusivos dentre todos os eventos não intrusivos, obteve bom resultado para todas as classes. O F1-Score, que representa uma média harmônica entre precisão e *recall*, apresentou resultados ruins para as classes *R2L* e *U2R*, devido à dificuldade de detecção das mesmas, conforme mencionado anteriormente.

A Tabela 3 apresenta os resultados dos experimentos com a arquitetura 2C82N.

Tabela 3 - Resultados dos experimentos com a arquitetura 2C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	95,57	4,43	96,09	95,50	95,66	95,78
DoS	96,10	3,90	92,46	97,80	95,12	94,95
probe	98,37	1,63	96,45	85,93	99,65	90,57
R2L	99,10	0,90	0,00	0,00	99,99	0,00
U2R	99,95	0,05	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Assim como na arquitetura com 41 neurônios, a arquitetura 2C82N apresentou uma deficiência muito grande na classificação dos ataques R2L e U2R, obtendo *recall* igual a 0, ou seja, nenhum ataque destas classes foi classificado corretamente como ataque.

A Tabela 4 apresenta os resultados dos experimentos com a arquitetura 3C41N.

Tabela 4 - Resultados dos experimentos com a arquitetura 3C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	96,04	3,96	94,43	98,41	93,30	96,38
DoS	97,99	2,01	97,52	96,92	98,60	97,22
probe	97,49	2,51	93,81	78,24	99,46	85,32
R2L	99,17	0,83	26,81	17,79	99,81	0,00
U2R	99,96	0,04	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

A arquitetura 3C41N apresentou as mesmas deficiências das arquiteturas de duas camadas, no que diz respeito a classificação dos ataques das classes R2L e U2R, bem como dos ataques do tipo *probe*, que inclusive obteve um *recall* ainda pior, de 78,24%. Um destaque positivo foi a detecção da classe normal, que apresentou um *recall* de 98,41%, superior às arquiteturas 2C41N e 2C82N, que obtiveram 96,81% e 95,50%, respectivamente, representando um ganho considerável para esta classe com o aumento do número de camadas ocultas.

A Tabela 5 apresenta os resultados dos experimentos com a arquitetura 3C82N.

Tabela 5 - Resultados dos experimentos com a arquitetura 3C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	97,21	2,79	96,75	98,14	96,11	97,43
DoS	98,58	1,42	98,34	97,69	99,08	98,01
probe	98,74	1,26	96,02	90,10	99,62	92,96
R2L	98,81	1,19	28,35	32,41	99,36	30,12
U2R	99,96	0,04	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Com relação a arquitetura com 41 neurônios, a 3C82N se saiu melhor na classificação dos ataques do tipo *probe*, obtendo um *recall* de 90,10%, se aproximando das classes que possuem um maior número de instâncias, sendo estas: *normal* e *DoS*. A classificação para *R2L* também obteve um aumento de 14,62% no *recall*, mostrando uma vantagem da arquitetura com 82 neurônios por camada oculta.

A Tabela 6 apresenta os resultados dos experimentos com a arquitetura 4C41N.

Tabela 6 - Resultados dos experimentos com a arquitetura 4C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	95,09	4,91	93,37	97,89	91,83	95,57
DoS	98,65	1,35	99,53	96,71	99,75	98,10
probe	96,44	3,56	82,43	75,79	98,51	78,88
R2L	99,29	0,71	0,00	20,73	100,00	0,00
U2R	99,98	0,02	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Das arquiteturas avaliadas até o momento, a 4C41N foi a que teve o pior desempenho para a classe *probe*, obtendo apenas 75,79% de *recall*, a pior taxa registrada até o momento. Com relação às arquiteturas com 3 camadas, considerando apenas os valores de *recall* obtidos, a única classe em que esta arquitetura apresentou melhor resultado foi a *R2L*, que apresentou *recall* de 20,73%, contra os 17,79% da arquitetura 3C41N.

A Tabela 7 apresenta os resultados dos experimentos com a arquitetura 4C82N.

Tabela 7 - Resultados dos experimentos com a arquitetura 4C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	94,23	5,77	97,29	91,88	96,95	94,25
DoS	96,31	3,69	94,04	96,43	96,26	95,14
probe	96,35	3,65	79,96	92,56	96,72	84,10
R2L	99,42	0,58	44,85	29,05	99,90	0,00
U2R	99,93	0,07	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Diferentemente da 4C41N, a arquitetura com 82 neurônios obteve a melhor taxa de *recall* registrada até o momento para a classe *probe*, com 92,56%. A taxa de *recall* dos ataques do tipo R2L também melhorou significativamente, obtendo 29,05%, ficando atrás apenas da arquitetura 3C82N que registrou 32,41% para esta mesma classe. Em contrapartida, a 4C82N obteve as piores taxas de *recall* para as classes *normal* e DoS. Ainda contou com a pior acurácia para a classe *normal* já registrada, com apenas 94,23%, ao contrário das arquiteturas anteriores que não apresentaram resultados inferiores a 95%.

A Tabela 8 apresenta os resultados dos experimentos com a arquitetura 5C41N.

Tabela 8 - Resultados dos experimentos com a arquitetura 5C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	97,90	2,10	97,34	98,81	96,83	98,06
DoS	98,99	1,01	98,46	98,80	99,08	98,62
probe	99,16	0,84	99,39	91,48	99,94	95,27
R2L	99,67	0,33	85,26	62,64	99,93	71,54
U2R	99,95	0,05	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

A arquitetura 5C41N se destacou pelo *recall* de 62,64% obtido na classificação dos ataques R2L, classe que todas as outras arquiteturas apresentaram dificuldade para identificar. Apesar de não ser um valor alto como nas demais classes, representa um valor satisfatório levando em consideração o baixo número de amostras da classe R2L. Já nos ataques da classe U2R, a arquitetura,

assim como as demais, obteve *recall* de 0% devido a pequena quantidade de amostras desta classe presente na base de dados utilizada.

A Tabela 9 apresenta os resultados dos experimentos com a arquitetura 5C82N.

Tabela 9 - Resultados dos experimentos com a arquitetura 5C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	97,90	2,10	96,47	99,70	95,90	98,05
DoS	99,10	0,90	99,90	97,66	99,94	98,77
probe	99,26	0,74	99,14	93,09	99,92	95,94
R2L	99,55	0,45	92,65	40,20	99,94	42,34
U2R	99,96	0,04	0,00	0,00	100,00	0,00

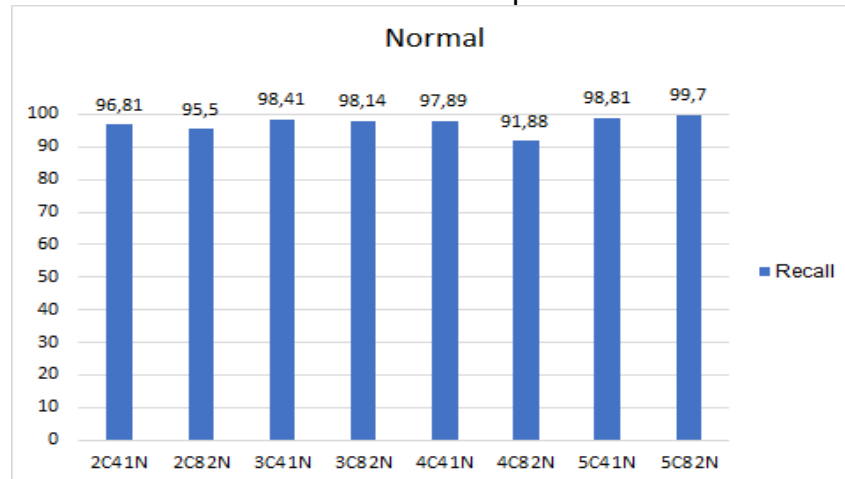
Fonte: PRÓPRIO (2020).

Com relação aos resultados apresentados na arquitetura 5C82N, podemos observar que ela obteve menores valores de *recall* para as classes DoS e R2L em comparação a arquitetura 5C41N. Em contrapartida, houve uma melhora desta métrica nas classes *normal* e *probe*. Apesar de haver algumas melhoras compensando as perdas, é possível afirmar que a arquitetura 5C41N foi melhor no experimento, devido ao *recall* obtido na identificação dos ataques R2L, de 62,64%, ser muito superior aos 40,20% obtidos na arquitetura 5C82N, representando uma variação muito grande para este tipo de ataque. Se levarmos em conta que as outras arquiteturas não obtiveram bom desempenho na identificação da classe R2L, podemos considerar a classe 5C41N a mais equilibrada, pois os valores obtidos nas métricas foram equivalentes em relação as demais arquiteturas, se sobressaindo no *recall* da classe R2L, considerada a mais deficiente do experimento se desconsiderarmos a classe U2R que unanimemente obteve 0% de *recall* em todas as arquiteturas testadas.

Para melhor visualização dos resultados, foram elaborados alguns gráficos comparando o desempenho das arquiteturas utilizadas através da métrica *recall*, demonstrando o quanto cada arquitetura conseguiu detectar em cada uma das 5 classes. Também foi realizada uma comparação específica entre arquiteturas com diferentes números de camadas e com diferentes números de neurônios, a fim de verificar se um dos objetivos específicos deste trabalho foi alcançado.

O Gráfico 1 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe normal.

Gráfico 1 - Resultados obtidos para classe Normal.

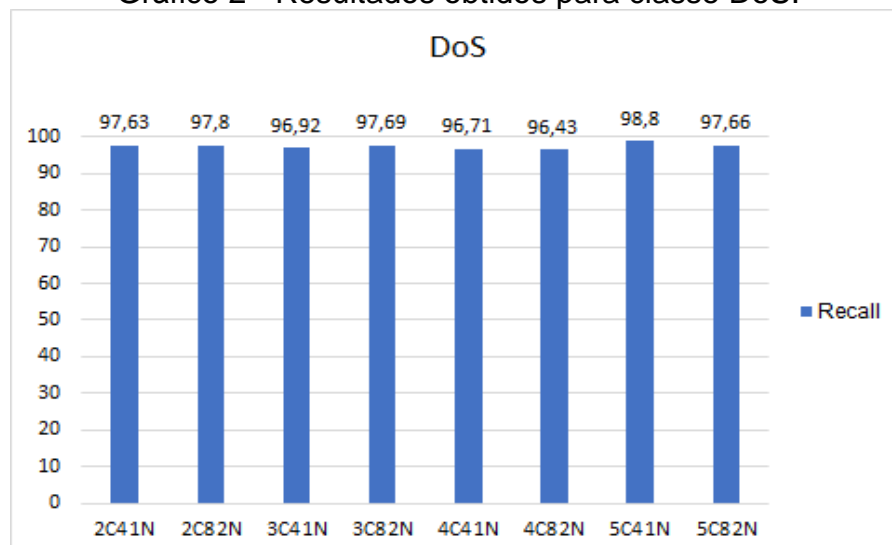


Fonte: PRÓPRIO (2020).

Conforme observado acima, todas as arquiteturas apresentaram valores satisfatórios de *recall* para a classe normal, ou seja, esta classe foi bem detectada nos experimentos. Destaque para as arquiteturas 5C41N e 5C82N, que apresentaram os melhores resultados, com 98,81% e 99,70% respectivamente.

O Gráfico 2 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe DoS.

Gráfico 2 - Resultados obtidos para classe DoS.

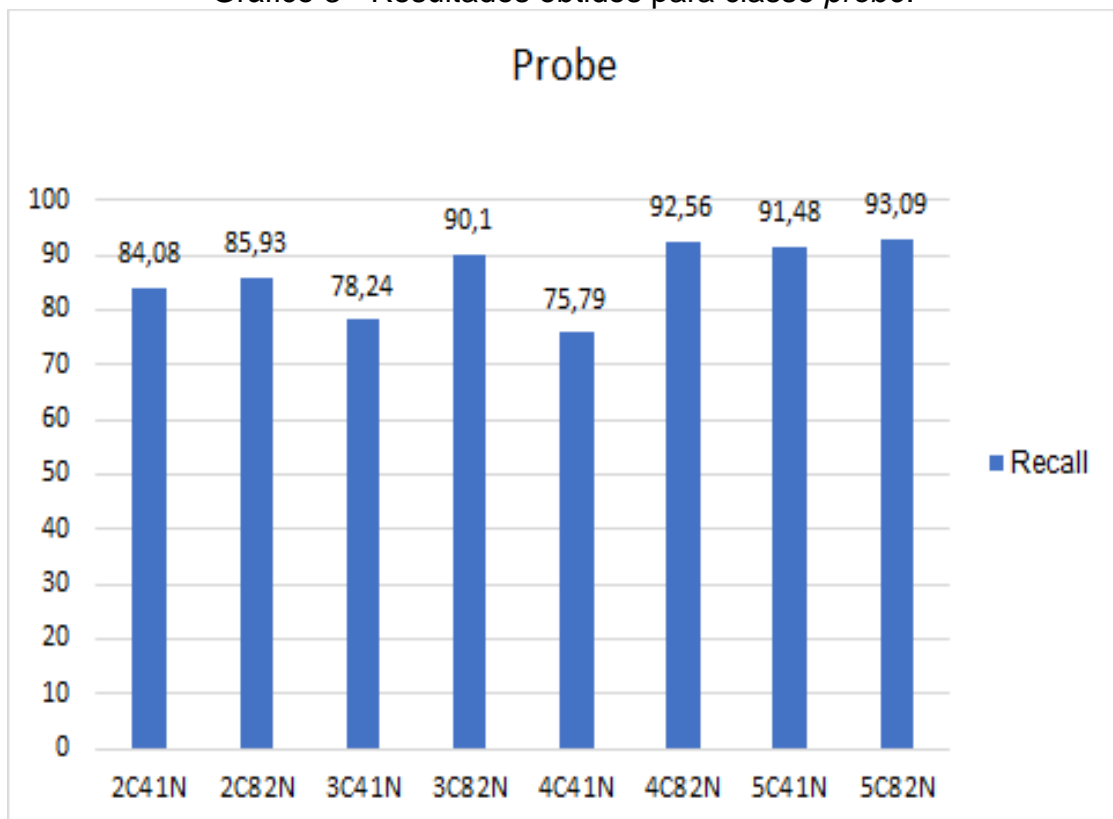


Fonte: PRÓPRIO (2020).

No Gráfico 2 é possível observar que a detecção dos ataques do tipo DoS, assim como na detecção da classe normal, os resultados foram satisfatórios. Esta classe foi a que obteve resultados mais consistentes, variando pouco entre as diferentes arquiteturas. A arquitetura que obteve a melhor taxa de detecção foi a 5C41N, com 98,80% de *recall*.

O Gráfico 3 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe *probe*.

Gráfico 3 - Resultados obtidos para classe *probe*.



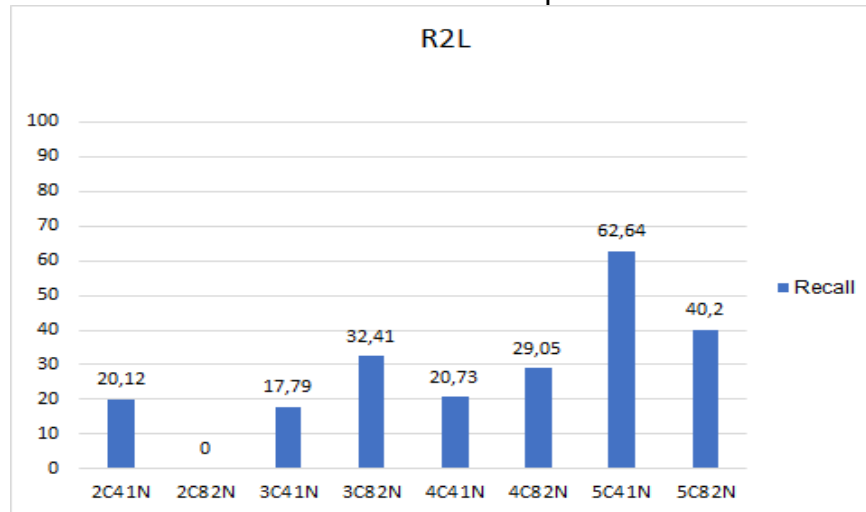
Fonte: PRÓPRIO (2020).

Diferente das classes analisadas anteriormente, a detecção dos ataques do tipo *probe* apresentaram uma maior variação entre as arquiteturas. É possível observar que nas arquiteturas com o mesmo número de camadas ocultas, o aumento do número de neurônios por camada melhorou a detecção desta classe. A arquitetura com o melhor resultado foi a 5C82N, com 93,09%.



O Gráfico 4 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe R2L.

Gráfico 4 - Resultados obtidos para classe R2L.

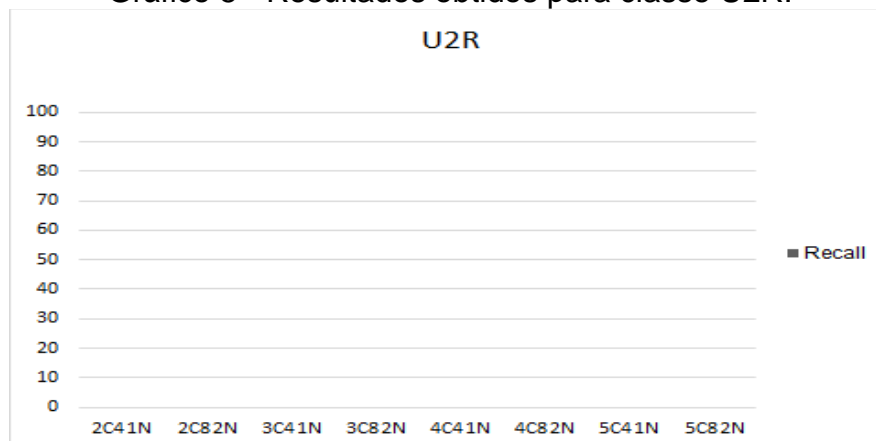


Fonte: PRÓPRIO (2020).

O Gráfico 4 demonstra que os valores de *recall* para os ataques do tipo R2L foram muito inferiores às classes previamente analisadas. A melhor arquitetura para esta classe foi a 5C41N, com 62,64% de detecção, que pode ser visto como um resultado satisfatório levando em consideração os resultados obtidos pelas demais arquiteturas.

O Gráfico 5 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe U2R.

Gráfico 5 - Resultados obtidos para classe U2R.

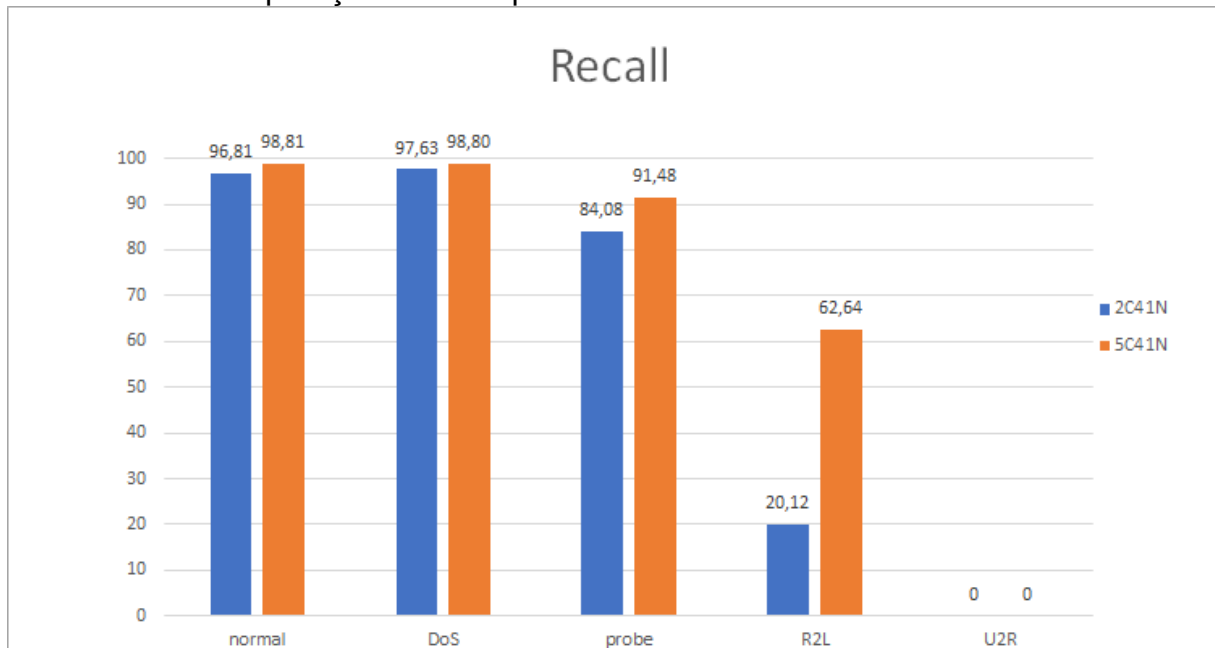


Fonte: PRÓPRIO (2020).

O Gráfico 5 demonstra que nenhuma arquitetura conseguiu detectar ataques do tipo U2R.

O Gráfico 6 apresenta uma comparação entre arquiteturas com diferentes números de camadas ocultas, com o objetivo de identificar se o aumento deste valor melhorou a detecção das classes.

Gráfico 6 - Comparação entre arquiteturas com diferentes números de camadas.

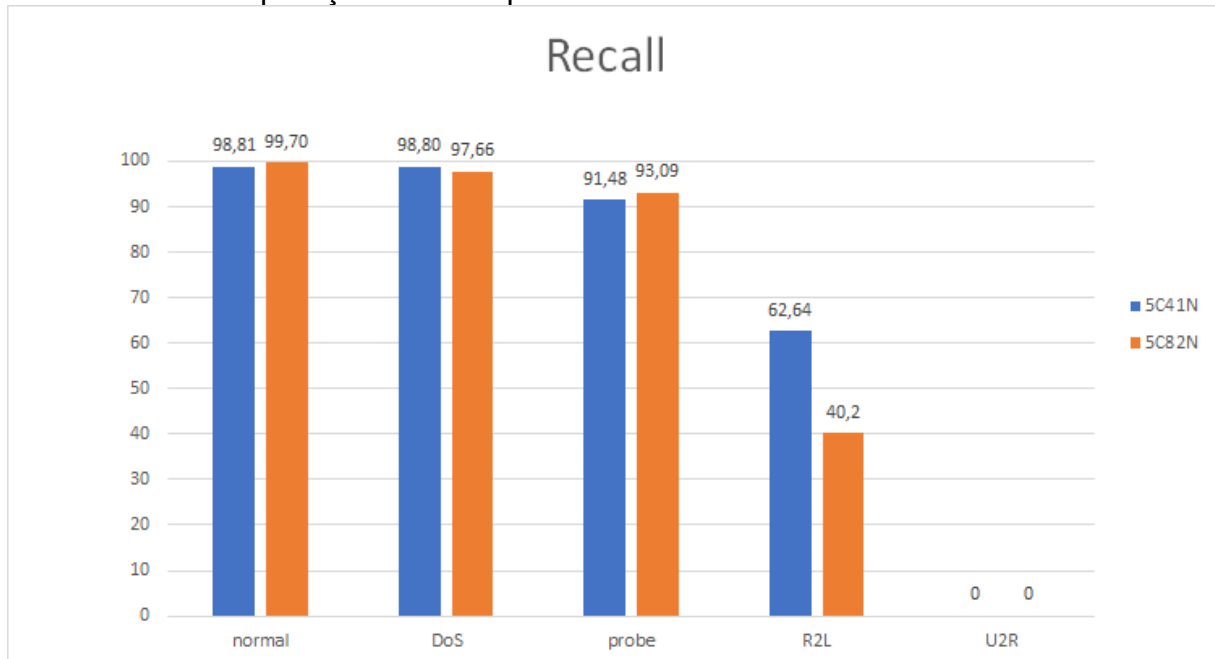


Fonte: PRÓPRIO (2020).

Conforme observado, a arquitetura 5C41N apresentou valores superiores a 2C41N, que utiliza um número menor de camadas ocultas. A principal diferença foi na detecção da classe R2L, onde a arquitetura com 5 camadas obteve 62,64% de *recall* contra os 20,12% da arquitetura mais simples. Portanto, é possível afirmar que o aumento do número de camadas ocultas representou uma melhora na detecção de cada uma das classes utilizadas.

O Gráfico 7 apresenta uma comparação entre arquiteturas com diferentes números de neurônios por camada oculta, com o objetivo de identificar se o aumento deste valor melhorou a detecção das classes.

Gráfico 7 - Comparação entre arquiteturas com diferentes números de neurônios.



Fonte: PRÓPRIO (2020).

Conforme os resultados apresentados acima, não é possível afirmar que o aumento do número de neurônios por camada oculta apresentou resultados melhores ou piores, pois houve uma variação de acordo com as classes. Porém, é possível destacar que a arquitetura 5C41N, com menor número de neurônios, apresentou uma detecção muito melhor para classe R2L do que a arquitetura 5C82N, podendo ser considerada melhor do que a mesma, se for levado em consideração que as variações obtidas nas demais classes são equivalentes.

### 5.3 DISCUSSÕES

Considerando as classes *normal*, *DoS* e *probe*, é possível afirmar que todas as arquiteturas obtiveram resultados satisfatórios de detecção, com altas taxas de acurácia, precisão e *recall*, especialmente nas duas primeiras classes. De modo geral, o aumento do número de camadas ocultas melhorou os valores do *recall*. Trabalhos futuros podem realizar experimentos com maiores quantidades de camadas, sempre levando em consideração o custo, pois quanto maior a quantidade

de neurônios da RNA, maior o custo de processamento, podendo tornar inviável a utilização da abordagem na prática.

Uma deficiência que pôde ser observada nos resultados comentados anteriormente da abordagem proposta, diz respeito a detecção e classificação dos ataques do tipo U2R e R2L. Os resultados obtidos nestas classes por todas as arquiteturas testadas não se equivalem aos encontrados nas demais, que majoritariamente apresentaram acurácia e precisão acima de 90%, com exceção da classe *probe*, que nas arquiteturas com 4 camadas obteve resultados na casa dos 80%. A classe U2R em especial foi a que apresentou os piores resultados, não sendo detectada por nenhuma das arquiteturas, apresentando sempre um *recall* de 0%.

O principal causador desta deficiência é a base de dados utilizada, que como mencionado na Seção 5.1.1, possui apenas 995 amostras de ataques da classe R2L e 52 da U2R. Porém, mesmo com resultados ruins, podemos considerar que a arquitetura com 5 camadas obteve um resultado satisfatório na detecção dos ataques R2L perante as demais arquiteturas, atingindo 62,64% na taxa de *recall*, no caso da arquitetura com 41 neurônios por camada oculta (5C41N), e 40,20% para a arquitetura 5C82N, que mesmo registrando uma queda significativa com o aumento do número de neurônios, manteve um valor maior do que as demais arquiteturas.

Para evitar esta discrepância nos resultados obtidos entre as diferentes classes, trabalhos futuros podem realizar experimentos com o balanceamento de classes, onde são utilizadas técnicas para que as classes de ataque da base de dados possuam quantidades similares.

## 6 CONCLUSÃO

É evidente que a tecnologia IoT é um paradigma revolucionário, que tende a se tornar cada vez mais presente no nosso dia a dia. Da mesma forma, é possível inferir que a segurança nos meios computacionais é uma área que tende a crescer muito, tanto em pesquisa quanto em desenvolvimento de novos métodos, pois ela é fundamental para garantir que não só a IoT, mas que outras tecnologias sejam mais seguras para os seus usuários.

Neste trabalho foram apresentadas algumas das dificuldades que os dispositivos IoT possuem quando tratamos de segurança, especialmente pela sua baixa capacidade de processamento, o que impossibilita o uso de métodos tradicionais de detecção de intrusão. Tendo isto em vista, foi proposta uma abordagem de detecção e identificação de intrusão situada no nível da *fog*, se aproveitando da capacidade de processamento deste paradigma para capturar e analisar os pacotes enviados pelos dispositivos da rede, a fim de detectar e identificar eventos intrusivos.

A abordagem proposta utilizou diferentes arquiteturas de Redes Neurais Artificiais que foram treinadas e testadas utilizando o conjunto de treino da base de dados NSL-KDD. Este *dataset* foi dividido em 10 subconjuntos com o auxílio do método de *cross-validation*, com o intuito de estimar o quão precisa a abordagem é na prática.

Com a realização e avaliação dos experimentos, pode-se concluir que a arquitetura mais eficiente deste trabalho foram as que utilizaram 5 camadas ocultas (5C41N e 5C82N). Trabalhos futuros podem ser realizados em cima de arquiteturas que utilizam 5 camadas ocultas, ou aumentando ainda mais a quantidade de camadas, explorando com maior variedade o número de neurônios em cada uma delas, a procura de uma arquitetura ainda mais eficiente do que as apresentadas.

## REFERÊNCIAS

- ALHARBI, Ali; ALHAIDARI, Sulaiman; ZOHDY, Mohamed. Denial-of-Service, Probing, User to Root (U2R) & Remote to User (R2L) Attack Detection using Hidden Markov Models. **International Journal Of Computer And Information Technology**. Rochester, p. 204-210. set. 2018.
- AL-HAWAWREH, Muna; MOUSTAFA, Nour; SITNIKOVA, Elena. **Identification of malicious activities in industrial internet of things based on deep learning models**. **Journal Of Information Security And Applications**, [s.l.], v. 41, p.1-11, ago. 2018.
- ALJAWARNEH, Shadi; ALDWAIRI, Monther; YASSEIN, Muneer Bani. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. **Journal Of Computational Science**, [S.L.], v. 25, p. 152-160, mar. 2018.
- ALMIANI, Muder; ABUGHAZLEH, Alia; AL-RAHAYFEH, Amer; RAZAQUE, Abdul. Cascaded hybrid intrusion detection model based on SOM and RBF neural networks. **Concurrency And Computation: Practice and Experience**, [S.L.], v. 32, n. 21, p. 1-14, 7 mar. 2019.
- AL-SARAWI, Shadi et al. Internet of Things (IoT) communication protocols: Review. **2017 8th International Conference On Information Technology (icit)**, [s.l.], p.685-690, maio 2017.
- BACE, Rebecca; MELL, Peter. **NIST Special Publication on Intrusion Detection Systems**. Scotts Valley: National Institute Of Standards And Technology, 2001.
- BONOMI, Flavio et al. Fog computing and its role in the internet of things. **Proceedings Of The First Edition Of The Mcc Workshop On Mobile Cloud Computing - Mcc '12**, [s.l.], p.13-16, 2012.
- BUTUN, Ismail; MORGERA, Salvatore D.; SANKAR, Ravi. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. **IEEE Communications Surveys & Tutorials**, [s.l.], v. 16, n. 1, p.266-282, 2014.
- CAMPELLO, Rafael; WEBER, Raul. **Sistemas de Detecção de Intrusão**. Florianópolis: Xix Simpósio Brasileiro de Redes de Computadores, 2001. 173 slides, color.
- GARDNER, M.W; DORLING, S.R. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric Environment**, [S.L.], v. 32, n. 14-15, p. 2627-2636, ago. 1998. Elsevier BV.
- HAYKIN, Simon. **Redes Neurais: Princípios e Prática**. [S. L]: Bookman, 2001.

ILGUN, K.. USTAT: a real-time intrusion detection system for UNIX. **Proceedings 1993 IEEE Computer Society Symposium On Research In Security And Privacy**, [s.l.], p.16-28, 1993.

IORGA, Michaela et al. **Fog Computing Conceptual Model**. [s.l.]: National Institute Of Standards & Technology, 2018.

JAFIER, Shatha H.. Utilizing feature selection techniques in intrusion detection system for internet of things. **Proceedings Of The 2nd International Conference On Future Networks And Distributed Systems - Icfnds '18**, [s.l.], p.1-3, 2018.

JEYA, P. Gifty; MURALIDHARAN, Ravichandran; RAVICHANDRAN, C. S.. Efficient Classifier for R2L and U2R Attacks. **International Journal Of Computers And Applications**. [S. L.], p. 28-32. jan. 2012.

JHAVERI, Rutvij H.; PATEL, Sankita J.; JINWALA, Devesh C.. DoS Attacks in Mobile Ad Hoc Networks: a survey. **2012 Second International Conference On Advanced Computing & Communication Technologies**, [S.L.], p. 535-541, jan. 2012. IEEE.

KHAMPHAKDEE, Nattawat; BENJAMAS, Nunnapus; SAIYOD, Saiyan. Improving Intrusion Detection System based on Snort rules for network probe attack detection. **2014 2Nd International Conference On Information And Communication Technology (Icoict)**, [S.L.], p. 69-74, maio 2014.

KOLIAS, Constantinos *et al.* DDoS in the IoT: mirai and other botnets. **Computer**, [S.L.], v. 50, n. 7, p. 80-84, 2017. Institute of Electrical and Electronics Engineers (IEEE).

MACHADO, Renato Bobsin. **UMA ABORDAGEM DE DETECÇÃO DE INTRUSÃO BASEADA EM SISTEMAS IMUNOLÓGICOS ARTIFICIAIS E AGENTES MÓVEIS**. 2005. 164 f. Dissertação (Mestrado) - Curso de Pós-graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2005.

MELL, Peter.; GRANCE, Timothy. **The NIST Definition of Cloud Computing**. Gaithersburg: National Institute Of Standards & Technology, 2011.

MUKHERJEE, B.; HEBERLEIN, L.t.; LEVITT, K.n.. Network intrusion detection. **IEEE Network**, [s.l.], v. 8, n. 3, p.26-41, maio 1994.

NAQA, Issam El; MURPHY, Martin J.. What Is Machine Learning? Machine Learning In Radiation Oncology, [S.L.], p. 3-11, 2015. Springer International Publishing.

NORTHCUTT, Stephen et al. **Intrusion Signatures and Analysis**. Thousand Oaks: New Riders Publishing, 2001.

SHEIKHAN, Mansour; BOSTANI, Hamid. **A hybrid intrusion detection architecture for Internet of things. 2016 8th International Symposium On Telecommunications (ist)**, [s.l.], p.601-606, set. 2016.

SOUZA, Cristiano Antonio de. **MÉTODO HÍBRIDO DE DETECÇÃO DE INTRUSÃO APLICANDO INTELIGÊNCIA ARTIFICIAL**. 2018. 142 f. Dissertação (Mestrado) - Curso de Programa de Pós-graduação em Engenharia Elétrica e Computação, Universidade Estadual do Oeste do Paraná, Foz do Iguaçu, 2018.

SPAFFORD, Eugene H; ZAMBONI, Diego. Intrusion detection using autonomous agents. **Computer Networks**, [s.l.], v. 34, n. 4, p.547-570, out. 2000.

TAVALLAEE, Mahbod et al. A detailed analysis of the KDD CUP 99 data set. **2009 IEEE Symposium On Computational Intelligence For Security And Defense Applications**, [S.L.], p. 1-6, jul. 2009. IEEE.

U.FAROOQ, M. et al. A Review on Internet of Things (IoT). **International Journal Of Computer Applications**, [S.L.], v. 113, n. 1, p. 1-7, 18 mar. 2015. Foundation of Computer Science.

WORTMANN, Felix; FLÜCHTER, Kristina. Internet of Things: Technology and Value Added. **Internet Of Things. Business & Information Systems Engineering**, [s. L], p.221-224, jun. 2015.

XIE, Junfeng et al. A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): research issues and challenges. **IEEE Communications Surveys & Tutorials**, [S.L.], v. 21, n. 1, p. 393-430, 2019. Institute of Electrical and Electronics Engineers (IEEE).

YANNUZZI, M. et al. Key ingredients in an IoT recipe: fog computing, cloud computing, and more fog computing. **2014 IEEE 19th International Workshop On Computer Aided Modeling And Design Of Communication Links And Networks (Camad)**, [S.L.], p. 325-329, dez. 2014.



## APÊNDICE A – ARTIGO DA MONOGRAFIA

### ABORDAGEM PARA DETECÇÃO E IDENTIFICAÇÃO DE INTRUSÃO EM AMBIENTES DE FOG COMPUTING E IOT

**Pedro Silveira Dalenogare**

Centro Tecnológico - Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

#### **ABSTRACT**

*Security is one of the key points for the success of existing technologies in general. In the Internet of Things (IoT), where there is a large amount of devices generating and sharing a large volume of data, it is no different. In this recent technology, security is, without a doubt, one of the fields that deserves more attention, since it is fundamental to guarantee the integrity and privacy of users' data. One of the existing methods for maintaining security in IoT are the Intrusion Detection Systems, which aim to detect attack attempts and intrusion in intelligent devices. A recurring practice in the development of these systems is the use of Artificial Neural Networks, a structure that allows the creation of machine learning algorithms for intrusion detection. Most of the existing approaches focus on binary detection, that is, without specifying the type of attack. Another existing model is the multiclass classification, which is able to detect and identify what type of intrusion. The multiple classification approaches serve to assist in the execution of countermeasures, once it is known which type of attack is being executed, it becomes easier to contain. This course conclusion paper aims to propose an approach to detect and identify intrusion in fog computing and IoT using Artificial Neural Networks trained to identify possible attacks to devices.*

**Keywords:** *Internet of Things. Intrusion Detection. Artificial Neural Networks.*

#### **RESUMO**

A segurança é um dos pontos chaves para o sucesso das tecnologias existentes em geral. Na Internet das Coisas (Internet of Things - IoT), onde há uma grande quantidade de dispositivos gerando e compartilhando um grande volume de dados, não é diferente. Nesta recente tecnologia, a segurança é, sem dúvidas, uma das áreas que merece mais atenção, pois é fundamental para a garantia da integridade e privacidade dos dados dos usuários. Um dos métodos existentes para a manutenção da segurança em IoT são os Sistemas de Detecção de Intrusão, que possuem como objetivo detectar tentativas de intrusão nos dispositivos inteligentes. Uma prática recorrente no desenvolvimento destes sistemas é a utilização de Redes Neurais Artificiais, uma estrutura que permite a criação de algoritmos de aprendizado de máquina para detecção de intrusão. A maior parte das abordagens existentes focam na detecção binária, ou seja, sem especificar o tipo do ataque. Outro modelo existente é o de classificação multiclasse, capaz de detectar e identificar o tipo de intrusão. As abordagens de classificação múltipla servem para auxiliar na execução de contramedidas, pois identificando tipo de ataque que está sendo executado, torna-se mais fácil contê-lo. Este trabalho de conclusão de curso tem como objetivo propor uma abordagem para detecção e identificação

de intrusão em fog computing e IoT utilizando Redes Neurais Artificiais treinadas para identificar possíveis ataques aos dispositivos.

**Palavras-chave:** Internet das Coisas. Detecção de Intrusão. Redes Neurais Artificiais.

## 1 INTRODUÇÃO

Com a popularização da Internet das Coisas (*Internet of Things* - IoT), tecnologia em alta nos tempos atuais e utilizada em diversas áreas como agricultura, saúde, casas inteligentes, entre outras aplicações (U.FAROOQ *et al.*, 2015), algumas preocupações acerca deste tema vêm se destacando.

Aliada à inovação, esta recente tecnologia é seguida de muitos desafios, sendo o principal deles, sem dúvidas, a segurança. Para o sucesso do conceito da IoT, é imprescindível que haja a garantia da segurança e privacidade dos dados dos usuários. A detecção de intrusão é uma das técnicas utilizadas para aumentar segurança dos dispositivos, tendo como objetivo detectar e identificar tentativas de intrusão em ambientes computacionais por parte de invasores internos e externos (MUKHERJEE; HEBERLEIN; LEVITT, 1994). Os Sistemas de Detecção de Intrusão (*Intrusion Detection Systems* - IDS) se diferenciam em diversos aspectos, como o tipo de detecção realizada, a arquitetura do sistema de acordo com o alvo das detecções, o local onde está disposto, entre outras características.

Com o aumento da quantidade de dados geradas pelos dispositivos IoT, se fez necessária a utilização de outras tecnologias para auxiliar no processamento e armazenamento dos dados. Inicialmente foi utilizada a *cloud computing*, tanto no processamento quanto no armazenamento de dados. Esta tecnologia, porém, apresentou alta latência e baixa eficiência no processamento dos dados, por atuar distante do dispositivo IoT (YANNUZZI *et al.*, 2014). Visto que os dispositivos necessitam de uma resposta imediata, a *fog computing* foi adotada para atuar no processamento dos dados, já que este paradigma atua na borda da rede, mais próxima dos dispositivos (BONOMI *et al.*, 2012). Como o processamento dos dados dos dispositivos IoT são feitos no nível da *fog*, é possível propor uma abordagem de detecção e identificação de intrusão atuando neste nível, através da análise dos tráfegos enviados pelos dispositivos.

Outra tecnologia bastante utilizada nos dispositivos IoT, especialmente em métodos de detecção de intrusão, são as Redes Neurais Artificiais (RNA), que podem ser utilizadas no processo de aprendizado de máquina, através da utilização de um conjunto de treinamento (HAYKIN, 2001), ou seja, permite o desenvolvimento de abordagens eficientes com os mais diversos objetivos.

Neste trabalho é proposta uma abordagem que utiliza RNA com o objetivo de detectar e identificar pacotes maliciosos. Esta abordagem atua no nível da *fog*, que como dito anteriormente, é responsável pelo processamento dos dados da tecnologia IoT, ou seja, permite a captura e análise dos tráfegos entre os dispositivos.

Atualmente, existem diversas abordagens de detecção de intrusão que utilizam RNA. Porém, grande parte destes trabalhos focam exclusivamente na detecção binária, ou seja, apenas detectam se um evento é intrusivo ou não, sem especificar o tipo de ataque. Um outro modelo existente é o de classificação multiclasse, que é capaz de detectar a intrusão e identificar a qual tipo de ataque ela pertence. As abordagens de classificação múltipla podem auxiliar na execução de contramedidas, uma vez que é conhecido o tipo de ataque que está

sendo executado, torna-se mais fácil contê-lo. Este tipo de classificação também facilita para o gerenciador da rede identificar os tipos de ataques realizados e garantir uma melhor segurança para o sistema em questão. Por exemplo, se a abordagem multiclasse detecta um ataque de acesso remoto (U2R), o gerente da rede pode tratar este evento intrusivo e já exercer uma contramedida para um possível ataque de escalonamento de privilégio (R2L), tendo em vista que estes dois ataques geralmente acontecem nesta ordem. O número inferior de trabalhos com este tipo de abordagem em relação aos de detecção binária é uma das motivações para a realização deste trabalho.

Com o objetivo de encontrar a abordagem mais eficiente utilizando RNA, serão apresentadas diferentes arquiteturas, variando os números de camadas ocultas e de neurônios. Após a realização do treinamento das arquiteturas, é realizada uma avaliação individual por arquitetura, levando em conta as métricas selecionadas. Posteriormente, uma avaliação geral dos resultados é realizada observando o desempenho das arquiteturas considerando as duas principais métricas desta abordagem: acurácia e precisão.

O objetivo deste trabalho de conclusão de curso é propor uma abordagem baseada em Redes Neurais Artificiais para detecção e identificação de intrusão, visando a classificação de ataques à *fog computing* e IoT.

As principais contribuições deste trabalho são:

1. Uma revisão do estado da arte relacionado a detecção de intrusão em ambiente de *fog computing* e IoT utilizando Redes Neurais Artificiais e *machine learning*, identificando os principais problemas e soluções existentes;
2. A proposta de uma abordagem baseada em Redes Neurais Artificiais para detecção e identificação de intrusão em ambientes IoT e *fog computing*, utilizando Redes Neurais Artificiais com método de classificação.
3. Realização de detecção multiclasse, especificando o tipo de intrusão detectada;
4. Investigação da quantidade ideal de camadas ocultas e neurônios, analisando a influência da alteração destes valores nos resultados.

## 2 ESTADO DA ARTE

Esta Seção possui como objetivo contextualizar o estado atual das pesquisas e estudos voltados à detecção de intrusão em dispositivos IoT. Para seleção do material, foi realizada uma breve revisão bibliográfica, selecionando artigos escritos em língua inglesa e portuguesa, publicados nas fontes: *IEEE*, *ACM*, *Elsevier* e *Springer*. Além disso, foram considerados também trabalhos de dissertação na área em questão.

No trabalho da autora Jafier (2018) foi apresentado um sistema de detecção de intrusão para dispositivos IoT baseado em *Data Mining*, para ser utilizado em ambientes de rede sem fio. Segundo a autora, alguns algoritmos de *Data Mining* como *clustering* e classificação já são utilizados em sistemas de detecção de intrusão. Na abordagem proposta, ela adota a utilização do algoritmo ID3 como classificador em conjunto com outros algoritmos buscando a otimização do sistema de detecção de intrusão. O modelo proposto recebe como entrada conexões de qualquer ambiente de rede (com fio, sem fio e objetos inteligentes), e como saída classifica a conexão como normal ou intrusiva, para esta última,

sendo especificado o tipo de intrusão. Como resultado, o sistema se mostrou capaz de descobrir informações sobre os ataques e as utiliza para construir o conjunto de dados final para treinar e testar o classificador ID3, tornando-o mais eficiente na detecção de intrusão.

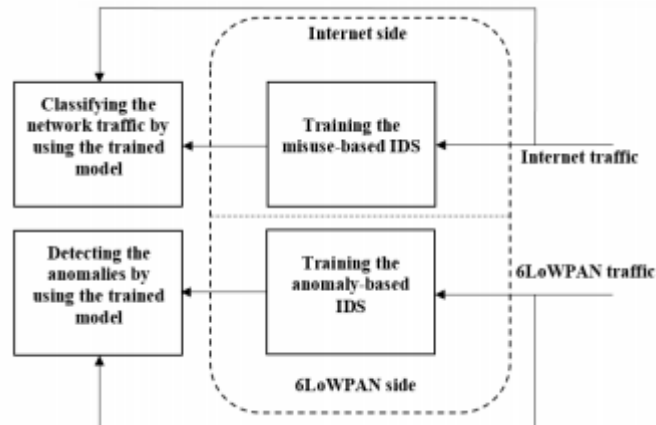
Na dissertação de mestrado de Souza (2018), o autor apresenta um método híbrido de detecção de intrusão utilizando o método de classificação *K-Nearest Neighbors* (KNN), muito conhecido pela acurácia na detecção de eventos intrusivos e não intrusivos, porém, com um tempo de execução elevado na etapa de classificação. Outra técnica utilizada neste trabalho foram as Redes Neurais Artificiais (RNA), que possuem um tempo de execução melhor, quando comparado ao KNN, porém, com acurácia de detecção inferior. O objetivo desta proposta é melhorar a taxa de detecção do método RNA, que possui menor necessidade de processamento do que o KNN na etapa de classificação. Em contrapartida, a função do KNN é determinar a classe de um elemento a partir de uma correlação com exemplos pré-estabelecidos. Com a melhoria da acurácia do método RNA é possível obter um método de detecção de intrusão mais eficiente, utilizando estas duas técnicas em conjunto, extraindo o melhor de cada uma. Para comprovação da eficiência desta abordagem, foram realizados diversos testes de eficiência com os métodos RNA e KNN em conjunto, e com cada uma destas técnicas sendo utilizadas individualmente, a fim de gerar uma base de resultados permitindo a comparação entre os diferentes modelos testados. Como conclusão, o autor cita que o método híbrido proposto obteve desempenho superior ao da técnica RNA em termos de acurácia em todas as bases de dados utilizadas nos testes. Outra conclusão que se obteve, foi de que as taxas de acurácia do método híbrido foram equivalentes ao do método KNN, porém, levou vantagem em relação ao tempo de processamento, apresentando melhor eficiência. Por fim, a abordagem comprova que o uso híbrido destas técnicas na detecção de intrusão é superior ao uso individual de cada uma delas.

No trabalho de Al-hawawreh, Moustafa e Sitnikova (2018), os autores propõe uma técnica de detecção por anomalia para sistemas de controle industrial. Devido a deficiência dos sistemas de detecção existentes na parte de coleta de informação e utilização desta para aperfeiçoamento próprio, a técnica apresentada pelos autores é baseada em modelos de *deep learning*, capazes de aprender e validar a partir de informações obtidas em pacotes TCP/IP. Para confecção desta técnica, são utilizados processos de treinamento de redes neurais com auxílio de dois *datasets* famosos, o NSL-KDD e o UNSW-NB15. Os métodos de *deep learning* utilizados neste trabalho foram o *deep auto-encoder*, utilizado na fase de treinamento do sistema de detecção e responsável por criar os parâmetros de inicialização utilizados no treinamento da *deep feedforward neural network*, outra técnica de *deep learning* utilizada no trabalho. Os autores utilizaram ainda um algoritmo de *backpropagation* no cálculo da performance do resultado obtido e na propagação deste valor para as camadas ocultas, onde os pesos inicialmente fornecidos pelo *deep auto-encoder* são atualizados, visando sempre a melhora da acurácia do modelo. Como resultado, os autores conseguiram uma acurácia de 98.6 e 92.4, taxa de detecção de 99 e 93, e taxa de falso positivo de 1.8 e 8.2, para os *datasets* NSL-KDD e UNSW-NB15, respectivamente, comprovando a eficiência da técnica desenvolvida quando comparada com outros métodos existentes, de acordo com as tabelas comparativas apresentadas pelos autores.

No trabalho de Sheikhan e Bostani (2016), os autores propõe uma abordagem de detecção de intrusão por anomalia, capaz de detectar ataques conhecidos ou não, e detecção por abuso, conhecida por reconhecer os ataques pela sua assinatura. O modelo proposto é baseado no modelo de programação *MapReduce*, que processa grandes quantidades de dados em paralelo, possibilitando a detecção distribuída. O método de classificação utilizada foi o

*Optimum-Path Forest*, e o *dataset* utilizado para treinamento e teste deste classificador foi o NSL-KDD. O modelo proposto consiste em dois módulos, o da *Internet*, utilizado na detecção por abuso, e do *6LoWPAN*, responsável pela detecção por anomalia, conforme demonstrado na Figura 4.

Figura 1 - Modelo proposto.



Fonte: (SHEIKHAN; BOSTANI, 2016).

Devido a estrutura do modelo proposto, foi possível realizar a detecção de intrusão paralela. Após realização de testes com o *dataset* escolhido, os autores obtiveram para o módulo de detecção por anomalia uma taxa de detecção de 80.95 e taxa de alarme falso de 5.92. No módulo de detecção por abuso, os valores encontrados foram de 96.20 e 1.44. Os resultados obtidos demonstraram uma performance superior na detecção simultânea de ataque internos e cibernéticos no ambiente da Internet das Coisas.

Grande parte dos trabalhos analisados focam em detecção binária, onde um tráfego é analisado e classificado entre normal ou ataque. Já os estudos que utilizaram a detecção multiclasse, apresentaram alguma deficiência na detecção de tipos específicos de ataques. O incentivo de pesquisas e estudos focados na detecção multiclasse é importante, pois neste modelo o IDS é capaz de detectar a intrusão e especificar o tipo de intrusão que está sendo realizada, permitindo ao gerenciador da rede a execução de contramedidas específicas para determinados ataques.

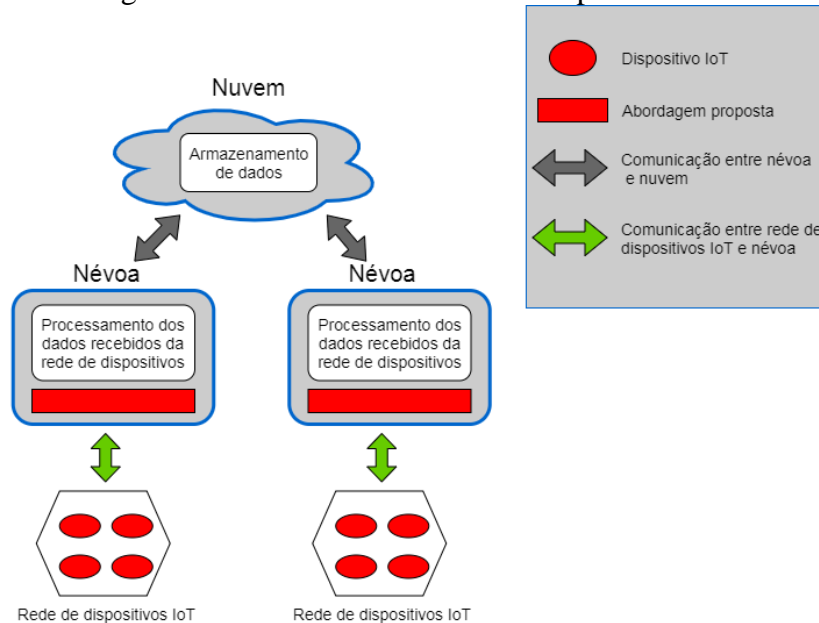
### 3 PROPOSTA DE DESENVOLVIMENTO DO TRABALHO

Neste trabalho é proposta uma abordagem de detecção e identificação de intrusão para dispositivos IoT utilizando redes neurais artificiais.

Como já mencionado neste trabalho, os dispositivos inteligentes possuem recursos limitados, com pouca capacidade de processamento (YANNUZZI et al., 2014). Por este motivo, não é possível utilizar um método de detecção de intrusão tradicional por anomalia, assim como a abordagem não pode atuar diretamente nos dispositivos IoT. Como a *fog* é utilizada no processamento dos dados destes dispositivos, atuando mais próximo da rede, obtendo assim menor latência em comparação a *cloud*, a abordagem proposta está situada no nível da *fog* e é responsável por monitorar a rede de dispositivos IoT.

A proposta consiste em uma abordagem de detecção e identificação de intrusão baseado em Redes Neurais Artificiais, treinada com uma base de dados contendo diversos tipos de ataques, possibilitando a detecção e identificação dos pacotes enviados pelos dispositivos IoT à *fog*. A Figura 1 ilustra um ambiente com dispositivos IoT com a abordagem proposta inserida no nível da *fog*.

Figura 2 - Ambiente com redes de dispositivos IoT.



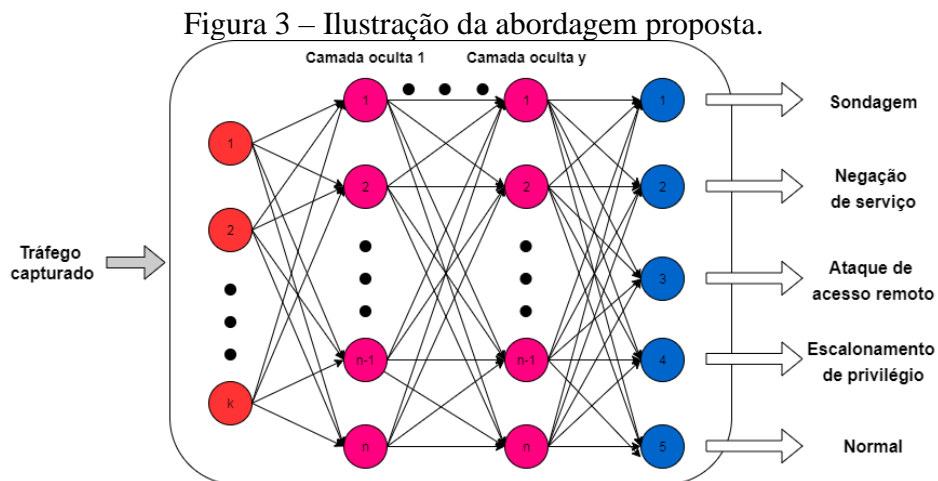
Fonte: PRÓPRIO (2020).

A abordagem proposta atua na névoa capturando os pacotes enviados pelos dispositivos IoT, analisando cada pacote e classificando-os entre 5 classes diferentes, sendo uma classe para eventos não intrusivos, chamada de *normal*, e quatro classes de ataques, sendo estas:

- Sondagem (*Probe*): ataque que visa obter informações a respeito do sistema atacado, como endereço IP, sistema operacional utilizado entre outras informações confidenciais. É um tipo de ataque que como o próprio nome diz, sonda o alvo, capturando informações a respeito do mesmo, informações estas que futuramente podem ser utilizadas em ataques mais intrusivos, como ataques negação de serviço (KHAMPHAKDEE; BENJAMAS; SAIYOD, 2014).
- Negação de serviço (*Denial of Service - DoS*): possui como objetivo tornar os serviços do sistema alvo indisponíveis, através do envio de pacotes maliciosos. Este ataque pode fazer com que os pacotes maliciosos consumam todos os recursos disponíveis do sistema, tornando-o inútil, ou então o ataque pode ainda obstruir a comunicação entre o sistema alvo e os utilizadores dele (JHAVERI; PATEL; JINWALA, 2012).
- Ataque de acesso remoto (*Remote to Local - R2L*): o atacante tenta obter acesso ao usuário local do sistema remoto através de alguma vulnerabilidade do mesmo. Este ataque precede ao ataque de escalonamento de privilégio (ALHARBI; ALHAIDARI; ZOHDY, 2018).

- Escalonamento de privilégio (*User to Root - U2R*): neste tipo de ataque, o usuário malicioso que já conquistou acesso ao sistema remoto busca obter acesso a recursos privilegiados, ou seja, recursos que aquele não deveria ter acesso, permitindo que ele possa executar *malwares*, tarefas de administrador, entre outras funções que podem comprometer o sistema (JEYA; MURALIDHARAN; RAVICHANDRAN, 2012).

A Figura 2 ilustra a abordagem proposta, responsável por analisar o tráfego obtido na entrada e fornecer uma classificação na camada de saída.



Fonte: PRÓPRIO (2020).

Conforme observado na Figura 2, a RNA possui uma camada de entrada, com  $k$  neurônios, valor este que deve ser equivalente ao número de atributos que cada amostra da base de dados possui. O número de camadas ocultas  $y$  é variável e será testado com diferentes valores, assim como o número de neurônios  $n$  em cada camada oculta. A alteração destes valores tem como objetivo investigar a influência do aumento do número de camadas ocultas e de neurônios na detecção das classes estabelecidas. Para tal, os processos de treinamento e teste da RNA serão realizados com diferentes arquiteturas. Os neurônios das camadas ocultas, responsáveis pelo aprendizado da rede, utilizam a função de ativação *ReLU*, que trabalha como uma função linear quando os valores de entrada repassados pelo somador são maiores que 0, dando como saída da função o mesmo valor recebido na entrada. Caso o valor do *input* seja negativo, ela se comporta como uma função não linear, pois transforma estas entradas em 0, e este valor é considerado no *output* do neurônio. Esta função de ativação se tornou a função padrão utilizada para diversos tipos de Redes Neurais, pois os modelos que a utilizam são mais fáceis de treinar e normalmente atingem melhores resultados. A camada de saída conta com 5 neurônios, um para cada classe: sondagem, negação de serviço, ataque de acesso remoto, escalonamento de privilégio e normal. Estas classes foram definidas de acordo com os tipos de ataques presentes na base de dados utilizada. Estes neurônios utilizam a função de ativação *softmax*, que gera um valor de classificação entre 0 e 1 para cada neurônio. Com esta função, as saídas para cada uma das classes são transformadas em valores entre 0 e 1, dando a probabilidade de uma determinada entrada pertencer a cada uma das classes.

## 4 EXPERIMENTOS

Nesta Seção são apresentados os experimentos realizados para avaliar a abordagem proposta. São apresentados detalhes sobre a base de dados utilizada, a técnica de *cross-validation* e as métricas de avaliação consideradas. Por fim, os resultados obtidos nos experimentos são apresentados e comentados.

### 4.1 Base de Dados

A base de dados utilizada no treinamento da abordagem proposta foi a NSL-KDD. Este conjunto de dados é uma versão melhorada do KDDCUP1999, outro conjunto de dados de referência para detecção de intrusão (TAVALLAEE et al., 2009). O conjunto de treino do NSL-KDD consiste em 125973 amostras, enquanto o conjunto de teste possui 22544 amostras. Este *data set* abrange a maioria das categorias de ataques, entre elas: *Probe*, DoS, U2R e R2L. Nesta abordagem, são trabalhados apenas os dados referentes ao conjunto de treino da NSL-KDD, com o auxílio do método *k-fold cross-validation* com 10 *folds*, ou seja, as 125973 amostras são divididas em 10 subconjuntos, onde cada um destes subconjuntos é utilizado uma vez para teste, enquanto os demais são utilizados na etapa de treinamento.

### 4.2 Métricas de Avaliação

Uma das principais etapas na validação da abordagem proposta é a utilização de métricas para compreender o desempenho obtido nos experimentos. As métricas utilizadas na avaliação deste trabalho podem ser observadas a seguir (ALMIANI et al., 2019):

1. **Acurácia (ACC):** métrica que apresenta a proporção de eventos classificados corretamente sobre o número total de classificações. A acurácia é calculada através da Equação 3.1, apresentada abaixo:

$$ACC = \frac{PV + NV}{FP + FN + PV + NV} \quad (4.1)$$

2. **Erro (ERR):** apresenta o percentual de eventos classificados incorretamente sobre o número total de classificações. O erro é calculado pela Equação 3.2.

$$ERR = \frac{FP + FN}{FP + FN + PV + NV} \quad (4.2)$$

3. **Precisão (PRE):** métrica bastante utilizada para avaliação de algoritmos de *machine learning*. Apresenta a taxa de eventos classificados corretamente como intrusivos dentre todos os eventos classificados como intrusivos. É obtida através da Equação 3.3:

$$PRE = \frac{PV}{FP + PV} \quad (4.3)$$



4. **Recall:** apresenta a taxa de eventos classificados corretamente como intrusivos dentre todos os eventos intrusivos do conjunto de dados. É calculada pela Equação 3.4:

$$Recall = \frac{PV}{PV + FN} \quad (4.4)$$

5. **True Negative Rate (TNR):** apresenta a taxa de eventos classificados não intrusivos entre todos os eventos não intrusivos. É calculada a partir da Equação 3.5:

$$TNR = \frac{NV}{NV + FP} \quad (4.5)$$

6. **F1-Score:** esta métrica apresenta uma média harmônica de precisão e *recall*, onde o melhor valor é 1, ou seja, boas taxas de precisão e *recall*, e o pior valor é igual a 0. É calculada pela Equação 3.6:

$$F1-SCORE = 2 * \frac{Precisão * Recall}{Precisão + Recall} \quad (4.6)$$

### 4.3 Arquiteturas Utilizadas

Para fim de experimentos, foram utilizadas oito diferentes arquiteturas na execução dos testes e avaliação da abordagem proposta. Estas arquiteturas se diferem pela quantidade de camadas ocultas que a Rede Neural Artificial tem, bem como o número de neurônios em cada uma destas camadas. Em todas as arquiteturas, as camadas de entrada e saída possuem uma quantidade fixa de neurônios, 41 e 5, respectivamente, respeitando sempre os 41 atributos da base de dados utilizados na camada de entrada, e os 5 diferentes tipos de classificação, correspondente a camada de saída.

O Quadro 1 apresenta a nomenclatura das arquiteturas, de acordo com a composição das mesmas.

Quadro 1 - Nomenclatura das arquiteturas.

Abreviação	Nomenclatura
2C41N	2 camadas ocultas com 41 neurônios cada
2C82N	2 camadas ocultas com 82 neurônios cada
3C41N	3 camadas ocultas com 41 neurônios cada
3C82N	3 camadas ocultas com 82 neurônios cada
4C41N	4 camadas ocultas com 41 neurônios cada
4C82N	4 camadas ocultas com 82 neurônios cada
5C41N	5 camadas ocultas com 41 neurônios cada
5C82N	5 camadas ocultas com 82 neurônios cada

Fonte: PRÓPRIO (2020).

#### 4.4 Ferramentas Utilizadas

Para configuração do ambiente dos experimentos, foi utilizada a ferramenta *Google Colaboratory*, por onde foi possível implementar e executar a abordagem proposta. Para criação e execução da abordagem proposta, foram utilizadas algumas bibliotecas *Python*. A biblioteca *panda* foi utilizada no carregamento da base de dados NSL-KDD, armazenando-a em uma variável. A biblioteca *NumPy*, que é utilizada para manipulação de *arrays* e matrizes, serviu para agrupar os ataques da base de dados em 5 classes diferentes. Também foi utilizada a *Scikit-Learn*, uma biblioteca de aprendizado de máquina, através da qual foi possível utilizar o método de *cross-validation*, realizar os cálculos das métricas utilizadas na avaliação, entre outras tarefas. Para a RNA, foi utilizada a biblioteca *Keras*, que permite a manipulação e configuração de algumas características da RNA, como por exemplo, o número de camadas. Esta biblioteca se trata de uma interface de alto nível para a biblioteca *TensorFlow*, que é a responsável pela criação e execução da RNA.

#### 4.5 Resultados

Nesta Seção são apresentados os resultados dos experimentos realizados com as arquiteturas propostas, demonstrando percentualmente a taxa obtida pelas arquiteturas em cada uma das métricas de avaliação definidas na Seção 3.2. Posteriormente, é realizada uma análise refletindo sobre os resultados obtidos e o motivo das variações observadas, procurando identificar qual das arquiteturas propostas é a mais eficiente.

A Tabela 1 apresenta os resultados dos experimentos com a arquitetura 2C41N.

Tabela 1 - Resultados dos experimentos com a arquitetura 2C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	95,66	4,34	95,16	96,81	94,34	95,96
DoS	97,94	2,06	96,77	97,63	98,12	97,19
probe	98,11	1,89	95,68	84,08	99,55	89,02
R2L	98,99	1,01	23,94	20,12	99,57	21,24
U2R	99,96	0,04	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Os resultados obtidos na detecção das classes *normal*, *DoS* e *probe* se mostraram satisfatórios. Já os resultados obtidos para as classes *R2L* e *U2R* demonstraram uma deficiência da arquitetura na classificação destes ataques, levando em consideração os resultados ruins obtidos no *recall*. O *recall* de cada uma destas classes apresenta a taxa de eventos classificados corretamente como intrusivos (neste caso *R2L* e *U2R*) dentre todos os eventos intrusivos (*R2L* e *U2R*) do conjunto de dados. Então, mesmo que estas classes tenham apresentado acurácias com valores altos, esta arquitetura apresentou grande deficiência na classificação destes ataques, em especial do *U2R*, que obteve o *recall* igual a 0.

A Tabela 2 apresenta os resultados dos experimentos com a arquitetura 2C82N.

Tabela 2 - Resultados dos experimentos com a arquitetura 2C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	95,57	4,43	96,09	95,50	95,66	95,78
DoS	96,10	3,90	92,46	97,80	95,12	94,95
probe	98,37	1,63	96,45	85,93	99,65	90,57
R2L	99,10	0,90	0,00	0,00	99,99	0,00
U2R	99,95	0,05	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Assim como na arquitetura com 41 neurônios, a arquitetura 2C82N apresentou uma deficiência muito grande na classificação dos ataques R2L e U2R, obtendo *recall* igual a 0, ou seja, nenhum ataque destas classes foi classificado corretamente como ataque.

A Tabela 3 apresenta os resultados dos experimentos com a arquitetura 3C41N.

Tabela 3 - Resultados dos experimentos com a arquitetura 3C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	96,04	3,96	94,43	98,41	93,30	96,38
DoS	97,99	2,01	97,52	96,92	98,60	97,22
probe	97,49	2,51	93,81	78,24	99,46	85,32
R2L	99,17	0,83	26,81	17,79	99,81	0,00
U2R	99,96	0,04	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

A arquitetura 3C41N apresentou as mesmas deficiências das arquiteturas de duas camadas, no que diz respeito a classificação dos ataques das classes R2L e U2R, bem como dos ataques do tipo *probe*, que inclusive obteve um *recall* ainda pior, de 78,24%. Um destaque positivo foi a detecção da classe normal, que apresentou um *recall* de 98,41%, superior às arquiteturas 2C41N e 2C82N, que obtiveram 96,81% e 95,50%, respectivamente, representando um ganho considerável para esta classe com o aumento do número de camadas ocultas.

A Tabela 4 apresenta os resultados dos experimentos com a arquitetura 3C82N.

Tabela 4 - Resultados dos experimentos com a arquitetura 3C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	97,21	2,79	96,75	98,14	96,11	97,43
DoS	98,58	1,42	98,34	97,69	99,08	98,01
probe	98,74	1,26	96,02	90,10	99,62	92,96
R2L	98,81	1,19	28,35	32,41	99,36	30,12
U2R	99,96	0,04	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Com relação a arquitetura com 41 neurônios, a 3C82N se saiu melhor na classificação dos ataques do tipo *probe*, obtendo um *recall* de 90,10%, se aproximando das classes que possuem um maior número de instâncias, sendo estas: *normal* e DoS. A classificação para

R2L também obteve um aumento de 14,62% no *recall*, mostrando uma vantagem da arquitetura com 82 neurônios por camada oculta.

A Tabela 5 apresenta os resultados dos experimentos com a arquitetura 4C41N.

Tabela 5 - Resultados dos experimentos com a arquitetura 4C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	95,09	4,91	93,37	97,89	91,83	95,57
DoS	98,65	1,35	99,53	96,71	99,75	98,10
probe	96,44	3,56	82,43	75,79	98,51	78,88
R2L	99,29	0,71	0,00	20,73	100,00	0,00
U2R	99,98	0,02	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Das arquiteturas avaliadas até o momento, a 4C41N foi a que teve o pior desempenho para a classe *probe*, obtendo apenas 75,79% de *recall*, a pior taxa registrada até o momento. Com relação às arquiteturas com 3 camadas, considerando apenas os valores de *recall* obtidos, a única classe em que esta arquitetura apresentou melhor resultado foi a R2L, que apresentou *recall* de 20,73%, contra os 17,79% da arquitetura 3C41N.

A Tabela 6 apresenta os resultados dos experimentos com a arquitetura 4C82N.

Tabela 6 - Resultados dos experimentos com a arquitetura 4C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	94,23	5,77	97,29	91,88	96,95	94,25
DoS	96,31	3,69	94,04	96,43	96,26	95,14
probe	96,35	3,65	79,96	92,56	96,72	84,10
R2L	99,42	0,58	44,85	29,05	99,90	0,00
U2R	99,93	0,07	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

Diferentemente da 4C41N, a arquitetura com 82 neurônios obteve a melhor taxa de *recall* registrada até o momento para a classe *probe*, com 92,56%. A taxa de *recall* dos ataques do tipo R2L também melhorou significativamente, obtendo 29,05%, ficando atrás apenas da arquitetura 3C82N que registrou 32,41% para esta mesma classe. Em contrapartida, a 4C82N obteve as piores taxas de *recall* para as classes *normal* e DoS. Ainda contou com a pior acurácia para a classe *normal* já registrada, com apenas 94,23%, ao contrário das arquiteturas anteriores que não apresentaram resultados inferiores a 95%.

A Tabela 7 apresenta os resultados dos experimentos com a arquitetura 5C41N.

Tabela 7 - Resultados dos experimentos com a arquitetura 5C41N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	97,90	2,10	97,34	98,81	96,83	98,06
DoS	98,99	1,01	98,46	98,80	99,08	98,62
probe	99,16	0,84	99,39	91,48	99,94	95,27
R2L	99,67	0,33	85,26	62,64	99,93	71,54
U2R	99,95	0,05	0,00	0,00	100,00	0,00

Fonte: PRÓPRIO (2020).

A arquitetura 5C41N se destacou pelo *recall* de 62,64% obtido na classificação dos ataques R2L, classe que todas as outras arquiteturas apresentaram dificuldade para identificar. Apesar de não ser um valor alto como nas demais classes, representa um valor satisfatório levando em consideração o baixo número de amostras da classe R2L. Já nos ataques da classe U2R, a arquitetura, assim como as demais, obteve *recall* de 0% devido a pequena quantidade de amostras desta classe presente na base de dados utilizada.

A Tabela 8 apresenta os resultados dos experimentos com a arquitetura 5C82N.

Tabela 8 - Resultados dos experimentos com a arquitetura 5C82N.

Classe	Acurácia (%)	Erro (%)	Precisão (%)	Recall (%)	TNR (%)	F1-Score (%)
normal	97,90	2,10	96,47	99,70	95,90	98,05
DoS	99,10	0,90	99,90	97,66	99,94	98,77
probe	99,26	0,74	99,14	93,09	99,92	95,94
R2L	99,55	0,45	92,65	40,20	99,94	42,34
U2R	99,96	0,04	0,00	0,00	100,00	0,00

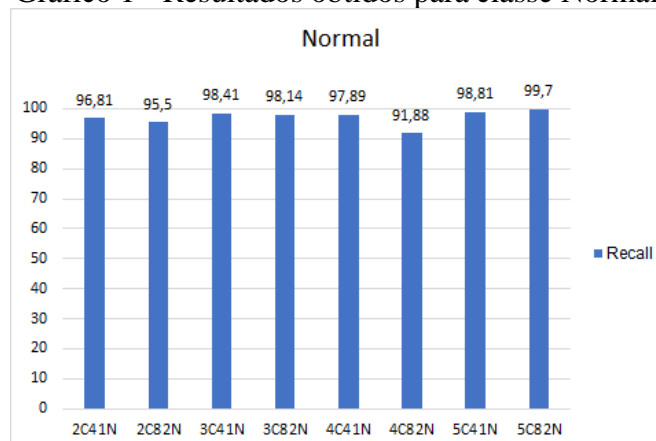
Fonte: PRÓPRIO (2020).

Com relação aos resultados apresentados na arquitetura 5C82N, podemos observar que ela obteve menores valores de *recall* para as classes DoS e R2L em comparação a arquitetura 5C41N. Em contrapartida, houve uma melhora desta métrica nas classes *normal* e *probe*. Apesar de haver algumas melhoras compensando as perdas, é possível afirmar que a arquitetura 5C41N foi melhor no experimento, especialmente devido ao *recall* obtido na identificação dos ataques R2L, de 62,64%, ser muito superior aos 40,20% obtidos na arquitetura 5C82N, representando uma variação muito grande para este tipo de ataque.

Para melhor visualização dos resultados, foram elaborados alguns gráficos comparando o desempenho das arquiteturas utilizadas através da métrica *recall*, demonstrando o quanto cada arquitetura conseguiu detectar em cada uma das 5 classes. Também foi realizada uma comparação específica entre arquiteturas com diferentes números de camadas e com diferentes números de neurônios, a fim de verificar se um dos objetivos específicos deste trabalho foi alcançado.

O Gráfico 1 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe normal.

Gráfico 1 - Resultados obtidos para classe Normal.

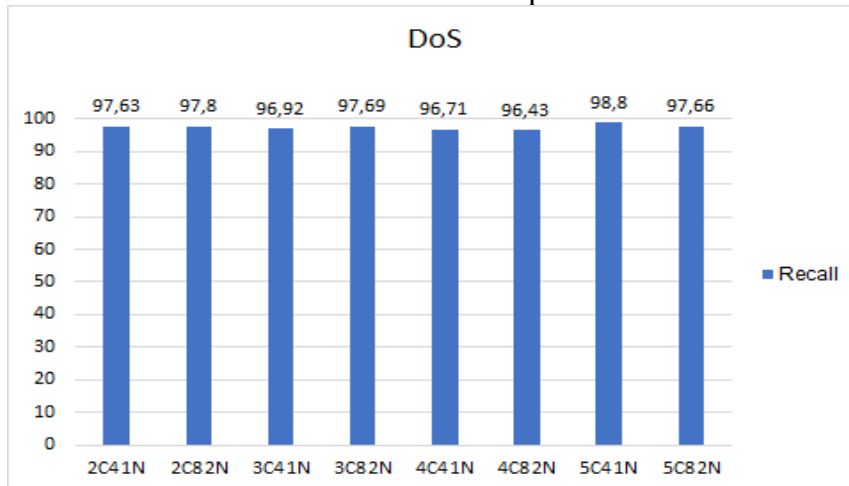


Fonte: PRÓPRIO (2020).

Conforme observado acima, todas as arquiteturas apresentaram valores satisfatórios de *recall* para a classe normal, ou seja, esta classe foi bem detectada nos experimentos. Destaque para as arquiteturas 5C41N e 5C82N, que apresentaram os melhores resultados, com 98,81% e 99,70% respectivamente.

O Gráfico 2 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe DoS.

Gráfico 2 - Resultados obtidos para classe DoS.

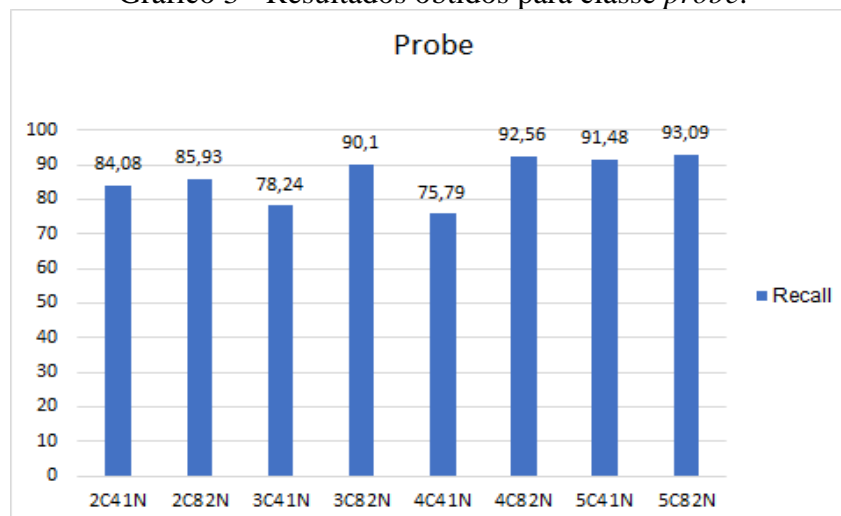


Fonte: PRÓPRIO (2020).

No Gráfico 2 é possível observar que a detecção dos ataques do tipo DoS, assim como na detecção da classe normal, os resultados foram satisfatórios. Esta classe foi a que obteve resultados mais consistentes, variando pouco entre as diferentes arquiteturas. A arquitetura que obteve a melhor taxa de detecção foi a 5C41N, com 98,80% de *recall*.

O Gráfico 3 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe *probe*.

Gráfico 3 - Resultados obtidos para classe *probe*.

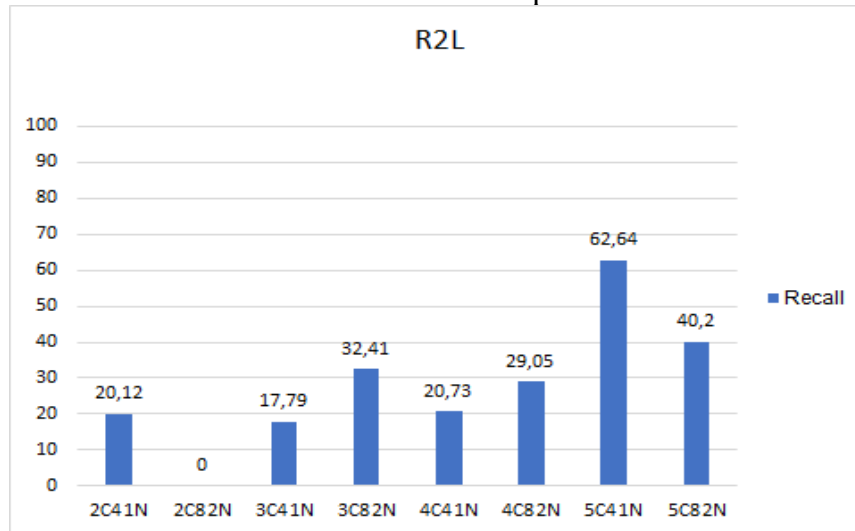


Fonte: PRÓPRIO (2020).

Diferente das classes analisadas anteriormente, a detecção dos ataques do tipo *probe* apresentaram uma maior variação entre as arquiteturas. É possível observar que nas arquiteturas com o mesmo número de camadas ocultas, o aumento do número de neurônios por camada melhorou a detecção desta classe. A arquitetura com o melhor resultado foi a 5C82N, com 93,09%.

O Gráfico 4 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe R2L.

Gráfico 4 - Resultados obtidos para classe R2L.

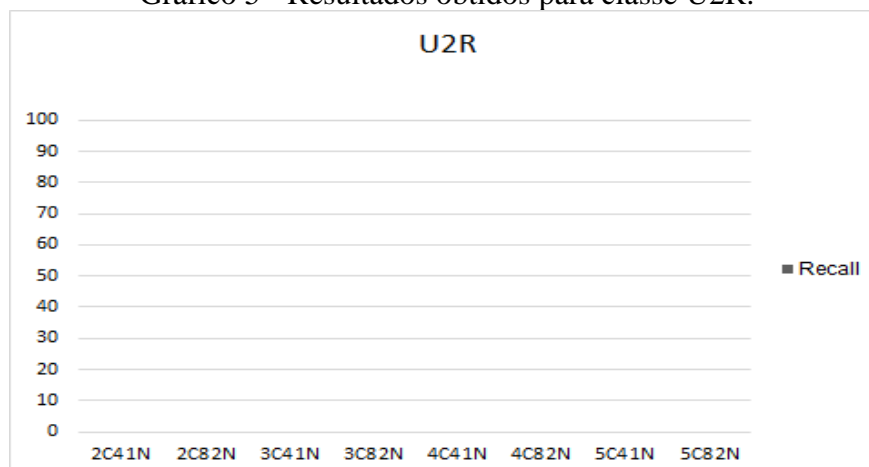


Fonte: PRÓPRIO (2020).

O Gráfico 4 demonstra que os valores de *recall* para os ataques do tipo R2L foram muito inferiores às classes previamente analisadas. A melhor arquitetura para esta classe foi a 5C41N, com 62,64% de detecção, que pode ser visto como um resultado satisfatório levando em consideração os resultados obtidos pelas demais arquiteturas.

O Gráfico 5 apresenta os resultados do *recall* obtido pelas diferentes arquiteturas para classe U2R.

Gráfico 5 - Resultados obtidos para classe U2R.

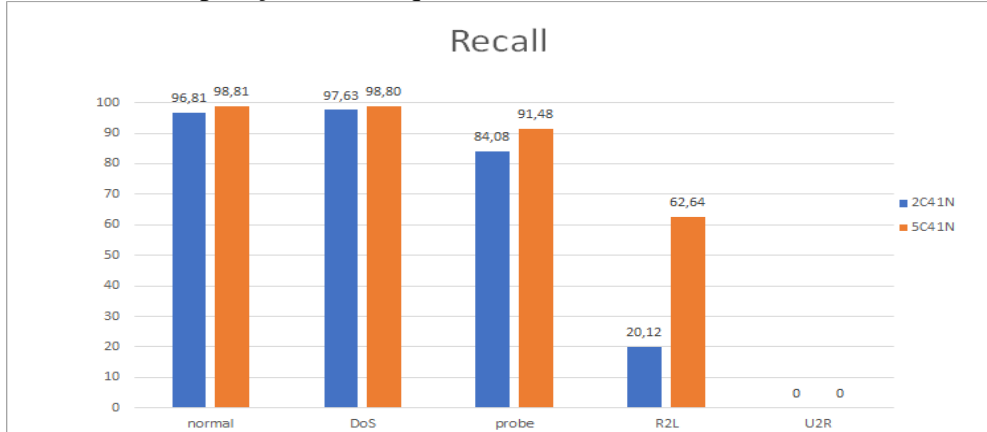


Fonte: PRÓPRIO (2020).

O Gráfico 5 demonstrou que nenhuma arquitetura conseguiu detectar ataques do tipo U2R.

O Gráfico 6 apresenta uma comparação entre arquiteturas com diferentes números de camadas ocultas, com o objetivo de identificar se o aumento deste valor melhorou a detecção das classes.

Gráfico 6 - Comparação entre arquiteturas com diferentes números de camadas.

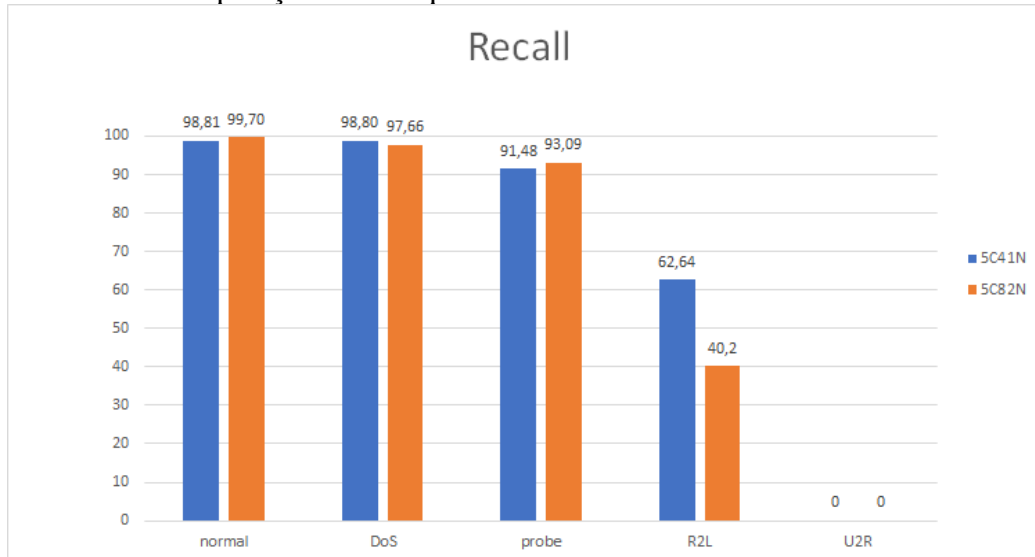


Fonte: PRÓPRIO (2020).

Conforme observado, a arquitetura 5C41N apresentou valores superiores a 2C41N, que utiliza um número menor de camadas ocultas. A principal diferença foi na detecção da classe R2L, onde a arquitetura com 5 camadas obteve 62,64% de *recall* contra os 20,12% da arquitetura mais simples. Portanto, é possível afirmar que o aumento do número de camadas ocultas representou uma melhora na detecção de cada uma das classes utilizadas.

O Gráfico 7 apresenta uma comparação entre arquiteturas com diferentes números de neurônios por camada oculta, com o objetivo de identificar se o aumento deste valor melhorou a detecção das classes.

Gráfico 7 - Comparação entre arquiteturas com diferentes números de neurônios.



Fonte: PRÓPRIO (2020).



Conforme os resultados apresentados acima, não é possível afirmar que o aumento do número de neurônios por camada oculta apresentou resultados melhores ou piores, pois houve uma variação de acordo com as classes. Porém, é possível destacar que a arquitetura 5C41N, com menor número de neurônios, apresentou uma detecção muito melhor para classe R2L do que a arquitetura 5C82N, podendo ser considerada melhor do que a mesma, se for levado em consideração que as variações obtidas nas demais classes são equivalentes.

## 5 CONCLUSÃO E TRABALHOS FUTUROS

Considerando as classes *normal*, DoS e *probe*, é possível afirmar que todas as arquiteturas obtiveram resultados satisfatórios de detecção, com altas taxas de acurácia, precisão e *recall*, especialmente nas duas primeiras classes. De modo geral, o aumento do número de camadas ocultas melhorou os valores da acurácia média. Trabalhos futuros podem realizar experimentos com maiores quantidades de camadas, sempre levando em consideração o custo, pois quanto maior a quantidade de neurônios da RNA, maior o custo de processamento, podendo tornar inviável a utilização da abordagem na prática.

Uma deficiência que pôde ser observada nos resultados comentados anteriormente da abordagem proposta, diz respeito a detecção e classificação dos ataques do tipo U2R e R2L. Os resultados obtidos nestas classes por todas as arquiteturas testadas não se equivalem aos encontrados nas demais, que majoritariamente apresentaram acurácia e precisão acima de 90%, com exceção da classe *probe*, que nas arquiteturas com 4 camadas obteve resultados na casa dos 80%. A classe U2R em especial foi a que apresentou os piores resultados, não sendo detectada por nenhuma das arquiteturas, apresentando sempre um *recall* de 0%.

O principal causador desta deficiência é a base de dados utilizada, que como mencionado na Seção 5.1.3, possui apenas 995 amostras de ataques da classe R2L e 52 da U2R. Porém, mesmo com resultados ruins, podemos considerar que a arquitetura com 5 camadas obteve um resultado satisfatório na detecção dos ataques R2L perante as demais arquiteturas, atingindo 62,64% na taxa de *recall*, no caso da arquitetura com 41 neurônios por camada oculta (5C41N), e 40,20% para a arquitetura 5C82N, que mesmo registrando uma queda significativa com o aumento do número de neurônios, manteve um valor maior do que as demais arquiteturas.

Para evitar este desempenho insatisfatório, trabalhos futuros podem realizar experimentos com o balanceamento de classes, onde são utilizadas técnicas para que as diferentes classes de ataque da base de dados possuam quantidades de amostras semelhantes, evitando a discrepância observada nos resultados desta abordagem.

Trabalhos futuros podem ainda realizar novos estudos em cima de arquiteturas que utilizam 5 camadas ocultas, ou aumentando ainda mais a quantidade de camadas, explorando com maior variedade o número de neurônios em cada uma delas, procurando uma abordagem ainda mais eficiente do que a apresentada neste trabalho.

## REFERÊNCIAS

ALHARBI, Ali; ALHAIDARI, Sulaiman; ZOHDY, Mohamed. Denial-of-Service, Probing, User to Root (U2R) & Remote to User (R2L) Attack Detection using Hidden Markov Models. **International Journal Of Computer And Information Technology**. Rochester, p. 204-210. set. 2018.

AL-HAWAWREH, Muna; MOUSTAFA, Nour; SITNIKOVA, Elena. **Identification of malicious activities in industrial internet of things based on deep learning models.** *Journal Of Information Security And Applications*, [s.l.], v. 41, p.1-11, ago. 2018.

ALJAWARNEH, Shadi; ALDWAIRI, Monther; YASSEIN, Muneer Bani. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal Of Computational Science*, [S.L.], v. 25, p. 152-160, mar. 2018.

ALMIANI, Muder; ABUGHAZLEH, Alia; AL-RAHAYFEH, Amer; RAZAQUE, Abdul. Cascaded hybrid intrusion detection model based on SOM and RBF neural networks. *Concurrency And Computation: Practice and Experience*, [S.L.], v. 32, n. 21, p. 1-14, 7 mar. 2019.

AL-SARAWI, Shadi et al. Internet of Things (IoT) communication protocols: Review. **2017 8th International Conference On Information Technology (icit)**, [s.l.], p.685-690, maio 2017.

BACE, Rebecca; MELL, Peter. **NIST Special Publication on Intrusion Detection Systems.** Scotts Valley: National Institute Of Standards And Technology, 2001.

BONOMI, Flavio et al. Fog computing and its role in the internet of things. **Proceedings Of The First Edition Of The Mcc Workshop On Mobile Cloud Computing - Mcc '12**, [s.l.], p.13-16, 2012.

BUTUN, Ismail; MORGERA, Salvatore D.; SANKAR, Ravi. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials*, [s.l.], v. 16, n. 1, p.266-282, 2014.

CAMPELLO, Rafael; WEBER, Raul. **Sistemas de Detecção de Intrusão.** Florianópolis: Xix Simpósio Brasileiro de Redes de Computadores, 2001. 173 slides, color.

GARDNER, M.W; DORLING, S.R. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, [S.L.], v. 32, n. 14-15, p. 2627-2636, ago. 1998. Elsevier BV.

HAYKIN, Simon. **Redes Neurais: Princípios e Prática.** [S. L]: Bookman, 2001.

ILGUN, K.. USTAT: a real-time intrusion detection system for UNIX. **Proceedings 1993 IEEE Computer Society Symposium On Research In Security And Privacy**, [s.l.], p.16-28, 1993.

IORGA, Michaela et al. **Fog Computing Conceptual Model.** [s.l.]: National Institute Of Standards & Technology, 2018.

JAFIER, Shatha H.. Utilizing feature selection techniques in intrusion detection system for internet of things. **Proceedings Of The 2nd International Conference On Future Networks And Distributed Systems - Icfnds '18**, [s.l.], p.1-3, 2018.

JEYA, P. Gifty; MURALIDHARAN, Ravichandran; RAVICHANDRAN, C. S.. Efficient Classifier for R2L and U2R Attacks. **International Journal Of Computers And Applications**. [S. L.], p. 28-32. jan. 2012.

JHAVERI, Rutvij H.; PATEL, Sankita J.; JINWALA, Devesh C.. DoS Attacks in Mobile Ad Hoc Networks: a survey. **2012 Second International Conference On Advanced Computing & Communication Technologies**, [S.L.], p. 535-541, jan. 2012. IEEE.

KHAMPHAKDEE, Nattawat; BENJAMAS, Nunnapus; SAIYOD, Saiyan. Improving Intrusion Detection System based on Snort rules for network probe attack detection. **2014 2Nd International Conference On Information And Communication Technology (Icoict)**, [S.L.], p. 69-74, maio 2014.

KOLIAS, Constantinos *et al.* DDoS in the IoT: mirai and other botnets. **Computer**, [S.L.], v. 50, n. 7, p. 80-84, 2017. Institute of Electrical and Electronics Engineers (IEEE).

MACHADO, Renato Bobsin. **UMA ABORDAGEM DE DETECÇÃO DE INTRUSÃO BASEADA EM SISTEMAS IMUNOLÓGICOS ARTIFICIAIS E AGENTES MÓVEIS**. 2005. 164 f. Dissertação (Mestrado) - Curso de Pós-graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2005.

MELL, Peter.; GRANCE, Timothy. **The NIST Definition of Cloud Computing**. Gaithersburg: National Institute Of Standards & Technology, 2011.

MUKHERJEE, B.; HEBERLEIN, L.t.; LEVITT, K.n.. Network intrusion detection. **IEEE Network**, [s.l.], v. 8, n. 3, p.26-41, maio 1994.

NAQA, Issam El; MURPHY, Martin J.. What Is Machine Learning? Machine Learning In Radiation Oncology, [S.L.], p. 3-11, 2015. Springer International Publishing.

NORTHCUTT, Stephen et al. **Intrusion Signatures and Analysis**. Thousand Oaks: New Riders Publishing, 2001.

SHEIKHAN, Mansour; BOSTANI, Hamid. **A hybrid intrusion detection architecture for Internet of things**. **2016 8th International Symposium On Telecommunications (ist)**, [s.l.], p.601-606, set. 2016.

SOUZA, Cristiano Antonio de. **MÉTODO HÍBRIDO DE DETECÇÃO DE INTRUSÃO APLICANDO INTELIGÊNCIA ARTIFICIAL**. 2018. 142 f. Dissertação (Mestrado) - Curso de Programa de Pós-graduação em Engenharia Elétrica e Computação, Universidade Estadual do Oeste do Paraná, Foz do Iguacu, 2018.

SPAFFORD, Eugene H; ZAMBONI, Diego. Intrusion detection using autonomous agents. **Computer Networks**, [s.l.], v. 34, n. 4, p.547-570, out. 2000.

TAVALLAEE, Mahbod et al. A detailed analysis of the KDD CUP 99 data set. **2009 Ieee Symposium On Computational Intelligence For Security And Defense Applications**, [S.L.], p. 1-6, jul. 2009. IEEE.

U.FAROOQ, M. et al. A Review on Internet of Things (IoT). **International Journal Of Computer Applications**, [S.L.], v. 113, n. 1, p. 1-7, 18 mar. 2015. Foundation of Computer Science.

WORTMANN, Felix; FLÜCHTER, Kristina. Internet of Things: Technology and Value Added. **Internet Of Things. Business & Information Systems Engineering**, [s. L], p.221-224, jun. 2015.

XIE, Junfeng *et al.* A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): research issues and challenges. **Ieee Communications Surveys & Tutorials**, [S.L.], v. 21, n. 1, p. 393-430, 2019. Institute of Electrical and Electronics Engineers (IEEE).

YANNUZZI, M. et al. Key ingredients in an IoT recipe: fog computing, cloud computing, and more fog computing. **2014 Ieee 19Th International Workshop On Computer Aided Modeling And Design Of Communication Links And Networks (Camad)**, [S.L.], p. 325-329, dez. 2014.