UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rodrigo Rodrigues Pires de Mello

A neural-symbolic BDI-agent as a Multi-Context System

Florianópolis
2024

Rodrigo Rodrigues Pires de Mello

A neural-symbolic BDI-agent as a Multi-Context System

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de Doutor em Ciência da Computação.
Orientador: Prof. Ricardo Azambuja Silveira, Dr.
Coorientador: Prof. Rafael de Santiago, Dr.

Florianópolis
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Mello, Rodrigo Rodrigues Pires de
A neural-symbolic BDI-agent as a Multi-Context System / Rodrigo Rodrigues Pires de Mello ; orientador, Ricardo Azambuja Silveira, coorientador, Rafael de Santiago, 2024.
140 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2024.

Inclui referências.

1. Ciência da Computação. 2. Inteligência Artificial. 3. Agentes neuro-simbólicos. 4. Redes Neurais. 5. Agentes inteligentes. I. Silveira, Ricardo Azambuja. II. Santiago, Rafael de. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

Rodrigo Rodrigues Pires de Mello

A neural-symbolic BDI-agent as a Multi-Context System

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

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

ICP-Edu
Documento assinado digitalmente
Marcio Bastos Castro
Data: 01/08/2024 18:10:43-0300
CPF: ***.707.720-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

GOV.BR
Documento assinado digitalmente
RICARDO AZAMBUJA SILVEIRA
Data: 01/08/2024 11:01:25-0300
CPF: ***.225.790-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ricardo Azambuja Silveira, Dr.
Orientador

Florianópolis, 2024.

AGRADECIMENTOS

À minha família, pelo apoio durante todos esses anos. À minha mãe, Angela, pelo amor incondicional e pelos seus conselhos. Ao meu pai, Nersí Pires de Mello (*em memória*), que não pode acompanhar a minha trajetória acadêmica, mas sempre me incentivou nos estudos. Às minhas irmãs, Mariana e Maieli, pelo suporte e carinho. Ao meu irmão, Diogo, que serviu de inspiração para a escolha da minha profissão.

Ao meu orientador, Ricardo Azambuja Silveira, pelo aprendizado e por ter contribuído significativamente na minha trajetória acadêmica.

Ao meu coorientador, Rafael de Santiago, pela ajuda e pelo tempo dedicado ao desenvolvimento deste trabalho.

Aos membros das bancas de qualificação e de defesa: Mateus Grellert da Silva, Jomi Fred Hübner, Anarosa Alves Franco Brandão, Diana Francisca Adamatti e Mauro Roisenberg, pelas contribuições a este trabalho.

Aos meus amigos José, Rafael, Luiz Filipe, Jean, Gabriela, Artur, Thiago e Luis, que me apoiaram durante o doutorado.

O presente trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

“Du brauchst immer nur genug Mut für den nächsten Schritt, nicht für die ganze Treppe.”

Unbekannter Autor

ABSTRACT

Intelligent systems have been deployed in several fields (e.g., medicine, education, automation, or legal), providing relevant tools to assist with our daily tasks. Traditionally, two categories can separate AI methods: symbolic and connectionist. Conciliating the statistical nature of learning with the logical nature of reasoning, aiming to integrate concept acquisition and manipulation, has been identified as a key research challenge and fundamental problem in computer science. Initially, we performed a Systematic Literature Mapping (SLM) to investigate how different works combine neural networks and intelligent agents to develop intelligent systems. Our findings show that most studies did not integrate learning into the agent's reasoning cycle; they mainly focused on using neural networks to define a policy for reinforcement learning agents. Aiming to bring the best of both methods, a field of study called neural-symbolic seeks to build modular systems that can learn from the environment and reason from what was learned. To integrate these two methods, this research proposes a neural-symbolic BDI (Beliefs, Desires, and Intentions)-agent based on Multi-Context Systems (MCS). MCSs allow the modular representation of information exchange among heterogeneous sources. BDI-agents offer robust and flexible behavior, rapid and modular development, intelligibility, and verifiability. We present case studies and two experiments to evaluate the proposed agent model. The case studies describe the integration method and how the proposed agents can be developed in the Sigon framework. Sigon enables the development of MCS agents in which different contexts can be integrated at a programming language level. The results from the first experiment showed that the proposed agent adapted to different situations and achieved the maximum utility value. However, executing the neural network in every reasoning cycle could not be adequate in some scenarios that require rapid responses. In the second experiment, compared to the baseline version, which achieved an accuracy of 84.6%, the neural-symbolic agent achieved 84.4% and was 48% faster during the training phase. As the last step of this research, we executed a new SLM, which focuses on how different studies modeled, developed, evaluated, and deployed neural-symbolic agents. Based on the results, the contributions of this research are: (i) integration of connectionist methods as part of the agent's decision-making; (ii) practical implementation of the integration method; (iii) the proposed integration method can mitigate the necessity of hard-coded and hand-crafted rules; and (iv) a neural-symbolic agent that improves the neural network's performance, thus enhancing the intelligent systems' decision-making.
Keywords: symbolic. connectionist. agents. neural-symbolic. multi-context systems. BDI. Sigon.

RESUMO

Sistemas inteligentes estão sendo implantados em diversos campos, como por exemplo: saúde, educação, automações e jurídico. Tradicionalmente, esses sistemas são implementados através de dois métodos de Inteligência Artificial: simbólica e conexionista. A IA simbólica opera através da realização de uma sequência de etapas de raciocínio lógico sobre símbolos que consistem em representações semelhantes à linguagem. A IA conexionista refere-se à adição de conhecimento por meio da atribuição de condutividades numéricas ou pesos às conexões dentro de uma rede. Um problema fundamental na ciência da computação é definir como conciliar a natureza estatística da aprendizagem com a natureza lógica do raciocínio, visando construir modelos computacionais robustos que integrem a aquisição e a manipulação de conceitos. Inicialmente, um Mapeamento Sistemático da Literatura (MSL) foi realizado para investigar como diferentes trabalhos combinam redes neurais e agentes inteligentes para desenvolver sistemas inteligentes. Os principais resultados desse mapeamento mostraram que a maioria dos estudos não integram a aprendizagem ao ciclo de raciocínio do agente, concentrando-se principalmente no uso de redes neurais para definir uma política de recompensa para agentes de aprendizagem por reforço. Com o objetivo de obter o melhor de ambos os métodos, o campo de estudo do neuro-simbólico busca construir sistemas modulares que possam aprender através de interações com o ambiente e raciocinar a partir do que foi aprendido. Desta forma, este trabalho propõe a utilização de agentes-BDI neuro-simbólico baseados em Sistemas Multi-Contexto (SMC). Os SMC permitem a representação da troca de informações entre fontes heterogêneas. Os agentes BDI fornecem um comportamento robusto e flexível, um desenvolvimento rápido e modular, e propriedades de inteligibilidade e verificabilidade. Para avaliar o agente proposto, um estudo de caso e dois experimentos são executados. Nos estudos de caso, descreve-se os detalhes do método de integração e como os agentes são desenvolvidos no framework Sigon. O Sigon permite o desenvolvimento de agentes SMC, nos quais os contextos podem ser integrados com os existentes a nível de linguagem de programação. Os resultados do primeiro experimento mostraram que o agente proposto adaptou-se a diferentes situações e atingiu o valor máximo da função de utilidade. Porém, a execução da rede neural em todos os ciclos de raciocínio pode não ser adequada em cenários que exigem respostas rápidas. No segundo experimento, comparado à versão de referência, que alcançou acurácia de 84.6%, o agente neuro-simbólico atingiu 84.4% e foi 48% mais rápido durante a fase de treinamento. Como última etapa desta pesquisa, um novo MSL foi executado. Esse MSL concentrou-se em estudar como diferentes trabalhos modelaram, desenvolveram, avaliaram e implantaram agentes neuro-simbólicos. Com base nas atividades realizadas, as contribuições desta pesquisa são as seguintes: (i) modelagem de uma integração de métodos conexionistas como parte da tomada de decisão do agente, possibilitando o desenvolvimento de agentes de forma modular e flexível; (ii) implementação prática do método de integração proposto; (iii) o método de integração pode mitigar a necessidade de regras codificadas e elaboradas manualmente; e (iv) um agente neural-simbólico capaz de melhorar sua tomada de decisão através da otimização do desempenho da rede neural.
Palavras-chave: simbólico. conexionista. agentes. neuro-simbólico. sistemas multi-contexto. BDI. Sigon.

RESUMO EXPANDIDO

Introdução

Sistemas inteligentes estão sendo implantados em diversos campos, como por exemplo: saúde, educação, automações e jurídico. Tradicionalmente, esses sistemas são implementados através de dois métodos de Inteligência Artificial: simbólica e conexionista. A IA simbólica opera através da realização de uma sequência de etapas de raciocínio lógico sobre símbolos que consistem em representações semelhantes à linguagem. A IA conexionista refere-se à adição de conhecimento por meio da atribuição de condutividades numéricas ou pesos às conexões dentro de uma rede (MINSKY, 1991). Artur d'Avila Garcez et al. (2019) defendem que, apesar das suas diferenças, tanto a IA simbólica como o conexionista compartilham características comuns que oferecem benefícios quando integradas de forma apropriada. Um problema fundamental na ciência da computação é definir como conciliar a natureza estatística da aprendizagem com a natureza lógica do raciocínio, visando construir modelos computacionais robustos que integrem a aquisição e a manipulação de conceitos (BESOLD; GARCEZ, et al., 2017; VALIANT, 2003). Como passo inicial desta pesquisa, um Mapeamento Sistemático da Literatura (MSL) investigou como diferentes trabalhos combinam redes neurais e agentes inteligentes no desenvolvimento de sistemas inteligentes (MELLO, R. R. P. d.; SILVEIRA; SANTIAGO, 2021). Os principais resultados desse mapeamento mostraram que a grande parte dos estudos não integram a aprendizagem ao ciclo de raciocínio do agente, concentrando-se principalmente no uso de redes neurais para definir uma política de recompensa para agentes de aprendizagem por reforço. Uma vez que as abordagens simbólica e conexionista são complementares, a integração de ambas poderia resolver as dificuldades na aquisição de conhecimento enfrentado pelos sistemas simbólicos e a capacidade de raciocinar na presença de informações ruidosas ou incertas (HITZLER et al., 2020). Considerando essas afirmações, define-se a seguinte pergunta de pesquisa: como aprimorar sistemas inteligentes, através da integração de métodos conexionistas no ciclo de raciocínio do agente? Com base nos resultados do Mapeamento Sistemático da Literatura e nos aspectos apresentados anteriormente, esse trabalho defende que o campo de neuro-simbólica pode fornecer as propriedades necessárias para essa integração. Desta forma, este trabalho propõe a utilização de agentes-BDI neuro-simbólico baseados em Sistemas Multi-Contexto (SMC) para desenvolver a integração entre esses métodos. Os SMC permitem a representação da troca de informações entre fontes heterogêneas (CABALAR et al., 2019; BREWKA; EITER, 2007; BREWKA; EITER; FINK, 2011; BREWKA; ELLMAUTHALER; PÜHRER, 2014). Cada contexto descreve diferentes fontes de informações que podem ser integradas a outros contextos por meio de regras especiais chamadas de regras de ponte (CABALAR et al., 2019). Bordini et al. (2020) argumentam que os agentes BDI oferecem comportamento robusto e flexível, desenvolvimento rápido e modular, inteligibilidade e verificabilidade. Levando essas abordagens em consideração, nossa hipótese de pesquisa é definida da seguinte forma: agentes BDI e Sistemas Multi-Contexto podem fornecer uma integração modular e eficaz de métodos conexionistas no ciclo de raciocínio do agente.

Objetivos

O principal objetivo desta pesquisa é aprimorar sistemas inteligentes através da integração da aprendizagem como parte do ciclo de raciocínio de um agente BDI. Os objetivos específicos desta pesquisa tomam como base os três principais campos envolvidos no desenvolvimento de agentes inteligentes: teoria dos agentes, linguagem e arquitetura. Desta forma, essa pesquisa possui os seguintes objetivos específicos: (i) especificar o modelo de um agente inteligente utilizando a arquitetura BDI e baseando-se em Sistemas Multi-Contexto; (ii) investigar diferentes maneiras de integrar métodos conexionistas em diferentes etapas do ciclo de raciocínio do agente; (iii) implantar o agente em um cenário do mundo real; e (iv) avaliar e validar o agente proposto.

Metodologia

Este trabalho segue a metodologia chamada *Design Science Research Methodology* (DSRM) (PEFFERS et al., 2007). Essa metodologia consiste na construção e avaliação iterativa de artefatos. O processo DSRM é separado em seis atividades: identificação e motivação do problema, definição dos objetivos para uma solução, design e desenvolvimento, demonstração, avaliação e comunicação. Nessa pesquisa foram realizadas três iterações. Na primeira iteração, as atividades executadas consistem na execução do Mapeamento Sistemático da Literatura (MSL), na modelagem inicial do agente proposto nesta pesquisa e também um estudo de caso. A modelagem do agente utiliza o framework Sigon (GELAIM et al., 2019). Sigon permite o desenvolvimento de agentes MCS, nos quais diferentes contextos podem ser integrados em nível de linguagem de programação (GELAIM et al., 2019). A principal limitação encontrada no artefato resultante da primeira iteração é a falta de avaliações quantitativas. Desta forma, a segunda iteração focou em aprimorar e avaliar quantitativamente o modelo proposto. A avaliação nesta segunda iteração explorou como a utilização de uma rede neural poderia mitigar a necessidade de regras codificadas manualmente no agente. Essa necessidade caracteriza-se como uma das principais limitações de métodos simbólicos. A terceira iteração dessa metodologia verificou se o agente neuro-simbólico é capaz de aprimorar a sua tomada de decisão através da utilização de estratégias para melhorar a performance da rede neural. Nessa iteração também foi executada um novo Mapeamento Sistemático da Literatura, cujo principal objetivo foi analisar como os trabalhos recentes modelam, desenvolvem, implantam e avaliam agentes neuro-simbólicos.

Resultados e Discussão

Por meio da construção e avaliação iterativa dos artefatos construídos nesta pesquisa, foi possível definir como o modelo proposto aprimora a tomada de decisão de um agente através da integração de métodos conexionistas como parte do raciocínio do agente. Na primeira iteração um Mapeamento Sistemático da Literatura (MSL) foi executado. Mais de 1000 artigos foram analisados e após a execução dos critérios de inclusão/exclusão restaram apenas 110 artigos. A descoberta mais relevante deste SLM foi que apenas 5% dos estudos exploraram a integração de redes neurais como parte do ciclo de raciocínio do agente. Como resultado da primeira iteração também definiu-se a versão inicial do método de integração. O método de integração consiste na modelagem de sensores e contextos customizáveis para lidar com diferentes tipos de dados e métodos de IA. O framework Sigon (GELAIM et al., 2019) foi utilizado para implementar o modelo proposto. Foram desenvolvidas algumas mudanças na gramática Sigon e também a implementação de uma nova versão em Python. Esta decisão permitiu-nos acomodar o modelo proposto e facilitar a integração entre as bibliotecas de ML mais relevantes. Como resultado do segundo artefato, aprimorou-se o método de integração através do uso da saída da rede neural como parte de uma regra de ponte. Os experimentos executados no segundo artefato mostraram que o agente adaptou-se a diferentes situações sem necessidade de adição de novas regras e atingiu o mesmo valor da função de utilidade dos agentes negociadores disponíveis no framework GENIUS. GENIUS é a ferramenta oficial utilizada na Competição de Agentes Negociadores Automatizados (ANAC), que auxilia a comunidade de pesquisa a comparar e avaliar agentes negociadores (JONKER et al., 2017). No terceiro artefato, explorou-se como um agente neuro-simbólico pode melhorar o desempenho da rede neural. Os resultados do experimento mostraram que, quando comparado à versão de referência que atingiu 84.6% de acurácia, o agente neuro-simbólico atingiu 84.4%, sendo 48% mais rápido no treinamento. Um novo Mapeamento Sistemático de Literatura (SLM) foi realizado como parte da terceira iteração. Ele permitiu analisar como diferentes trabalhos modelam, avaliam e implantam agentes neurais-simbólicos. 58 artigos foram analisados e os principais achados foram as seguintes: (i) as metodologias de classificação de sistemas neuro-símbolicos *learning for reasoning* e *reasoning for learning* foram usadas em 59% dos trabalhos recuperados; (ii) 88,2% dos trabalhos focaram nas áreas de estudo de planejamento, processamento de linguagem natural e aprendizagem por reforço; e (iii) 88% comparam o agente neuro-simbólico com diferentes trabalhos ou com diferentes versões implementadas em sua própria pesquisa. Neste trabalho empregou-se a saída da RNA como parte da tomada de decisão do agente. No entanto, também explorou-se diferentes abordagens, como por exemplo, a utilização da saída da RNA no corpo de uma regra de ponte e na alteração dos parâmetros da rede neural durante o ciclo de raciocínio do agente. Este trabalho comparou o agente proposto com diversos trabalhos de referência e com abordagens que utilizavam apenas um método de IA. Considerando estes resultados, o agente e método de integração proposto nesta pesquisa mitiga a necessidade de regras codificadas manualmente e melhora o desempenho da rede neural, de forma a aprimorar a tomada de decisão dos sistemas inteligentes nestes cenários.

Considerações Finais

Nesta pesquisa um agente neuro-simbólico foi modelado, implementado, avaliado e implantado em diferentes cenários. Para modelar e implementar a integração entre métodos simbólicos e conexionistas, Sistemas Multi-Contexto (SMC) e a arquitetura BDI foram utilizados. Considerando os resultados desta pesquisa, é possível notar que o modelo apresentado pode melhorar a tomada de decisão do agente, aumentando o nível de abstração e proporcionando uma integração flexível e modular. Este resultado é alcançado combinando Sistemas Multi-Contexto (SMC), contextos e sensores customizados, e Sigon como uma linguagem de alto nível. O modelo proposto pode servir de base para a implementação de uma arquitetura capaz de integrar e empregar diversos métodos de IA durante a tomada de decisão do agente. Essa abordagem aumenta a modularidade no desenvolvimento de agentes inteligentes capazes de utilizar diferentes técnicas de IA. As contribuições desta pesquisa podem ser definidas da seguinte forma: (i) integração de métodos conexionistas como parte da tomada de decisão do agente; (ii) sensores customizados para lidar com diferentes tipos de dados; (iii) implementação prática do método de integração proposto; e (iv) experimentos do método de integração proposto, com foco em mitigar a necessidade de regras

codificadas e elaboradas manualmente e melhorar a performance de redes neurais. Com base na modelagem do agente proposto e na taxonomia de sistemas neurais-simbólicos apresentada em (YU et al., 2021), esta pesquisa abordou os seguintes métodos de integração: (i) aprendizagem para o raciocínio: o agente negociador empregou a saída da rede neural durante sua tomada de decisão; e (ii) raciocínio para o aprendizado: melhora da precisão e tempo para completar o treinamento de uma Rede Neural Artificial através da alteração de seus parâmetros por meio da tomada de decisão do agente neuro-simbólico. Neste trabalho, não foi avaliado se o método proposto poderia modelar sistemas em que as abordagens simbólicas e conexionistas desempenham papéis semelhantes e trabalham de forma mutuamente benéfica. Para mitigar as limitações desta pesquisa e aprimorar o modelo proposto em trabalhos futuros, os seguintes aspectos devem ser considerados: evoluir o agente proposto para uma arquitetura mais genérica, de forma a explorar e avaliar a integração de diferentes métodos de IA durante a tomada de decisão do agente; explorar se o modelo proposto pode mitigar algumas limitações dos métodos conexionistas; aprimorar a implementação do Sigon, tendo como foco a disponibilização de uma documentação adequada, otimização da ferramenta e criação de testes unitários; implantar e analisar o funcionamento do agente proposto em cenários sem que seja necessária a intervenção humana.

Palavras-chave: simbólico. conexionista. agentes. neuro-simbólico. sistemas multi-contexto. BDI. Sigon.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AAT	Aspiration Adaptation Theory
AI	Artificial Intelligence
ANAC	Automated Negotiating Agents Competition
BC	Beliefs Context
BDI	Beliefs, Desires, and Intentions
CC	Communication Context
CL	Continual Learning
CNN	Convolutional Neural Network
DSRM	Design Science Research Methodology
EMBER	Elastic Malware Benchmark for Empowering Researchers
GENIUS	General Environment for Negotiation with Intelligent multi- purpose Usage Simulation
IC	Intentions Context
LOC	Lines of Code
MC	Metrics Context
MCS	Multi-Context Systems
ML	Machine Learning
MLP	MultiLayer Perceptron
NC	Negotiation Context
NN	Neural Networks
NNC	Neural Network Context
PC	Planner Context
PICOC	Population, Intervention, Comparison, Outcomes, and Context
ReLU	Rectified Linear Unit
SLM	Systematic Literature Mapping
SLR	Systematic Literature Review

CONTENTS

1 INTRODUCTION

Intelligent systems are being deployed in several fields (e.g., medicine, education, automation, or legal), providing relevant tools to support our daily tasks. It is claimed that the next step of Artificial Intelligence (AI) is the integration of connectionist and symbolic methods. Connectionist methods embody knowledge by assigning numerical conductivities or weights to the connections inside a network of nodes (MINSKY, 1991). Symbolic methods work by carrying on a sequence of logic-like reasoning steps over a set of symbols consisting of language-like representations (GARNELO; ARULKUMARAN; SHANAHAN, 2016). Artur d'Avila Garcez et al. (2019) defend that despite their differences, both the symbolic and connectionist methods share common characteristics offering benefits when integrated in a principled way.

Combining the strengths of connectionist and symbolic methods can allow the development of robust intelligent systems. On the one hand, connectionist methods, also known as Neural Networks (NN), stand out in learning from large amounts of data and identifying complex patterns. For instance, NN has been used for image recognition, natural language processing, and speech recognition. On the other hand, symbolic methods are suitable for tasks that require reasoning and manipulation of abstract concepts. They help develop expert systems in which significant knowledge is modeled in a symbolic form. The question of how to conciliate the statistical nature of learning with the logical nature of reasoning, aiming to build such robust computational models integrating concept acquisition and manipulation, has been identified as a key research challenge and fundamental problem in computer science (BESOLD; GARCEZ, et al., 2017; VALIANT, 2003).

Bordini et al. (2020) argue that Machine Learning (ML) has limitations and cannot form the sole basis of autonomous systems capable of intelligent behavior in complex environments. Systems that rely on deep learning frequently have to generalize beyond the specific data that they have seen, whether to a new pronunciation of a word or to an image that differs from one that the system has seen before, the ability of formal proofs to guarantee high-quality performance is more limited (MARCUS, 2018). Employing only connectionist methods can lead to the following disadvantages: (i) lack of interpretability and explainability, and (ii) highly dependent on the training data they process. These two limitations could make systems opaque and hard to interpret, creating the inability to extrapolate results to unseen instances or data that do not follow a similar distribution as the training data (PARISOTTO et al., 2016; BORDINI et al., 2020; MARCUS, 2018; ILKOU; KOUTRAKI, 2020).

Symbolic methods enable the use of deep human expert knowledge in their design and function (SARKER et al., 2021), resulting in expert systems that are easily understandable and transparent. However, these systems can require hand-crafted and hard-coded rules, leading to the Knowledge Acquisition Bottleneck, which refers to the cost of humans converting real-world problems into symbolic systems (ILKOU; KOUTRAKI, 2020). These disadvantages could make systems less efficient and hard to maintain the rules (ARRIETA et al., 2019; ANJOMSHOAE et al., 2019). Symbolic systems can be unsuitable when the environment is continuous and unstructured, in which manually extracting an ad-hoc symbolic model to perform planning may be infeasible (MOON, 2021; UMILI et al., 2021).

As an initial step to investigate how different studies combine connectionist and symbolic methods, we executed a Systematic Literature Mapping (SLM). An SLM differs from a Systematic Literature Review (SLR) in the sense that it presents a broader overview about a field of study, establishes the existence of research evidence, and provides an indication of the quantity of the evidence (KITCHENHAM; CHARTERS, 2007). Our SLM focuses on finding evidence about how different studies employ neural networks and intelligent agents to solve problems. The paradigm of developing agents as a knowledge-based system can be seen as a relevant approach of symbolic AI (WOOLDRIDGE; JENNINGS, 1995), and neural networks represent one of the main approaches in the connectionist AI.

According to the description presented in Chapter 4.1, this Systematic Literature Mapping (SLM) followed the guideline presented in Kitchenham and Charters (2007). We analyzed 1019 papers from Scopus and ACM, and 110 remained after applying the inclusion and exclusion criteria. We compiled them to answer the following research questions: (i) which class of agents and neural networks architecture are being employed; (ii) how these studies combine neural networks and agents; and (iii) which scenarios are these intelligent systems being deployed. Based on these research questions, the main findings of our SLM were the following (see Section 4.2 for details):

1. 64% of studies use neural networks to define the learning agent's reward policies;
2. 5% of studies explore the integration of neural networks as part of the agent's reasoning cycle;
3. although 55% of studies' main contributions are related to neural networks and agents design, the remaining 45% of the studies use both agents and neural networks to solve or contribute to a particular field of research or application.

Garnelo and Shanahan (2019a) state three relevant reasons for the necessity of using symbolic methods during intelligent system modeling. First, due to their declarative nature, symbolic representations lend themselves to re-use in multiple tasks, promoting data efficiency. Second, symbolic representations tend to be high-level and abstract, facilitating generalization. Third, symbolic representations are amenable to human understanding because of their language-like, propositional character. One of the main challenges toward combining symbolic and connectionist methods is integrating cognitive abilities, such as learning, reasoning, and knowledge representation (BESOLD; GARCEZ, et al., 2017).

1.1 RESEARCH QUESTION AND HYPOTHESIS

Since symbolic and connectionist approaches are complementary, integrating both could address the knowledge acquisition bottleneck faced by symbolic systems and the ability to reason in the presence of noisy or uncertain facts (HITZLER et al., 2020). Considering these statements, the research question is defined as:

How can intelligent systems be enhanced by integrating connectionist methods into the agent's reasoning cycle?

Connectionist methods are suitable for scenarios that require processing raw data, while symbolic systems are for scenarios in which it is necessary to represent and reason about some knowledge (YU et al., 2021). Based on our SLM findings and the aspects previously presented, we claim that the neural-symbolic field can provide a promising path toward this integration. Our work considered that the neural-symbolic area explores the effective integration of connectionist and symbolic methods, more precisely learning and reasoning (PARISOTTO et al., 2016). Garcez, Besold, et al. (2015) defend that neural-symbolic is suitable where large amounts of heterogeneous data exist, and knowledge descriptions are required as well.

We propose a neural-symbolic BDI-agent based Multi-Context Systems (MCS) to integrate these two methods. MCSs allow the representation of information exchange among heterogeneous sources (CABALAR et al., 2019; BREWKA; EITER, 2007; BREWKA; EITER; FINK, 2011; BREWKA; ELLMAUTHALER; PÜHRER, 2014). In MCSs, each context describes different sources that can be integrated with other contexts via special rules called bridge-rules (CABALAR et al., 2019). Bordini et al. (2020) argue BDI (Beliefs, Desires, and Intentions) agents offer robust and flexible behavior, rapid and modular development, intelligibility, and verifiability. Taking these approaches into consideration, our research hypothesis is defined as follows:

BDI-agents and Multi-Context Systems can enable a modular and effective integration of connectionist methods in the agent's reasoning cycle.

We develop the model using the Sigon framework (GELAIM et al., 2019). Sigon enables the development of MCS agents, in which different contexts can be integrated with existing ones at a programming language level (GELAIM et al., 2019). We presented a reasoning cycle and performed two evaluations to demonstrate and evaluate the agent model. The reasoning cycle provides details of the agent decision-making and how it can be implemented using the Sigon framework. In the evaluations, we focus on three perspectives: (i) how the proposed agent adapts in different situations; (ii) how the agent can mitigate the necessity of hand-crafted bridge-rules in different scenarios and (iii) how the agent can enhance its decision-making by improving the neural network's performance.

1.2 GOALS

The goal of this research is to enhance intelligent systems' by integrating learning into the BDI-agent's reasoning cycle.

1.2.1 Specific goals

We define our specific goals based on intelligent agents' three key fields: agent theory, language, and architecture. The specific goals are defined as follows:

1. Specify the model of an intelligent agent following a Multi-Context System and BDI-like approach;
2. Investigate ways of integrating connectionist methods in different phases of the agent's reasoning cycle;
3. Deploy the agent in a real-world scenarios;
4. Evaluate and validate the proposed model according to the deployed scenarios.

Bordini et al. (2020) defend that AI techniques can be integrated into several parts and phases of the BDI-like agent modeling and its reasoning cycle. These techniques can be employed as an external service or be embedded into the agent's components. In the BDI cycle perspective, this integration can be crucial in sensing, planning, and acting (BORDINI et al., 2020). The proposed model in this research focuses on integrating a neural network as an embedded component in the BDI reasoning cycle, more precisely, in the sensing and planning phase. Considering these, our research does not tackle fundamental problems such as (i) balancing between the agent's reactivity and deliberation, (ii) plan failure and recovery, and (iii) beliefs' consistency over time.

Based on the Systematic Literature Mapping's findings and the properties of neural-symbolic systems, our work's originality relies on the proposed integration method. The integration method's goal is to improve the agent's decision-making by raising the level of abstraction and providing a flexible and modular integration in an intuitive way. This result is achieved by combining Multi-Context Systems, custom contexts and sensors, and Sigon as a high-level language. These resources enable the developer to employ the neural network or other Machine Learning methods in different steps of the agent's reasoning cycle. We claim this approach increases the development of modular intelligent agents capable of using different AI techniques. With these contributions, we intend to help the community while shifting the paradigm of building a programming-based model to a trained-based model (BORDINI et al., 2020).

In this research, we model, implement, deploy, and evaluate the proposed neural-symbolic agent. Based on these steps, this research's expected contributions are summarized as follows:

1. Model:
    a) integration of connectionist methods as part of the agent's decision-making;
    b) custom sensors and contexts to handle and integrate different data types in an agent based on MCS.
2. Implement:
    a) Practical implementation of the integration method;
    b) Improve Sigon framework to accommodate the proposed integration method;
    c) Use Sigon to implement the proposed agent;
    d) Employ the integration method to develop a neural-symbolic BDI-agent based on MCS.
3. Deploy the neural-symbolic agent in relevant scenarios that represent real-world characteristics.
4. Evaluation:
    a) Mitigate the necessity of hard-coded and hand-crafted rules;
    b) Enhance the agent's decision-making by improving the neural network's performance.

1.3 THESIS STRUCTURE

This document is organized as follows: Chapter 2 presents the research methodology used in this study. Chapter 3 introduces the main topics investigated in this document. Chapter 4 presents the Systematic Literature Mapping (SLM) of combining intelligent agents and neural networks. Chapter 5 presents our agent model's initial proposal and an example of its reasoning cycle. Chapter 6 presents a case study of a negotiating agent. Chapter 7 presents the evaluation of the proposed model. Chapter 8 presents additional related works of our research. Chapter 9 presents our research's main contributions, limitations, and scientific publications. Finally, in chapter 10, a conclusion and future works are presented.

2 METHODOLOGY

This work follows the Design Science Research Methodology (DSRM) (PEFFERS et al., 2007). This methodology's research development consists of building and iteratively evaluating artifacts. The DSRM process is separated into six activities: problem identification and motivation, defining the objectives for a solution, design and development, demonstration, evaluation, and communication. Section 2.1 describes the first iteration of this research. Sections 2.2 and 2.3 describe the subsequent two iterations.

2.1 ITERATION 1

Since the first iteration establishes relevant attributes employed during the development of our research, we described in more depth the details about every activity of the methodology.

2.1.1 Problem identification and motivation

Even though connectionist methods have surpassed symbolic methods, they can result in opaque and hard to interpret systems. On the other hand, it is possible to trace how decisions are made in symbolic methods. However, it requires hand-crafted rules, for instance. The effective integration of automated learning and cognitive reasoning in real-world applications is a challenging task (VALIANT, 2003; PENNING et al., 2011; BORDINI et al., 2020). In this first iteration, we performed a Systematic Literature Mapping to investigate how different studies combine connectionist and symbolic methods. We analyzed over 1000 papers retrieved from SCOPUS and ACM. Chapter 4 presents the details about the employed protocol, research questions, search string, and findings.

2.1.2 Definition of the objectives for a solution

Based on what was previously presented, the objective of our solution is to model intelligent agents' decision-making by integrating connectionist methods into their reasoning cycle. To design this solution, we propose a neural-symbolic BDI-agent based on Multi-Context Systems (MCS). In this iteration, we explore how the neural network can mitigate the necessity of defining hand-crafted rules in symbolic systems.

2.1.3 Design and development

To start integrating learning and reasoning in BDI-agents, we propose using Multi-Context Systems (MCS). We model agent's mental attitudes and resources as contexts, and integrate them via custom bridge-rules. We follow a similar approach defined in Parsons, Sierra, and Jennings (1998), Casali, Godo, and Sierra (2005) and Rodrigo Rodrigues Pires de Mello, Gelaim, and Silveira (2018). In this step, we explore the following aspects:

- How to bring neural-symbolic properties into MCS agents development;
- How the agent's sensors can process different data types;
- How to model a custom context responsible for employing a neural network during the agent's reasoning;
- Which bridge-rules are required to integrate the new custom context into the existing contexts.

In Chapter 5, we present the details about the aspects of the proposed integration method.

2.1.4 Demonstration and evaluation

In the first iteration, we explored the agent's theory perspective and its development in Sigon. We present a case study of the agent's reasoning cycle in a negotiation scenario. In this scenario, the agent assists a human during a real-life negotiation in which the main goal is to sell a product. This agent creates proposals by integrating facial expression recognition with its negotiation strategy. The data set and its description used to model the neural network can be accessed in https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge. In this first iteration presented an initial version of a theoretical model and its development in Sigon. We focused on finding the main limitations of our proposal and establishing how different neural-symbolic properties can be employed. We also showed the required changes in the Sigon framework to accommodate our proposed model. Chapter 6 presents a case study containing the most relevant steps of how we implemented the proposed agent using the Sigon framework.

2.1.5 Communication

We wrote two papers to communicate this first resulting artifact. We reported the SLM findings in Rodrigo Rodrigues Pires de Mello, Silveira, and Santiago (2021), and we presented our agent proposal and a case study about a mediator agent and facial expression recognition in Rodrigo Rodrigues Pires de Mello, Silveira, and Santiago (2022). Based on the results achieved in this first iteration, it was required to improve the neural-symbolic agent's model, deploy this model in different scenarios, and evaluate the impacts of the proposed agent and integration method.

2.2 ITERATION 2

In the previous iteration, we were able to model and implement the proposed agent. Even though we presented a case study of a mediator agent, we still needed to execute experiments to evaluate the impact of the proposed solution. In this iteration, our main goals were to improve the proposed agent and analyze the impact of adding a neural network as part of the agent's reasoning cycle. To achieve these objectives, we designed and demonstrated a reasoning cycle of a negotiating agent and executed experiments to evaluate the proposed agent.

In Rodrigo Rodrigues Pires de Mello, Silveira, and Santiago (2022), we just employed the neural network during the agent's plan's precondition verification. Therefore, it does not take advantage of using it in different parts of the reasoning cycle. As part of the design and demonstration activities, we improved the integration with the neural network context by employing the neural network's output as part of a bridge-rule's body. We also presented a new case study about a negotiating agent, which enabled us to investigate some aspects of modularity and flexibility.

In these experiments, we focused on exploring three aspects: (i) how the agent adapts in different situations, (ii) how a neural network can mitigate the necessity of hand-crafted rules, and (iii) how to compare it with different works. The scenarios used in these experiments are presented in Chapter 6. This decision enabled us to compare our work with the negotiation agents available in the GENIUS framework (General Environment for Negotiation with Intelligent multipurpose Usage Simulation) (LIN et al., 2014). GENIUS is the official tool used in the Automated Negotiating Agents Competition (ANAC), which helps the research community benchmark and evaluate its work (JONKER et al., 2017). With this scenario, we were able to compare our agent with the agent based on Multi-Context System and Adaption Aspiration Theory (AAT) that we proposed in Rodrigo Rodrigues Pires de Mello, Gelaim, and Silveira (2018).

We divided the experiments activity into two parts: (i) a comparison with 136 negotiating agents available in the GENIUS framework; and (ii) a comparison with the theoretical version presented in our previous work Rodrigo Rodrigues Pires de Mello, Gelaim, and Silveira (2018). In the first part of these experiments, we focused on evaluating the utility functions achieved in two situations. The main goal was to analyze how the agent adapted when different conditions were presented. In the second experiment, we compared three versions of the negotiating agent. Our results in the first experiment showed that the negotiating agent adapted to different situations and achieved the maximum utility value in both cases, matching the best agents available in GENIUS. The results of the second experiment showed that the second version performed better (lower time to solve the conflict) and provided the same decision as versions 1 and 3. The last experiment also showed that executing the neural network in every reasoning cycle could not be adequate in scenarios that require rapid responses.

The achieved results enabled us to evaluate and find the limitation of the resulting artifact in the current DSRM cycle. In this sense, our work's limitation was that the scenario used during the case study and experiments did not represented a complex environment. For instance, in both experiments, the negotiation agent was able to solve the conflict without the necessity of learning or adapting during the conflict resolution or after a certain amount of encounters. Based on these results, we communicate our findings from this iteration in Mello et al. (2024).

2.3 ITERATION 3

We executed one final iteration in order to mitigate the limitations from the previous artifacts and improve the neural-symbolic agent. In the previous evaluation, we analyzed the impact of employing a neural network's outputs in the agent's decision-making. We focused on finding whether a neural network could mitigate some of the symbolic method's limitations. In this iteration, our motivations and goals were the following:

1. Analyze whether the integration method can improve the neural network's performance;
2. Deploy the proposed agent in more complex and dynamic scenarios;
3. Execute a Systematic Literature Mapping, focusing in neural-symbolic agents.

We designed and executed a new experiment in a complex scenario close to the real-world. These activities enabled us to model and deploy the agent in a scenario which the main goal is to detect malicious software. A dataset called EMBER was employed during this experiment. EMBER includes real applications from 2018 (ANDERSON; ROTH, 2018). We added in this experiment features that has Continual Learning (CL) characteristics. In this scenario, new data arrives at the end of each month, thus it is required to handle new information and its impact in the agent's decision-making. We employed techniques related to transfer learning to assist during the training. We modeled a neural-symbolic agent that dynamically defines which transfer learning technique to use based on the model's accuracy. We compared the accuracy in the test dataset when using the proposed agent with two version that only uses one of the strategies in the whole dataset. When compared to the baseline version of this experiment, which achieved accuracy of 84.6%, the neural-symbolic agent achieved 84.4% and was 48% faster during the training phase. See Sections 6.2 and 7.2 for details.

We also executed a new Systematic Literature Mapping (SLM), which enabled us to analyze how different works models, design, evaluate, and deploy neural-symbolic agents. We followed the same methodology employed in the SLM executed in Section 4.1. We analyzed 58 papers and the main findings from this SLM were the following:

(i) 59% of the retrieved works follow an approach similar to ours, in which they use the NN's output as part of the agent's decision-making or uses the agent's decision-making to improve the neural network's performance; (ii) 88,2% of the works focused in planning, natural language processing, and reinforcement learning as the field of study; and (iii) 88% compare the implemented neural-symbolic agent with different works or with different versions implemented on its own research. See Chapter 8 for details.

In the communication activity, we present this experiment's results, the execution of SLM and its findings. We also intend to write a paper reporting our latest results. Figures 1 and 2 summarizes the main results of each activity from every iteration of this research.

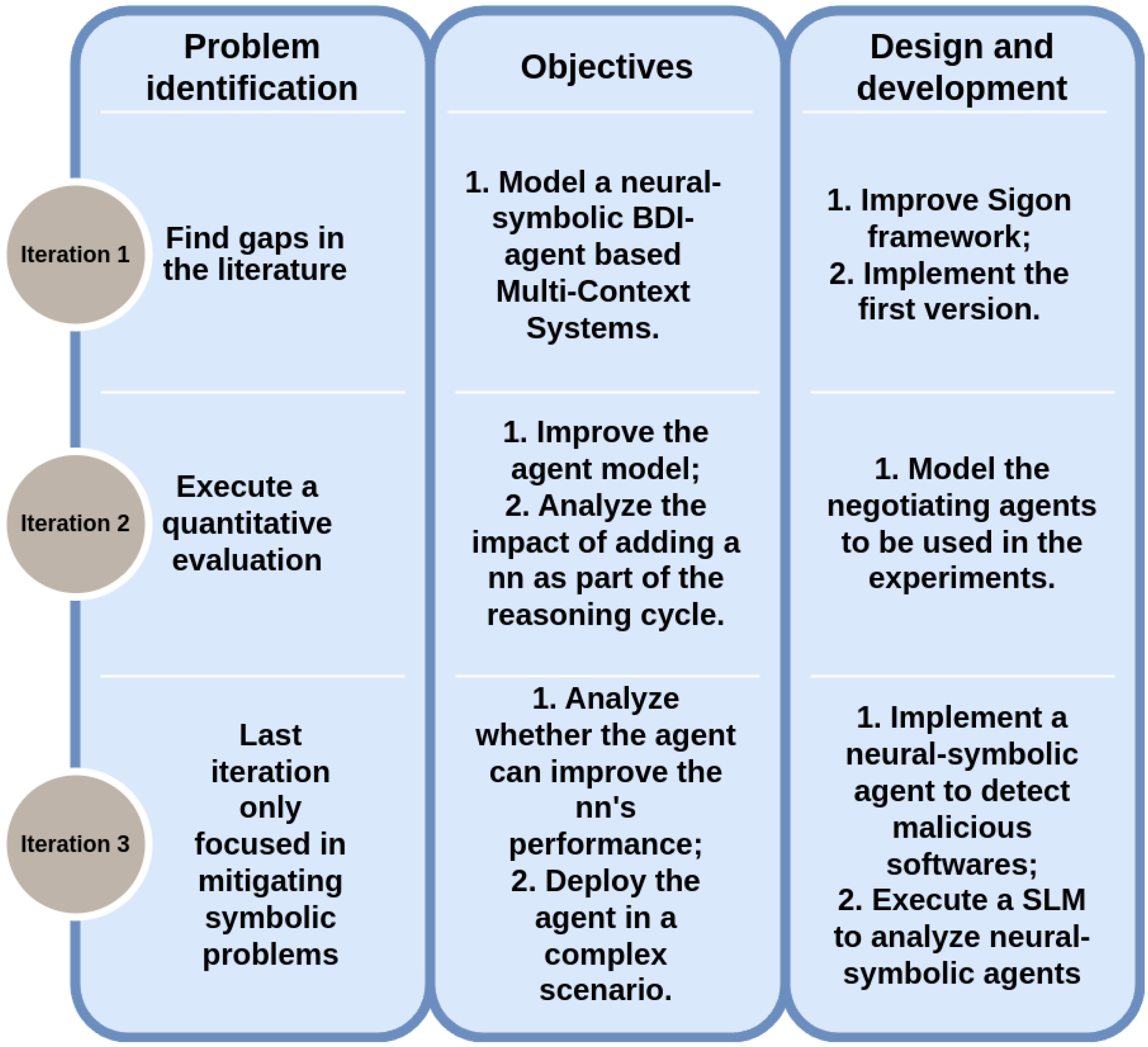
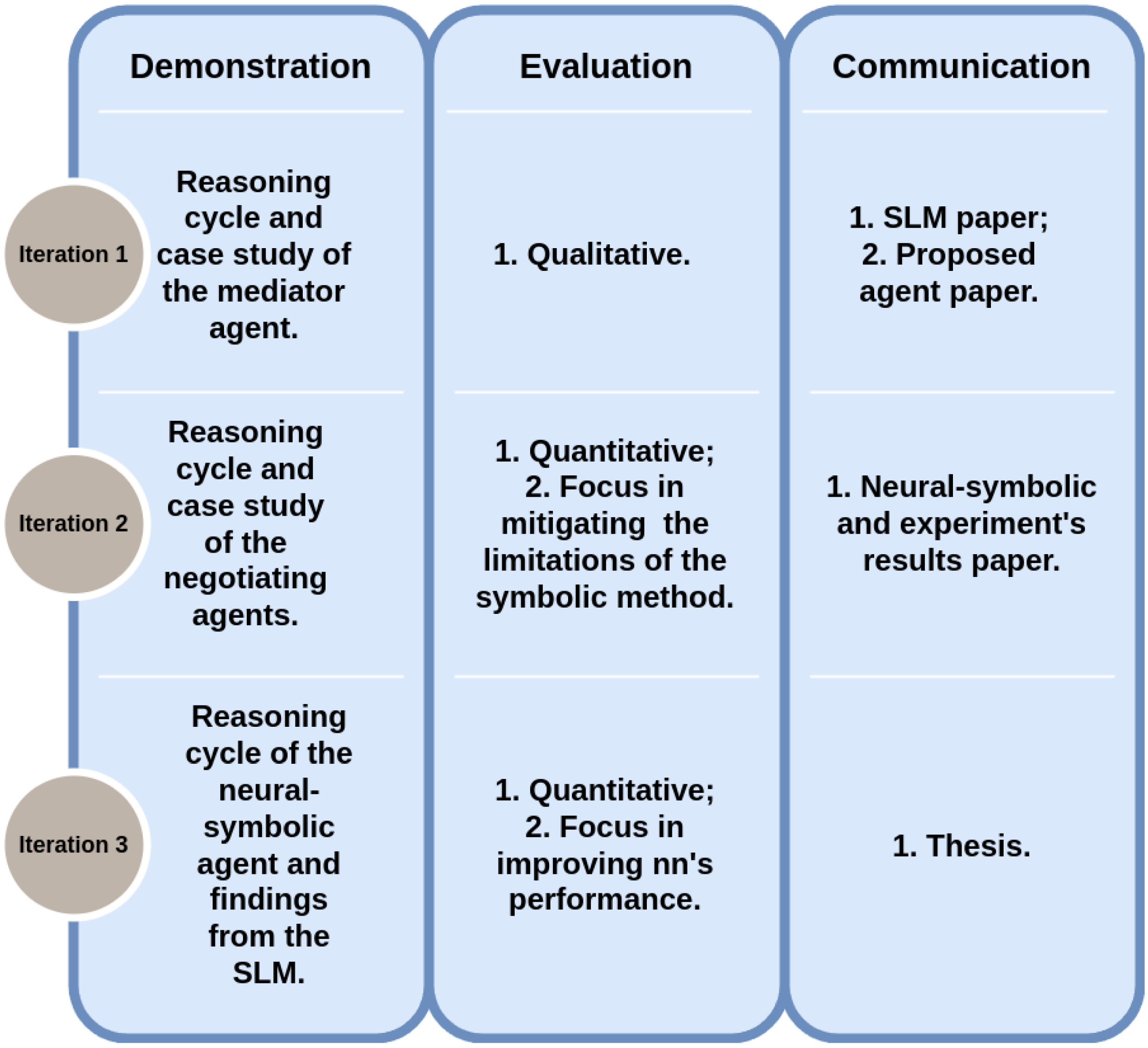

Problem identification
Objectives
Design and development
Iteration 1
Find gaps in the literature
1. Model a neural-symbolic BDI-agent based Multi-Context Systems.
1. Improve Sigon framework;
2. Implement the first version.
Iteration 2
Execute a quantitative evaluation
1. Improve the agent model;
2. Analyze the impact of adding a nn as part of the reasoning cycle.
1. Model the negotiating agents to be used in the experiments.
Iteration 3
Last iteration only focused in mitigating symbolic problems
1. Analyze whether the agent can improve the nn's performance;
2. Deploy the agent in a complex scenario.
1. Implement a neural-symbolic agent to detect malicious softwares;
2. Execute a SLM to analyze neural-symbolic agents

Figure 1 – Main remarks from every iteration of the problems, objectives, and design activities.



Demonstration
Evaluation
Communication
Iteration 1
Reasoning cycle and case study of the mediator agent.
1. Qualitative.
1. SLM paper; 2. Proposed agent paper.
Iteration 2
Reasoning cycle and case study of the negotiating agents.
1. Quantitative; 2. Focus in mitigating the limitations of the symbolic method.
1. Neural-symbolic and experiment's results paper.
Iteration 3
Reasoning cycle of the neural-symbolic agent and findings from the SLM.
1. Quantitative; 2. Focus in improving nn's performance.
1. Thesis.

Figure 2 – Main remarks from every iteration of the demonstrate, evaluate, and communicate activities.

3 BACKGROUND

This chapter introduces the background topics of our proposal. Section 3.1 presents the main concepts about symbolic and connectionist methods and their main limitations. Section 3.2 introduces the concept of neural networks. Section 3.3 presents the definition of intelligent agents. In Section 3.4, the concepts of BDI-agents as Multi-Context Systems are presented. Section 3.5 presents how these agents are modeled in Sigon. Section 3.6 shows the concepts of neural-symbolic systems.

3.1 SYMBOLIC AND CONNECTIONIST METHODS

Symbolic AI works by carrying a sequence of logic-like reasoning steps over a set of symbols consisting of language-like representations (GARNELO; ARULKUMARAN; SHANAHAN, 2016). Symbols can model actions, states, and objects. These symbols can be related via production rules, establishing relationships between symbols. Usually, these systems require hand-crafted and hard-coded rules that can be prone to errors. This characteristic leads to the Knowledge Acquisition Bottleneck, which refers to the cost of humans converting real-world problems into symbolic systems (ILKOU; KOUTRAKI, 2020). Symbolic methods are monotonic; the more rules added, the more knowledge is encoded in the system, but additional rules cannot undo old knowledge. These disadvantages could make systems less efficient and hard to maintain the rules (ARRIETA et al., 2019; ANJOMSHOAE et al., 2019).

The connectionist paradigm aims at massively parallel models that consist of many simple and uniform processing elements interconnected with extensive links, that is, artificial neural networks and their various generalizations (SUN, R., 1999). Connectionist methods helped AI achieve significant results in different fields, such as image recognition, classification, and visual computation. However, most criticism around connectionism concerns data inefficiency, poor generalization, highly dependent on the training data they process, and lack of interpretability (GARNELO; SHANAHAN, 2019a; CHOLLET, Francois et al., 2018).

Symbolic systems and neural systems diverge in terms of their data representations and problem-solving approaches. Symbolic systems rely on discrete symbolic representations and traditional search algorithms to discover solutions, while neural systems employ continuous feature vector representations and neural cells to learn mapping functions. Consequently, a significant challenge lies in designing a unified framework that seamlessly integrates both symbolic and neural components (YU et al., 2021).

3.2 NEURAL NETWORKS

Neural networks are models inspired by the structure of the brain (OZAKI, 2020; MCCULLOCH; PITTS, 1990), which provides a mechanism for learning, memorization, and generalization. A neural network is defined as a collection of units connected; the properties of the network are determined by its topology and the properties of the neurons (RUSSELL; NORVIG, 2002). A neuron in the network is able to receive input signals, to process them and to send an output signal (SVOZIL; KVASNICKA; POSPICHAL, 1997). The neural is activated when a linear combination of its inputs exceeds some (hard or soft) threshold (RUSSELL; NORVIG, 2002).

An artificial neural network consists of different neuron layers: input layers, one or more hidden layers, and an output layer (WANG, 2003). A neuron computes a weighted sum of its input signals, applies an activation function to the result, and produces an output signal. The weight coefficient reflects the degree of importance of the given connection in the neural network (SVOZIL; KVASNICKA; POSPICHAL, 1997). These models can differ not only by their weights and activation function but also in their structures, such as the feed-forward NN known for being acyclic, while recurrent NN has cycles (OZAKI, 2020). Definition 1 is presented in Kriesel (2007) and models a simple neural network.

Definition 1 *An NN is a sorted triple (N, V, w) with two sets N, V and a function w, where N is the set of neurons and V a set $\{(i,j)|i,j \in \mathbb{N}\}$ whose elements are called connections between neuron i and neuron j. The function $w : V \to \mathbb{R}$ defines the weights, where $w((i,j))$, the weight of the connection between neuron i and neuron j, is shortened to w_{ij}.*

There exist two main types of training: supervised and unsupervised training. Supervised training means that the desired output is known, and adjusting the neural network's weight coefficients is done so that the calculated and desired outputs are as close as possible. Unsupervised training means that the desired output is not known; the system is provided with a group of facts (patterns) and then left to itself to settle down (or not) to a stable state in some number of iterations (SVOZIL; KVASNICKA; POSPICHAL, 1997).

Training a particular NN architecture involves applying ordinated steps to adjust the weights and thresholds of its neurons. Each iteration of the training is called epoch. In each epoch, such an adjustment process, also known as a learning algorithm, aims to tune the network so that its outputs are close to the desired values (WILAMOWSKI, 2009). As the iterative process of incremental adjustment continues, the weights gradually converge to the locally optimal set of values (SVOZIL; KVASNICKA; POSPICHAL, 1997). Backpropagation is one of the most known algorithm responsible to execute this process. More details about this algorithm are presented in Wilamowski (2009), Svozil, Kvasnicka, and Pospichal (1997) and HECHT-NIELSEN (1992).

This work employed two neural networks: MultiLayer Perceptron (MLP) and Convolutional Neural Network (CNN). MLPs are composed of one or more hidden neural layers. They are employed in solving diverse problems related to function approximation, pattern classification, system identification, process control, optimization, robotics, and so on (WILAMOWSKI, 2009). CNNs are inspired by the natural visual perception mechanism of living creatures (GU et al., 2018). CNN made impressive achievements in many areas, including but not limited to computer vision and natural language processing (LI et al., 2022). Traditionally, a CNN consists of three layers: convolutional, pooling, and fully-connected. Following, we show the definitions presented in Gu et al. (2018):

- The convolutional layer aims to learn feature representations of the inputs. A convolution layer is composed of several convolution kernels which are used to compute different feature maps. The new feature map can be obtained by first convolving the input with a learned kernel and then applying an element-wise nonlinear activation function on the convolved results;
- The pooling layer aims to achieve shift-invariance by reducing the resolution of the feature maps. It is usually placed between two convolutional layers. Each feature map of a pooling layer is connected to its corresponding feature map of the preceding convolutional layer;
- The fully-connected layers aim to perform high-level reasoning. They take all neurons in the previous layer and connect them to every single neuron of the current layer to generate global semantic information. The last layer of CNNs is an output layer.

Figure 3 presents an example of a MLP. Figure 4 presents the LeNet-5, one of the most traditional CNNs to detect digital digits. The learning algorithm used during the training of an MLP and CNN is called backpropagation. More details about different neural network architectures and learning algorithms can be found in the works presented in Wilamowski (2009), Gu et al. (2018), Li et al. (2022) and Svozil, Kvasnicka, and Pospichal (1997).

3.3 INTELLIGENT AGENTS

Despite the existence of different definitions of intelligent agents, we assume that an agent has certain properties. Wooldridge, Jennings, et al. (1995) define the following properties:

- autonomy: agents operate without direct human intervention and can control their actions and internal state;

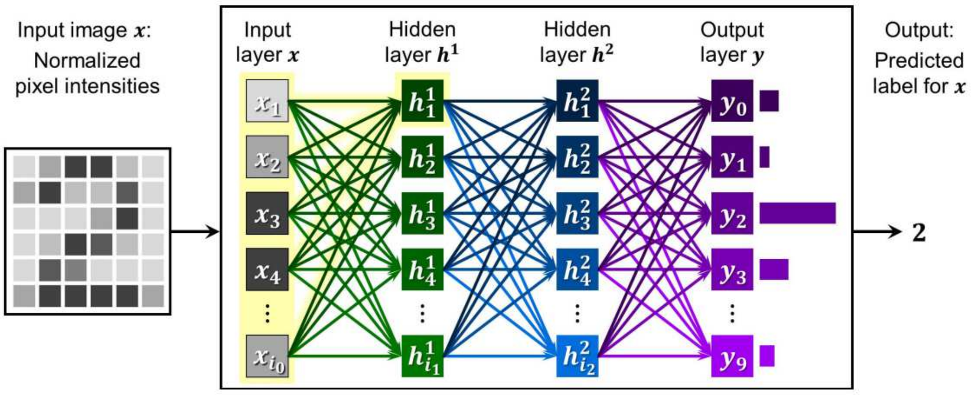

Input image x: Normalized pixel intensities
Input layer x
Hidden layer h^1
Hidden layer h^2
Output layer y
Output: Predicted label for x
x_1 x_2 x_3 x_4 x_{i_0}
h_1^1 h_2^1 h_3^1 h_4^1 $h_{i_1}^1$
h_1^2 h_2^2 h_3^2 h_4^2 $h_{i_2}^2$
y_0 y_1 y_2 y_3 y_9
2

Figure 3 – A MLP neural network for digit classification. Extracted from (CHENG et al., 2020).

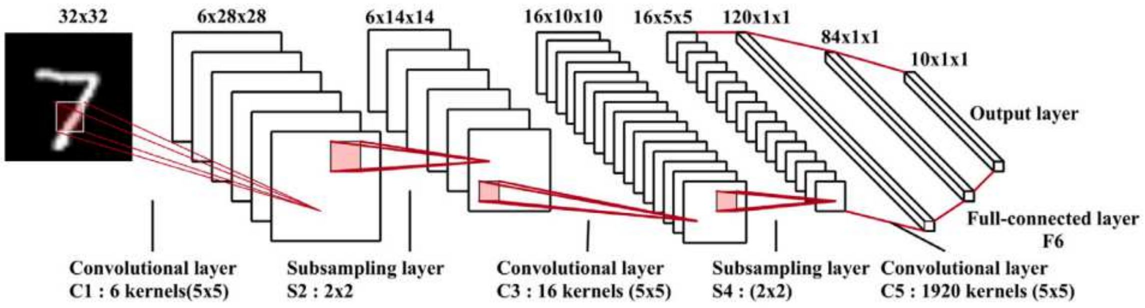

32x32
6x28x28
6x14x14
16x10x10
16x5x5
120x1x1
84x1x1
10x1x1
Output layer
Full-connected layer F6
Convolutional layer C1 : 6 kernels(5x5)
Subsampling layer S2 : 2x2
Convolutional layer C3 : 16 kernels (5x5)
Subsampling layer S4 : (2x2)
Convolutional layer C5 : 1920 kernels (5x5)

Figure 4 – The architecture of the LeNet-5 network. Extracted from (GU et al., 2018).

- social skills: agents interact with different agents and possibly humans. The language defines the communication;
- reactive: agents interact with the environment (real world, graphic interface, and a group of agents);
- proactive: agents act oriented toward goals, considering the possibility of occurrence of a specific world state.

The agent's behavior and properties can be determined by modeling its mental attitudes. In the *Belief-Desire-Intention* (BDI) architecture proposed by Bratman (1987), the three mental attitudes represent, respectively, the information, motivational, and deliberative states of the agents (RAO; GEORGEFF, et al., 1995). The behavior of an agent is specified in terms of beliefs, goals, and plans. Beliefs represent the agent's information about itself, the environment, and other agents. Goals represent a desired course of action or state of the environment the agent is trying to bring about. Plans are the means by which the agent can achieve its goals. Plans are typically predefined by the agent developer and consist of primitive actions that directly change the state of the environment and subgoals, which are, in turn, achieved by subplans. At run-time, an interpreter updates the agent's beliefs and goals in response to messages and sensory information from the agent's environment (percepts) and manages the agent's intentions. An intention is a future course of action the agent is committed to carrying out. In practice, an intention is often implemented as a stack of partially instantiated plans, the execution of which is expected to achieve a (top-level) goal or respond to the change in the agent's beliefs (typically reflecting perceived changes in the environment or new information communicated by other agents). The interpreter is also responsible for choosing which intention to execute and executing steps in the plan forming the top of the intention (BORDINI et al., 2020).

3.4 BDI-AGENT AS MULTI-CONTEXT SYSTEMS (MCS)

Interlinking knowledge sources to enable information exchange is a basic means to build enriched knowledge-based systems, which gain importance with the spread of the Internet (EITER et al., 2014). MCS is a suitable framework to link these knowledge sources. MCS describes the information available in a number of contexts (i.e., to a number of people/agents/databases/modules, etc.) and specifies the information flow between those contexts. The contexts themselves may be heterogeneous in the sense that they can use different logical languages and different inference systems, and no notion of global consistency is required. The information flow is modeled via so-called bridge rules, which can refer in their premises to information from other contexts (BREWKA; EITER, 2007).

Casali, Godo, and Sierra (2005) show that the MCS specification of an agent contains three essential components: units or contexts, logics, and bridge rules. Thus, an agent is defined as a group of inter-connected units: $\langle \{C_i\}_{i \in I}, \Delta_{br} \rangle$, in which a context $C_i \in \{C_i\}_{i \in I}$ is a tuple $C_i = \langle L_i, A_i, \Delta_i \rangle$, where L_i, A_i, and Δ_i are the language, axioms, and inference rules respectively. Δ_{br} is the set of bridge-rules. A bridge rule can be understood as rules of inference with premises and conclusions in different contexts, for instance:

$$\frac{C_1 : \psi, C_2 : \varphi}{C_3 : \theta}$$

means that if formula ψ is deduced in context C_1 and formula φ is deduced in context C_2 then formula θ is added to context C_3 (CASALI; GODO; SIERRA, 2005).

Based on these models, Parsons, Sierra, and Jennings (1998) describe how a BDI-agent can be developed as MCS. Four contexts represent beliefs, desires, intentions, and communication with the environment, in which the information flows between these contexts are defined via bridge-rules. In section 3.5, we present how a BDI-agent can be modeled using the Sigon framework. Following, we present an example of a bridge-rule with the Communication Context (CC) and Beliefs Context (BC):

$$\frac{CC : yearsOfExperience(Lang, Y > 3)}{BC : experience(Lang, senior).} \tag{1}$$

If yearsOfExperience(Language, Years > 3) is valid or can be deduced in CC, then the experience(Language, senior) will be added to BC. The integration between CC and BC is modular, in which the data type or source of information in CC should not affect how the BC is modeled.

3.5 SIGON: A FRAMEWORK FOR AGENTS' DEVELOPMENT

In this work, we develop the model using the Sigon framework (GELAIM et al., 2019). To our knowledge, Sigon is the first programming language for developing agents based on MCS. Sigon framework enables the development of agents' components as contexts and defines its integration via bridge-rules (GELAIM et al., 2019). The definitions of a Sigon agent are presented in 2.

Definition 2 (Sigon BDI-agent)

$$AG = \langle \{BC, DC, IC, PC, CC\}, \Delta_{br} \rangle, \tag{2}$$

where BC, DC, IC, PC, CC are the beliefs, desires, intentions, planning, and communication contexts; and Δ_{br} are the bridge rules for information exchange between contexts presented in definitions 5, 6, and 7.

Following the previously presented definition, the beliefs, desires, and intentions are modeled as a logical context. The communication and planning contexts are modeled as functional contexts. The communication context consists of a set of sensors and actuators. Sigon's plans and actions are based on Casali, Godo, and Sierra (2005) work. An action is defined as:

$$action(\alpha, Pre, Post, c_a) \tag{3}$$

where α is the name of the action, Pre is the set of preconditions for α execution, $Post$ is the set of post-conditions, and c_a is the α cost (GELAIM et al., 2019). A plan is defined as:

$$plan(\phi, \beta, Pre, Post, c_a) \tag{4}$$

where φ is what the agent wants to achieve, β is the action or the set of actions the agent must execute to achieve φ, Pre is the set of preconditions, $Post$ is the set of post-conditions, and c_a is the cost (GELAIM et al., 2019). Bridge-rules Δ_{br} are defined as follows:

$$\frac{CC\colon\ sense(\varphi)}{BC\colon\ \varphi} \tag{5}$$

$$\frac{DC\colon\ \varphi\ and\ BC\colon\ not\ \varphi\ and\ IC\colon\ not\ \varphi}{IC\colon\ \varphi} \tag{6}$$

$$\frac{\begin{array}{c} p = \ plan(\varphi, \beta, Pre, Pos, c_a) \\ PC\colon\ plan(\phi, \beta, Pre, Pos, c_a)\ and\ IC\colon\ \phi\ and\ \ BC\colon\ Pre \end{array}}{CC\colon\ \beta} \tag{7}$$

Sigon framework provides a BDI algorithm that can be used during the agent's development. Initially, an agent perceives data from the environment and executes the bridge-rule presented in definition 5. This first bridge-rule adds the perception captured by the Communication Context (CC) sensors to the Beliefs Context (BC). According to definition 6, the second bridge-rule is responsible for choosing an intention the agent wants to achieve. An intention is added when the agent desires it, does not believe it, and does not have it as an intention. The third bridge-rule presented in 7 selects an action to be executed. An action β is determined when the plan's precondition *Pre* is satisfied in the Beliefs Context (BC), and *phi* is true or can be inferred in the Intentions Context (IC) (GELAIM et al., 2019).

In Code 3.1, we present the main concepts about modeling the agent's contexts and bridge-rules. In Sigon, an agent's design can be divided into (i) contexts, sensors, actuators, and bridge-rules modeling; and (ii) implementation in Python of the contexts, sensors, and actuators. The agent's sensor and actuators are declared inside the Communication Context (CC) in lines 2 and 3. In Sigon, an underscore ('_') specifies a custom context name. In line 8, we present an example of how a custom context can be declared. An example of a bridge-rule declaration is given in line 10. A bridge rule starts with an exclamation ('!') symbol. The symbol ':-' separates the bridge-rule head and body. The head is on the left side of this symbol, which contains the context that will add X if the body on the right side is true or can be deduced.

In Sigon, the agent's contexts, actuators, sensors, and bridge-rules are defined in a file with a '.on' extension. An agent's modeling is processed by two main modules: the parser module, which handles the transformation of agents' source code defined in the '.on' file into an executable object, and the agent module, which links the definition of the agent's sensors, actuators, context, and bridge-rules with its Python's implementation (GELAIM et al., 2019). The Sigon parser module is based on the agent language definition and performs lexical and syntactic validations of the specified source code (GELAIM et al., 2019). For more details about Sigon implementation, we encourage the reader to access (GELAIM et al., 2019).

```
communication:
    sensor("textSensor", "integration.TextSensor").
    actuator("sendMessage", "actuator.SendMessage").

beliefs :
    somethingTrue.

_customContext:

! beliefs X :- communication textSensor(X).
```

Code 3.1 – Sigon syntax for declaring contexts and bridge-rules

3.6 NEURAL-SYMBOLIC SYSTEMS

Effective techniques such as deep learning usually require large amounts of data to exhibit statistical regularities. However, in many cases where collecting data is difficult a small dataset would make complex models more prone to overfitting. When prior knowledge is provided, e.g. from domain experts, a neural-symbolic system can offer the advantage of generality by combining logical rules/formulas with data during learning, while at the same time using the data to fine-tune the knowledge (GARCEZ, Artur d'Avila et al., 2019). The goals of neural-symbolic computation are to provide a coherent, unifying view for logic and connectionism, to contribute to the modeling and understanding of cognition and, thereby, behavior, and to produce better computational tools for integrated machine learning and reasoning (GARCEZ, Artur d'Avila et al., 2022).

Figure 5 presents a diagram of how these methods can be integrated. The green rectangle represents symbolic systems, which employ reasoning-based approaches to find solutions. Symbolic systems typically operate on structured data, such as logic rules, knowledge graphs, or time series data. Their fundamental unit of information processing is symbols. Through training, symbolic systems acquire the solution space of a search algorithm for a specific task and output higher-level reasoning results. On the other hand, the blue rectangle in the figure represents neural systems, which excel at learning-based approaches to approximate the ground truth. Neural systems usually operate on unstructured data, such as images, videos, or texts, and their primary information processing unit is a vector. Through training, neural systems learn a mapping function for a specific task and output lower-level learning results. The outer blue box represents neural-symbolic learning systems, which encompass the characteristics of both symbolic and neural systems. These systems combine the reasoning capabilities of symbolic systems with the learning capabilities of neural systems to achieve a comprehensive and integrated approach to problem-solving (YU et al., 2021).

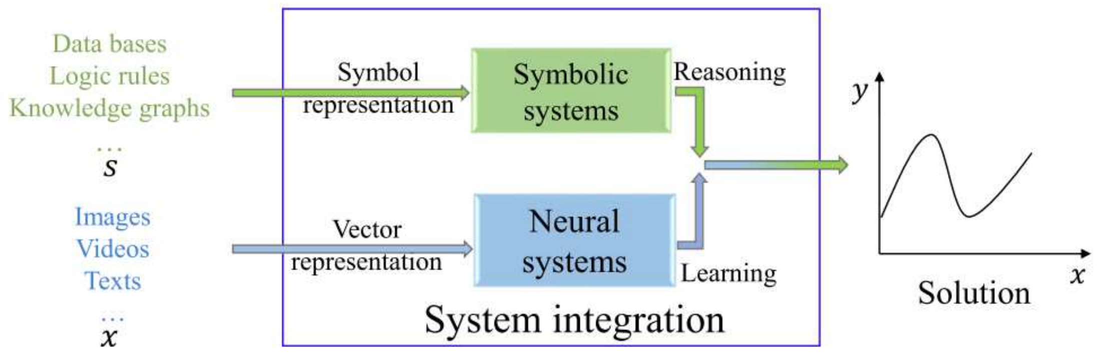


Figure 5 – Schematic of diagram of neural-symbolic integration. Extracted from Yu et al. (2021).

Our work employed the proposed neural-symbolic taxonomy presented in Yu et al. (2021). The taxonomy is determined by the combination of neural systems and symbolic systems, which has three primary combination methodologies:

1. learning for reasoning: this category's primary goal is to search for solutions using symbolic systems (symbolic reasoning techniques) and integrates the benefits of the neural networks to assist in finding solutions;
2. reasoning for learning: this category uses neural systems' learning capabilities to map functions and integrates the advantages of symbolic systems (symbolic knowledge) into the learning process to enhance the learning ability of neural systems;
3. learning-reasoning: both neural and symbolic systems play equal roles and work together in a mutually beneficial way.

In the context of learning for reasoning, there are two main aspects to consider. The first one involves the use of neural networks to reduce the search space of symbolic systems, thereby accelerating computation. This can be achieved by replacing traditional symbolic reasoning algorithms with neural networks. The neural network effectively reduces the search space, making the computation more efficient. The second aspect of learning for reasoning is the abstraction or extraction of symbols from data using neural networks to facilitate symbolic reasoning. In this sense, neural networks serve as a means of acquiring knowledge for symbolic reasoning tasks. They learn to extract meaningful symbols from input data and use them for subsequent reasoning processes (YU et al., 2021; GARCEZ; GORI, et al., 2019).

The underlying idea in reasoning for learning is to leverage neural systems for machine learning tasks while incorporating symbolic knowledge into the training process to enhance performance and interpretability. Symbolic knowledge is typically encoded in a format suitable for neural networks and used to guide or constrain the learning process. For instance, symbolic knowledge may be represented as a regularization term in the loss function of a specific task. This integration of symbolic knowledge helps improve the learning process and can lead to better generalization and interpretability of the neural models (YU et al., 2021).

In the learning-reasoning approach, the output of the neural network becomes an input to the symbolic reasoning component, and the output of the symbolic reasoning becomes an input to the neural network. By allowing the neural systems and symbolic systems to exchange information and influence each other iteratively, this approach aims to leverage the strengths of both paradigms and enhance the overall problem-solving capability. In this case, the neural network component generates hypotheses or predictions, which are then used by the symbolic reasoning component to perform logical reasoning or inference. The results from symbolic reasoning can subsequently be fed back to the neural network to refine and improve the predictions (YU et al., 2021).

The concepts presented in this Chapter are crucial in establishing how we intend to answer this work's research question. In order to enhance intelligent systems, our goal is to integrate learning as part of the agent's reasoning cycle. The BDI-agent represents the symbolic method, and we employ neural networks to model the connectionist method. To model and integrate both methods, we employ Multi-Context Systems. We claim that BDI-agent and Multi-Context Systems provide the required modularity and flexibility to implement this integration. Modularity is achieved by representing the agent's mental attitudes (beliefs, desires, and intentions) and learning capabilities (neural networks) with contexts. The flexibility is achieved by integrating these contexts through bridge-rules. We employ Sigon to implement the proposed integration method. We believe that combining these approaches can result in a neural-symbolic agent.

4 SYSTEMATIC LITERATURE MAPPING

It is claimed that the next step of AI is the integration of connectionist and symbolic methods (BORDINI et al., 2020). Combining symbolic and connectionist suggests a promising strategy to develop robust applications to assist us in our daily tasks (ARRIETA et al., 2019; ADADI; BERRADA, 2018). The main goal of this integration is to increase intelligent systems' expressiveness, trust, and efficiency (BENNETOT et al., n.d.; GARNELO; ARULKUMARAN; SHANAHAN, 2016; MARRA et al., 2019; GARCEZ, Artur d'Avila et al., 2019). According to what was previously presented, we are interested in studying neural networks and intelligent agent approaches as the connectionist and symbolic methods, respectively. In this sense, the main goal of this systematic literature mapping is to explore how neural networks are combined with the agent's decision-making.

Before our study, we found that different parts of Jedrzejowicz (2011), Garnelo and Shanahan (2019b) and Rizk, Awad, and Tunstel (2018) works are similar to ours, although most of these works are surveys and do not present a systematic review with a well-defined protocol. Garnelo and Shanahan (2019a) present compelling arguments about integrating symbolic and deep neural networks. However, Garnelo and Shanahan (2019a) does not show a systematic literature review, and its work focuses on object representation and compositionality and how they can be accommodated in a deep learning framework. Rizk, Awad, and Tunstel (2018) present a survey about how reinforcement learning, dynamic programming, evolutionary computing, and neural networks can be used to design algorithms for Multi-Agent systems decision-making. Jedrzejowicz (2011) also explores the integration of machine learning and agents. However, we believe it is necessary to revisit the last decade of advances in AI.

This chapter is organized as follows. Section 4.1 presents the protocol used to execute this Systematic Literature Mapping. The results are shown in section 4.2. A brief road-map and how this systematic literature mapping influences this research are presented in section 4.3.

4.1 SYSTEMATIC LITERATURE MAPPING PROTOCOL

We follow Kitchenham and Charters (2007) works as a guideline to perform this Systematic Literature Mapping (SLM). An SLM differs from a Systematic Literature Review (SLR) because it presents a broader overview of a field of study, establishes the existence of research evidence, and indicates the number of evidence (KITCHENHAM; CHARTERS, 2007). According to Kitchenham and Charters (2007), a systematic literature review or mapping involves several discrete activities. Three main phases with different tasks can divide this process. The phases and tasks that we executed are the following:

- Planning: identification of the need for a review, specifying the research question(s), developing a review protocol, evaluating the review protocol;
- Conducting: identification of research, selection of primary studies, study quality assessment, data extraction, and data synthesis;
- Reporting: formatting the main report and evaluating it.

4.1.1 SLM's research questions

We employed the five criteria Population, Intervention, Comparison, Outcomes, and Context (PICOC) described in Petticrew and Roberts (2008) to define our SLM research questions. Since our research questions explore the combination of two different approaches, it is worth mentioning that we did not use the comparison criteria of the PICOC method in our study. The main reason for this decision is that our initial research focused on investigating how agents and neural networks are integrated. The PICOC criteria and its definitions are the following:

- P (population or problem): intelligent agents and their different classes;
- I (intervention or interest): which neural networks architecture are employed;
- O (Outcome/results): main contributions achieved by the system originated by combining neural networks and intelligent agents;
- C (Context): scenarios in which the proposed approach was used.

The SLM research questions are defined as follows:

- RQ1: Which class of agents are employed?
- RQ2: Which architectures of neural networks are employed?
- RQ3: How do these works combine neural networks and agents?
- RQ4: Which scenarios are these intelligent systems being deployed?
- RQ5: Do these works' contributions focus on improving neural networks, intelligent agents, or both fields?

4.1.2 Search string

Since the primary goal of this SLM is to study and analyze the integration between connectionist methods and intelligent agents, the search strings executed in SCOPUS and ACM are the following:

- Scopus: ("deep learning" OR "neural network") AND ("intelligent agent" OR "autonomous agent");
- ACM: Title:((("deep learning" OR "neural network") AND ("intelligent agent" OR "autonomous agent"))) OR Abstract:((("deep learning" OR "neural network") AND ("intelligent agent" OR "autonomous agent"))) OR Keyword:((("deep learning" OR "neural network") AND ("intelligent agent" OR "autonomous agent")));

Since performing an initial search with different search strings is common, we noticed that some works use the term ‘deep learning’ to refer to neural networks during one of these searches. Considering that, we added this term in our final search string.

Table 1 presents the inclusion and exclusion criteria used to filter the relevant studies in our SLM. As previously mentioned, Jedrzejowicz (2011) also explores machine learning and agent integration. However, this work did not examine the last five years of advances in AI. We believe revisiting the previous five years of AI contributions is required.

Table 1 – Inclusion and exclusion criteria

Inclusion (I)	Exclusion (E)
published between 2015 to 2021	published before 2015
written in English	not written in English
available to download	unavailable to be read
combines neural network and intelligent agent to build an intelligent system	does not use intelligent agents
present a qualitative or quantitative evaluation	does not use the neural network
published in conference or journal	do not present quantitative or qualitative evaluation
primary studies	secondary or tertiary studies

4.1.3 Selection process

We perform our systematic mapping review in the following order:

1. Execute search string in selected digital libraries;
2. Apply inclusion and exclusion criteria;
3. Select papers based on titles, abstracts, and keywords;
4. Data extraction from selected papers;
5. Report main findings based on data extracted from the selected papers.

Figure 6 presents the steps performed during the selection, data extraction, and analysis. Each step contains the number of papers selected for the next step. It is important to note that even after step 4 when inclusion and exclusion criteria were applied, some papers did not fit these criteria; therefore, they were removed before the data extraction step. Some papers were not available to download, which caused a reduction in the number of papers used in the data extraction step. We considered the unavailability of these papers to be one of the validity threats. Taking that into consideration, the final number of analyzed papers was 110.

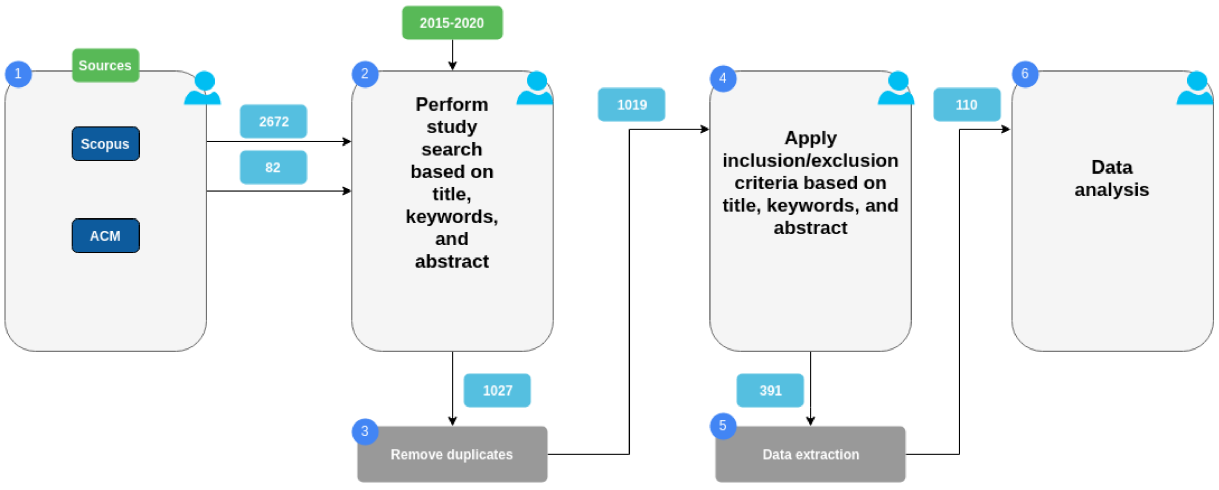

2015-2020
1
Sources
Scopus
ACM
2672
82
2
Perform study search based on title, keywords, and abstract
1019
4
Apply inclusion/exclusion criteria based on title, keywords, and abstract
110
6
Data analysis
1027
3
Remove duplicates
391
5
Data extraction

Figure 6 – Systematic literature mapping executed steps.

4.1.4 Data extraction

The fields and their definitions used during data extraction are the following:

1. RQ1 - Agents class: since an agent terminology and its architectures vary across different works and fields, we chose to use the classification presented in Russell and Norvig (2002), which defines the following agent's types:
   - simple reflex agents;
   - model-based reflex agents;
   - goal-based agents;
   - utility-based agents;
   - learning agents.
2. RQ2 - Neural networks types: this field provides information on the kind of neural network used;
3. RQ3 - Integration: the primary goal of this field is to study how different works combine neural networks and agents during AI systems development;
4. RQ4 - Contributions: this field identifies the main contribution resulting from the combination of neural network and agents;

5. RQ5 - Scenario: this field intends to report where the proposed agent was or intended to be deployed and whether there exists a concern about using these approaches to assist in real-world problem resolution.

To access the Data extraction form, the reader could access https://drive.google.com/open?id=1FGZOGqFGGd0FfVMG-ItNtC_Pg9Xov2oKrB6k-G2SBLw.

4.1.5 Validity threats

According to Figure 6, the selection and data extraction were executed by just one researcher. This decision is the one that represents more risks to our study and originates the following threats:

1. Researcher expertise: since the steps of study selection and data extraction were executed only by one researcher who has a background in intelligent agents, some of the relevant features of the neural network could be ignored or wrongfully reported;
2. Researcher bias: some of the works did not present a quantitative evaluation or explicitly define the main contributions; hence, in some of the works, the reported contribution and main findings could be limited or imprecise;
3. Data aggregation: based on what is presented in section 4.2, it was necessary to define classes of agents and the employed neural networks to answer some research questions. In this sense, the interpretation of the main findings could present imprecision and limitations;
4. Unavailable papers: we noticed that some articles published in relevant conferences and journals were not available to download in our institution, which limited our SLM results.

4.2 RESULTS FROM THE DATA ANALYSES

This section reports the most important findings we gathered during the data analysis step. The analysis method and the results employed in our work are called thematic. This method aims to describe and present an overview of existing works (DIXON-WOODS et al., 2006). The decision to use this approach is supported by the fact that this work is a systematic literature mapping and does not require a qualitative analysis.

We start by showing in subsection 4.2.1 the retrieved studies distribution. In subsections 4.2.2, 4.2.3, and 4.2.4, we discuss the first three research questions, which are related to intelligent agent's different groups, neural networks architectures, and the combination of neural networks and intelligent agents. Subsection 4.2.5 presents the main findings of the two remaining research questions.

The following subsections present the most important findings of the combination of neural networks and intelligent agents. We start by showing the retrieved studies distribution in the subsection 4.2.1.

4.2.1 Studies distribution between 2015 and 2020

Table 2 presents the distribution of studies returned after executing the search string in SCOPUS and ACM digital libraries. This figure shows the interest in intelligent agents and neural networks in the last five years. The interest in this field started increasing in 2017, with the number of works between 2018 and 2020 representing 59.22%. It is also worth mentioning that this search occurred on 05/04/2020; therefore, it does not include the year 2020 in its totality.

Year	Quantity
2015	111
2016	131
2017	176
2018	297
2019	282
05-04-2020	28

Table 2 – Studies distribution returned from 2015 to 2020.

4.2.2 RQ1 - Intelligent agents groups

Figure 7 shows the distribution of different classes of agents returned after the data extraction step. It is important to mention that one work could employ more than one class. Analyzing Figure 7 it is possible to observe that:

- Reinforcement learning agents usage. Following the results obtained by employing deep neural networks, the usage of reinforcement learning agents is noticeable. This result could be explained by using deep neural networks to define reward policies, which represented the main limitation of reinforcement learning. For instance, in Mnih, Kavukcuoglu, Silver, Graves, et al. (2013) and Mnih, Kavukcuoglu, Silver, Rusu, et al. (2015), relevant results were achieved using deep neural networks and reinforcement learning agents.

- Simple-reflex agents. As one of the most explored agents, it is still relevant to point out its usage. One of the main reasons is that it is simple to combine this type of agent with other techniques since, most of the time, the chosen technique acts as decision-making, and the agent only possesses sensors and actuators.

Even though we did not fully explore the Multi-Agent System (MAS) usage, we noticed a relevant increase in combining MAS with reinforcement learning and deep neural networks. This approach points towards a direction where agents could use different policies to coordinate their actions.

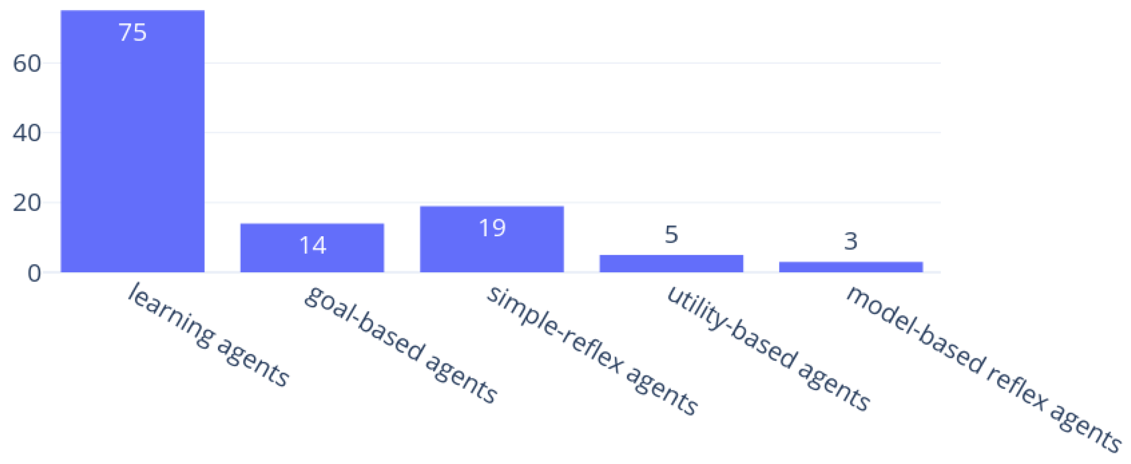


Figure 7 – Agents class distribution during the period of 2015 to 2020.

4.2.3 RQ2 - Neural networks

Following the same approach explored in Figure 7, Figure 8 presents the most used types of neural network architectures during data extraction. Some relevant information noticed during the data extraction step is that some works did not accurately report the neural network used. We removed the works that did not present information about the used neural network. Since some works use more than one type of neural network, the total amount of neural networks differs from the number of analyzed papers. Convolution, feed-forward, and recurrent neural networks were more frequent. These numbers also agree with the approach used in Mnih, Kavukcuoglu, Silver, Graves, et al. (2013) and Mnih, Kavukcuoglu, Silver, Rusu, et al. (2015), in which these types of neural networks were employed.

To answer the remaining research questions, we use a scope analysis approach. A scoping analysis represents a flexible way of providing a broader view of the selected research, which fits the primary goal of a Systematic Literature Mapping (DIXON-WOODS et al., 2006).

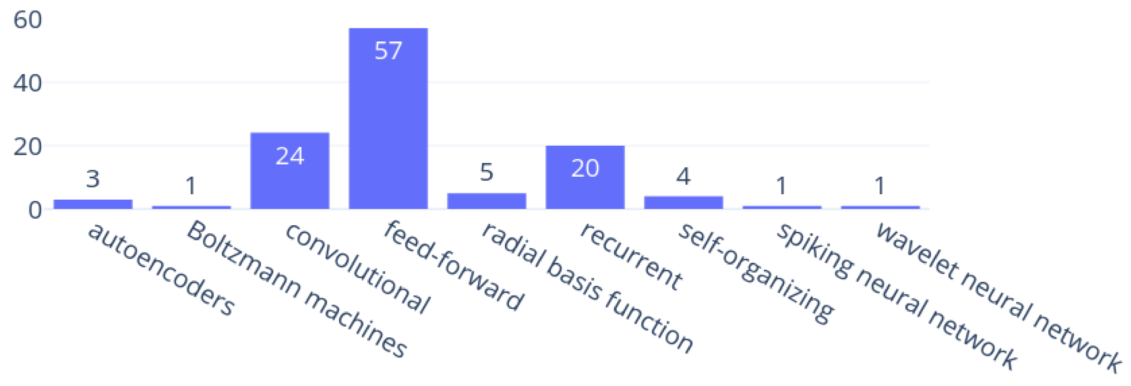


Figure 8 – Neural networks architecture distribution during the period of 2015 to 2020.

4.2.4 RQ3 - Combination of neural networks and agents

Figure 9 summarizes how studies combine different neural network architectures and intelligent agents. Two of our study's most relevant findings could be explained through Figure 9. The first one is the usage of feed-forward, convolution, and recurrent neural networks to define reward policies for learning agents. The second one is that few studies use the neural networks as input or combine them with the agent's reasoning. This decision could be linked to the difficulty of combining neural networks in these steps.

One of the main reasons for the numbers presented in Figure 9 could be explained by using neural networks to define actions or policies previously explored in the literature and do not require changing the agent's reasoning cycle during decision-making. In this sense, using the neural network as input or part of the decision-making process could require a more robust implementation of these agents.

4.2.5 RQ4 - Contributions and RQ5 - Scenarios

Even though the spam of contributions from the different works varies, it is possible to define in which group these works focus their contributions. In our study, we define the following group:

- intelligent agents contribution;
- neural networks contribution;
- intelligent agents and neural network contributions;
- area of contribution of the resulting intelligent system.

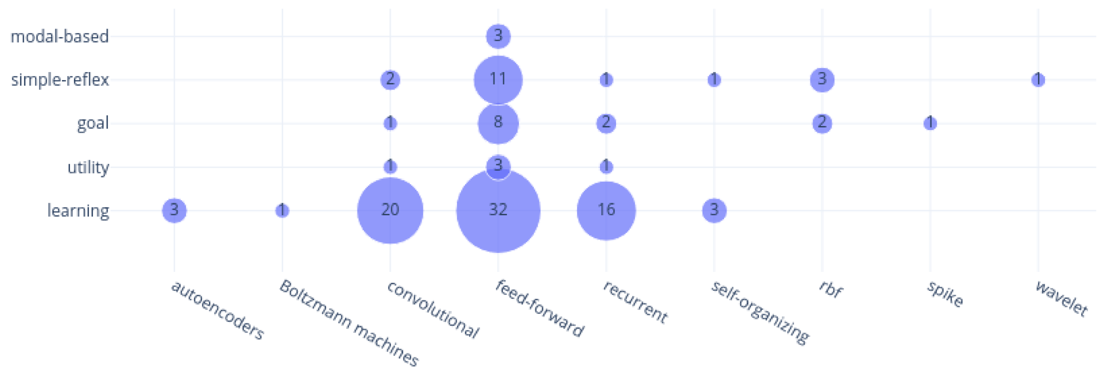


Figure 9 – Studies distribution based on the usage of different neural networks and intelligent agents.

Figure 10 presents the findings related to these contribution groups. Even though there is a concern about contributing to intelligent agents and neural network development, there is also a more frequent concern about solving a problem from a particular field of study. Figure 10 shows a lower concern about the contribution to intelligent agents and neural network fields. In this sense, the evaluations of these studies focused only on agents or neural networks. The scenarios used in the different works varied during the execution of this study. However, it is possible to define which scenario is more frequent. Many studies contributed by employing agents and neural networks to assist during problem resolution or to simulate real-world scenarios.

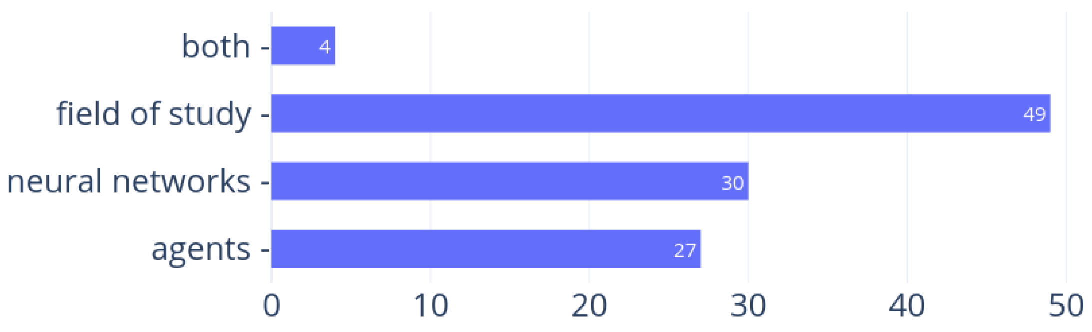


Figure 10 – Distribution of contribution between the period of 2015 to 2020.

In Wang and Tan (2015), De Paula and Gudwin (2015), Hou, Feng, and Ong (2016), Diallo, Sugiyama, and Sugawara (2017), Serafim et al. (2017), Zhu et al. (2018) and Sotnikov and Mazurenko (2020) the main goal was to build a system able to act in an electronic game, in a player's role, or as a training step. Other studies presented a simulation of a particular environmental situation or specific behaviors. For instance, in Yang (2017), it was modeled a multi-agent system that simulated a crowd behavior. In Yuksel (2018) and Sharma et al. (2018), evacuation systems were designed and evaluated. It is also essential to point out the usage of robots to explore hostile environments, as presented in Ramezani Dooraki and Lee (2018), Luo, Subagdja, et al. (2019) and Majumdar, Benavidez, and Jamshidi (2018). As presented in Lamouik, Yahyaouy, and Sabri (2017), Chen, Zhao, and Chan (2019), Loumiotis et al. (2018), Garg, Chli, and Vogiatzis (2019), Amrani et al. (2019), Klose and Mester (2019) and Kotyan, Vargas, and Venkanna (2019) different studies built systems able to assist humans during task resolution. Some studies showed an agent responsible for driving a car or controlling a traffic light signal autonomously.

4.3 DISCUSSION

In this section, we presented a Systematic Literature Mapping (SLM). The main goal was to report how different works employ neural networks and intelligent agents to solve various problems. To achieve our goal, we defined five research questions. The first two research questions are related to the type of agents and neural networks used. The third research question reports on which steps of the agent's decision-making the neural network was employed. The fourth and fifth questions present details about the main contribution of these studies and the scenarios in which these studies were deployed. The amount of 1019 papers from 2015 to 2020 shows the relevance of the field explored. The 2018, 2019, and 2020 studies were responsible for 73,76% of the works used in our systematic literature mapping, showing the field's growth after 2017.

One of the most important findings of our SLM shows that few studies explore the integration of neural networks as part of the agent's decision-making. Most studies use neural networks to define learning agents' reward policies. Even though these approaches provide significant results, these systems have suffered from a lack of transparency and require considerable data (ADADI; BERRADA, 2018; ARRIETA et al., 2019). This criticism also limits the field of study in which an AI system can be deployed, such as in health care, finance, and legal (GARNELO; SHANAHAN, 2019a). Although many studies contributed to neural networks and agent design, several studies use agents and neural networks to solve or contribute to a particular study field.

A promising path towards integrating neural networks into the agent's reasoning cycle can be achieved by considering the neural-symbolic field. Neural-symbolic provides the effective integration of connectionist and symbolic methods, more precisely learning and reasoning (PARISOTTO et al., 2016). Neural-symbolic can be employed where large amounts of heterogeneous data exist, and knowledge descriptions are required (GARCEZ; BESOLD, et al., 2015).

Besides the SLM executed in our research, Bordini et al. (2020) also provides a survey and roadmap about integrating AI techniques into BDI-agents. This survey defends the idea that this integration can raise the level of abstraction of agent programming by increasing the basic competence of agent languages and platforms. Bordini et al. (2020) discuss how AI techniques can be defined as external services or endogenous components of the agent, which could be combined in different architectural strategies that can be relevant in the sensing, planning, and acting phases of the BDI cycle. Bordini et al. (2020) seem to agree with our work in the sense that a promising way to integrate AI techniques is to use different approaches to developing the agent's capabilities in a flexible and modular way.

Based on the findings of the Systematic Literature Mapping execution and the survey in Bordini et al. (2020), we believe that our work can contribute in: (i) proposing an integration method inspired by Multi-Context Systems, in which neural networks are part of the agent's reasoning cycle; and (ii) designing a novel approach for developing modular intelligent agents with the possibility of employing different Machine Learning techniques. With these contributions, we intend to help the community while shifting the paradigm of building a programming-based model to a trained-based model (BORDINI et al., 2020).

5 MODEL OF THE NEURAL-SYMBOLIC BDI-AGENT BASED ON MULTI-CONTEXT SYSTEMS

In this chapter, we present the proposed agent model. We defend some relevant aspects of integrating AI methods into the agent's decision-making that could lie in the fact that we are employing BDI as the agent's architecture based on a Multi-Context System (MCS). BDI-like agents provide robust and flexible behavior for modeling intelligent systems deployed in dynamic environments. MCS enables the modeling of different knowledge sources and integrates them via bridge-rules. Taking that into account, a fully-fledged BDI-based framework integrating AI techniques should ideally allow the use of different approaches to developing additional capabilities of an agent in a flexible and modular way (BORDINI et al., 2020).

Combining BDI-like agents and MCS also gives rise to two key aspects of neural-symbolic: modularity and generally hierarchical organization. Besold, Garcez, et al. (2017) define these characteristics as follows:

1. modularity refers to using different networks during reasoning, where each network can be responsible for different tasks. This characteristic also takes into account the comprehensibility and maintenance of the developed AI system;
2. generally hierarchical organization main goal is to build a system where networks might be trained independently, possibly also combining unsupervised and supervised learning at different levels of the hierarchy.

In our agent, modularity can be achieved by adding custom contexts. Each context can be responsible for different parts of the agent's mental abilities. The property of the generally hierarchical organization can be addressed by defining bridge-rules. These bridge-rules help integrate learning and reasoning techniques. Its execution order describes when and how a resource of a specific context should be used during an agent's reasoning cycle. Our work focuses on building a system around property learning for reasoning. This property's primary goal is to employ the benefits of a neural network to assist a symbolic system in finding a solution. In this sense, in the proposed model, the neural network's output will be employed during the agent's reasoning cycle.

AI techniques can be applied in several parts and phases of the BDI-like agent modeling and its reasoning cycle. In our research, we take into consideration two aspects discussed in Bordini et al. (2020):

1. Modeling perspective: AI techniques can be employed as an external service or be embedded into the agent's components;
2. BDI cycle: AI techniques can be crucial in sensing, planning, and acting.

The properties previously discussed could be combined into different strategies, resulting in many intelligent systems. However, the proposed model in this research focuses on employing neural networks as an embedded component in the BDI reasoning cycle, more precisely, in the sensing and planning phase.

Since one of our specific goals is to deploy the proposed agent in a real-world scenario, it is necessary to consider that some specific scenarios require processing different data types. Considering that BDI-like agents are suitable for dynamic and time-constraint environments (CHONG; TAN; NG, 2007), the proposed agent can also process different types of perceptions during its reasoning cycle. We start modeling our proposal integration method by adding custom sensors and contexts into a BDI-agent based on MCS. Each custom sensor is responsible for handling different data and creating perception. The integration between each context is modeled by defining bridge-rules. We also define a custom context responsible for representing a connectionist method and enabling it to be used during the agent's reasoning cycle. Our agent's core architecture is presented in Definition 3.

Definition 3

$$AG = \langle\{BC, DC, IC, PC, NNC, CC, AC_1, AC_2, ..., AC_n\}, \Delta_{br}\rangle, \tag{8}$$

where BC, DC, IC, PC, NNC, CC are the beliefs, desires, intentions, planner, neural network, and communication contexts; $AC_1, AC_2, ..., AC_n$ are the auxiliary contexts; and Δ_{br} are the bridge-rules for exchanging information between contexts.

Figure 11 follows the approach given in Definition 3, in which each vertex is a context, and the edges are bridge-rules. The main difference is that the auxiliary contexts are not defined in Figure 11. These bridge-rules model the information exchange between contexts. The bridge-rules 5, 6, 7 are defined in Subsection 3.5. The bridge-rules 9 and 10 will be introduced in more detail in Subsection 5.2. These new bridge-rules are the ones that will be responsible for integrating the new Neural Network Context (NNC) into the agent's reasoning cycle.

It is worth mentioning that the agent's designer should implement the Neural Network Context (NNC). The agent's designer could add different custom contexts, such as the auxiliary contexts ($AC_1, AC_2, ..., AC_n$), and define bridge-rules to provide information exchange among the agent's contexts. Custom contexts can receive specific perceptions from the Communication Context (CC). These perceptions represent raw image, audio, video, or textual data. In a broader sense, these custom contexts are responsible for processing different perceptions and providing strategies or knowledge to use during the agent's decision-making. Section 5.1 presents how the agent handles different data types and how these data types are integrated into the MCS.

In the design and development activity of the first iteration of the employed methodology, we proposed custom contexts that receive specific perceptions from the

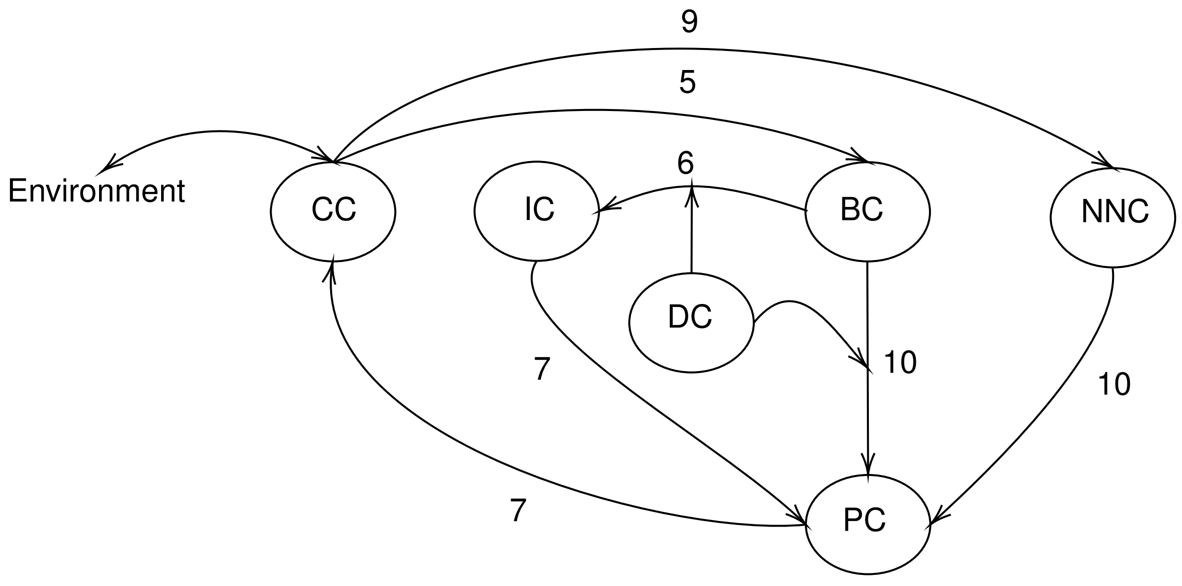


Figure 11 – Neural-symbolic BDI-agent based on Multi-Context Systems.

communication context. These perceptions can represent raw image, audio, or video data. In a broader sense, the custom contexts are responsible for processing different perceptions and providing strategies or knowledge that can be used during the agent's planning phase. The proposed custom contexts can be described as functional contexts, using either a connectionist or symbolic method.

5.1 COMMUNICATION CONTEXT'S CUSTOM SENSORS

As previously mentioned, the Communication Context (CC) is responsible for modeling the interaction between the agent and the environment (GELAIM et al., 2019). Since dynamic environments have the presence of different data types, the proposed model must handle these data in an effective and modular way. In this sense, we are concerned about providing these mechanisms without the necessity of changing the Communication Context (CC) or adding more responsibilities to the Communication Context. For instance, one agent can have two custom sensors: one for processing image data and one for processing textual data. Each sensor defines how these data will be transformed into perception, enabling the agent to use it during its reasoning cycle.

We propose adding custom sensors to implement these mechanisms, in which each sensor is responsible for processing specific data and generating a perception. This method is based on software engineering design patterns, more precisely, the decorator pattern. The decorator pattern attaches additional responsibilities to an object dynamically (KASSAB et al., 2018). A custom sensor must define the following operations to map an observation to a perception correctly:

1. How the perceived data are processed and transformed into a perception;
2. How the perception will be added into the Communication Context (CC);
3. How the Communication Context (CC) can verify whether a perception exists or is valid.

Sigon also provides a flexible approach to model actuators. The actuators are integrated with the planner context, enabling an agent to trigger different actions based on the selected plan. Subsection 5.2 presents how to add and integrate the Neural Network Context (NNC) into the agents' reasoning cycle, representing one of this work's main contributions. In Subsection 5.2.1, details about this implementation are presented. Chapter 6 shows a case study and how to model the proposed agent in the Sigon framework.

5.2 NEURAL NETWORK CONTEXT (NNC)

One of the main obstacles during symbolic systems design is defining hard-coded and hand-crafted rules, requiring human intervention and the cost of converting the real-world problem into symbolic systems (ILKOU; KOUTRAKI, 2020). An intelligent system could take advantage of the neural networks' mechanism for learning, memorization, and generalization (OZAKI, 2020; MCCULLOCH; PITTS, 1990). Since the agent's designer must define bridge-rules, we considered them hard-coded and hand-crafted rules. In our work, we mitigate this problem by employing the information provided by a neural network's output, modeled in the Neural Network Context (NNC). The information provided by the NN's output is utilized as part of the bridge-rule's body.

After defining how each sensor can map an observation to perception, it is necessary to integrate these perceptions with the agent's reasoning cycle. The bridge-rule presented in Definition 9 models how to achieve this integration. Bridge-rule shown in Definition 9 is similar to the one presented in Section 3 that adds perceptions into the Beliefs Context (BC).

$$\frac{CC : sensor_i(\beta)}{NNC : \beta} \tag{9}$$

The operations executed by the custom sensors enable us to create bridge-rules that integrate different types of perceptions with custom contexts. These bridge-rules can generically handle several data types, separating the implementation details from the agent's multi-context definition. For instance, one can define a sensor that can perceive image data that custom contexts can handle without changing other contexts or bridge-rules. The perception generated by this custom sensor can be employed as an input to a Convolutional Neural Network (CNN) of the NNC. Following, we summarize how this execution works:

1. *sensor$_i$* processes the received data α, which triggers the agent's reasoning cycle;
2. *sensor$_i$* defines how this data α is transformed to a perception β;
3. *sensor$_i$* adds the perception β to the Communication Context (CC);
4. During the execution of the bridge-rule presented in Definition 9, *sensor$_i$* verifies whether the information in the bridge-rule's body is valid in the CC;
5. Since the perception β was added into the CC, the bride-rule in Definition 9 becomes valid;
6. When the bridge-rule body is valid, the perception β created by *sensor$_i$* is added to the Neural Network Context (NNC).

The final step of this initial integration is achieved by using a neural network's output as part of the agent's decision-making. In this work, we can employ the neural network's output as part of a bridge-rule's body or as a precondition of the agent's plan. Using this output as part of the bridge-rule enables us to decrease the necessity of defining hand-crafted rules as part of the bridge-rules' bodies. The existing version of the Sigon framework only supports verifying whether a specific plan's preconditions are satisfied in the Beliefs Context (BC). We believe this approach did not take advantage of the Multi-Context System's primary goal of considering different knowledge sources. We changed the Sigon grammar to enable modeling preconditions to reference different contexts to accommodate this property. For each context and term of a precondition, the planning context will verify whether a precondition is satisfied by the referenced contexts. This approach allows us to model the interaction between the planning context and several contexts. This bridge-rule is presented in 10.

$$\frac{PC:\ plan(\phi,\beta,Pre,Pos,cost)\ and\ IC:\ \phi\ and\ C_i: Pre}{CC:\ \beta} \tag{10}$$

where ϕ is something the agent wants to be true and can be deduced in the Intentions Context (IC); β the set of actions to be executed; *Pre* is the set of the plan's preconditions; *Pos* is the set of the plan's postconditions; and *cost* is the cost to execute this plan's actions. C_i is in the set of existing contexts of the agent *AG*, in which for logical contexts, *Pre* is true or can be inferred, or for functional contexts, it can be verified whether it is true or not. In Chapter 6, we present a reasoning cycle of the proposed agent and how it can be implemented in Sigon.

5.2.1 Integration's implementation details

This subsection presents the implementation of the proposed agent modeling and the integration method. The BDI agent's modeling revolves around defining custom sensors and contexts. It is important to state that the agent's designer is responsible for defining and implementing the custom sensors and contexts. Even though we can use different AI methods as part of the agent's custom context and integrate them as part of the agent's reasoning cycle, in this research, we focus on exploring the impact of using a Neural Network's output to improve the BDI-agent's reasoning cycle.

In the proposed agent, each sensor is responsible for handling specific data. From the implementation point of view, the sensors and actuators follow the publish/-subscribe pattern, in which the sensors are the publishers and the actuators the subscribers. This pattern allows subscribers to express their interest in an event or a pattern of events to be notified subsequently of any event generated by a publisher (EUGSTER et al., 2003). For instance, assume that an agent has two custom sensors responsible for processing image and text data. In this sense, each custom sensor could have different ways of handling image and textual data and verifying whether a perception is valid. The class Sensor in Sigon has a method called perception. This method is responsible for publishing an event containing the data to be processed by the custom sensor and also provides the implementation of the add method.

In Sigon, a custom sensor should be implemented as a class inherited from the Sensor class. It is required to implement the two following methods:

1. add: this method is responsible for defining how to create a perception from the received data;
2. verify: this method is responsible for validating whether certain information is valid in the Communication Context (CC). Every time the data should be verified in CC during a bridge-rule execution, the verify method implemented by the sensor is executed. This validation can also be performed in different custom bridge-rules.

A similar approach is required to extend the agent's context and add a new custom context. A custom context should be implemented as a class that inherits from the ContextService class. A custom context's class should implement the following methods: append_fact and verify. The append_fact and verify have the same responsibilities as the methods add and verify of the custom sensors. They also define how to handle certain perceptions and how they can be verified in the bridge-rule body. It is essential to notice that a custom context could have several different methods, which can be combined with append_fact and verify executions. Figure 12 summarizes the main aspects of adding a new custom sensor and context. For instance, the Neural Network Context (NNC) class follows the requirements of the Desires Context (DC) class. Two custom sensors were presented: (i) ImageSensor, responsible for handling and creating the perception of images, and (ii) ContractSensor, responsible for handling and creating the perception of textual data.

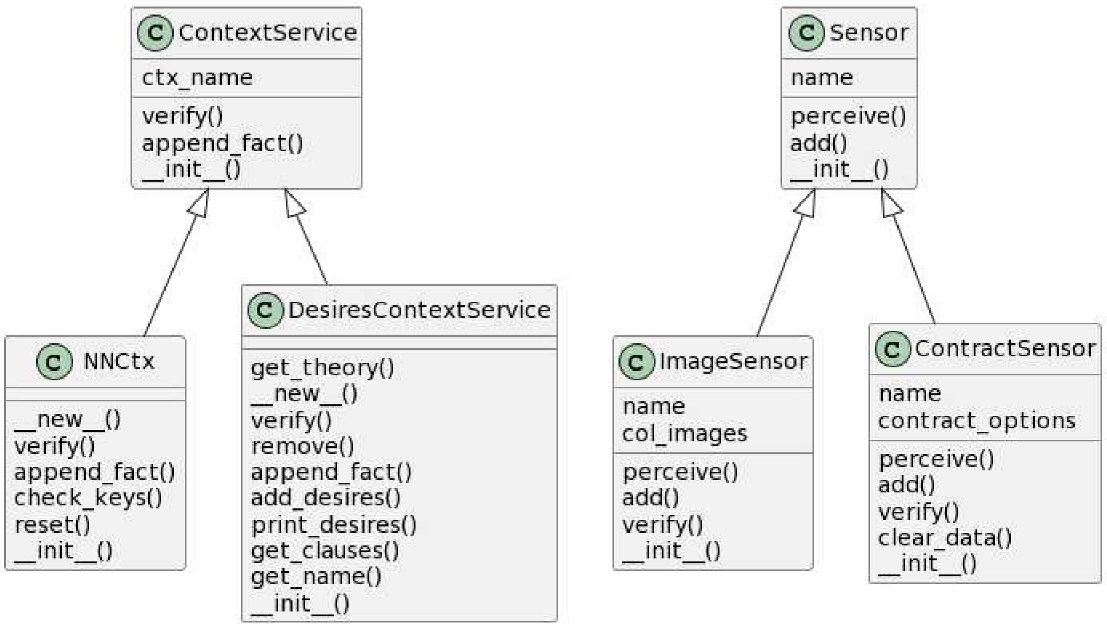


Figure 12 – Diagram class of contexts and sensors.

After defining how each custom sensor and context will handle different data types, perceptions, and information, the agent's designer can also specify custom bridge-rules and how they will interact with the agent's contexts. As mentioned in Chapter 3, Sigon's parser and agent modules will handle the agent's modeling. These modules will transform the source code defined in the '.on' file and link it with the agent's actuators, sensors, contexts, and bridge-rules implementation.

5.3 AGENT'S REASONING CYCLE EXAMPLE

This section presents an example of the agent reasoning cycle and its implementation in Sigon. Negotiation and conflicts arise in almost every social and organizational setting (BAARSLAG et al., 2017). Agents can use negotiation to establish deals and coordinate their actions to achieve their design goals. In this sense, we modeled a negotiating agent that can use facial expression recognition as part of its negotiation strategy to achieve this goal. Facial expression recognition is relevant for studying Emotion Recognition Accuracy (ERA). ERA explores the impacts of emotions on objective outcomes in negotiation, a setting that can be highly emotional and in which real-life stakes can be high (ELFENBEIN et al., 2007). In this negotiation scenario, two persons are trying to define the selling price of a particular item. The negotiation agent will assist the seller by creating proposals on whether the seller should increase or decrease the item's price. In this scenario, we assume that:

1. The agent can perceive information about a negotiation scenario (i.e., price, current negotiation iteration, preferences, and rules) and a person's facial expression

(i.e., happiness, sadness, fear, anger, surprise, or neutrality);

2. In each iteration of the negotiation, the agent can capture a picture of the buyer's face and the new current selling price.

Following the definition presented in Definition 3, the agent's contexts represent the mental attitudes and different knowledge sources. The details about the scenario and the practical implementation are presented in Appendix A. Code 5.1 shows the state of the communication, beliefs, desires, intentions, planner, and _neuralNetwork contexts. The contexts of beliefs, desires, and intentions are modeled in lines 6, 12, and 15. The Beliefs Context (BC) in this scenario is responsible for modeling the negotiation information. For instance, BC holds information about the price of *itemA*, which can be seen in line 7. The desires and intentions are modeled in lines 13 and 16. The custom context called _neuralNetwork detects emotions based on facial expressions. Line 10 shows an example of how the neural network output is visible in the _neuralNetwork context. The planner context is modeled through lines 18 to 20. The planner context holds the predefined plans that can be executed. The predefined plans are presented in lines 19 and 20 in Code 5.1.

```
 1 communication:
 2     sensor("textSensor", "perception.Image").
 3     sensor("textSensor", "perception.Text") .
 4     actuator("sendMessage", "actuator.SendMessage").
 5
 6 beliefs :
 7     price(itemA, 100).
 8
 9 _neuralNetwork:
10     currentEmotion(buyer, happy).
11
12 desires:
13     updateDecision.
14
15 intentions :
16     updateDecision.
17
18 planner:
19     plan(updateDecision,[action(sendMessage(increase))],[_neuralNetwork:
           currentEmotion(buyer,happy)],_).
20     plan(updateDecision,[action(sendMessage(decrease))],[_neuralNetwork:
           currentEmotion(buyer,sad)],_).
21
```

```
22 ! _neuralNetwork X :- communication imageSensor(X).
23 ! beliefs X :- communication textSensor(X).
```

Code 5.1 – Sigon syntax for modeling the proposed agent

The sensors in lines 2 and 3 in Code 5.1 follow the approach presented in Section 5.1, in which each sensor is responsible for defining how a particular data is processed and added into the agent's communication context. For instance, the sensor called *textSensor* captures the picture from the buyer's face and maps it to a perception. The sensor called *textSensor* maps the textual data about the negotiation to perceptions. The actuator called *sendMessage* is responsible for sending messages to the seller of this item. The perceptions added into the communication context trigger the reasoning cycle of this agent. From now on, we assumed that the *textSensor* sensor processed one image from the buyer's face.

Bridge-rules are presented in lines 22 and 23 of Code 5.1, and they are responsible for the information exchange between the contexts. For instance, after the picture of the buyer's face is transformed into perception and added to the communication context, the body of the bridge-rule in line 22 will be valid, and then the information will be added to the _neuralNetwork context (NNC). The information added in the NNC will be used as an input for the neural network. The output provided by the neural network will then be transformed to the term presented in line 10.

A plan's selection will occur based on the definition provided in 10. The plan in line 19 of Code 5.1 will be selected since the term *updateDecision* is true or can be deduced in the intentions context, and the precondition that models the detection of a happy emotion from the buyer is true in the _neuralNetwork context. In this example, the selected plan will trigger the actuator responsible for sending a message to increase the selling price.

5.4 DISCUSSION

This section showed how we could integrate different types of perceptions with custom contexts and how the information provided by these custom contexts can be used during the agent's reasoning cycle without the necessity of changing other existing contexts and bridge-rules. We believe that the presented approach represents appropriate steps during the integration of AI techniques in the sense and planning phases of the BDI cycle. Chapter 6 shows a case study about the proposed agent.

In this chapter, we defined the architecture of the proposed neural-symbolic agent. This agent uses a BDI-like approach and Multi-Context Systems (MCS). BDI-like agents have flexible and robust behavior and are suitable dynamic environments (CHONG; TAN; NG, 2007). MCS enables the development of modular and flexible intelligent systems, which represent important features for neural-symbolic systems.

Besides the modularity and flexibility, the proposed agent also increases the level of abstraction during intelligent systems development. This increase in abstraction can result in the possibility of using different resources without changing other parts of the agent's modeling and reasoning.

The integration of AI methods during the agent's reasoning cycle can be implemented in the three phases of the BDI cycle: sensing, planning, and acting (BORDINI et al., 2020). In this chapter, we addressed the integration of neural networks in the BDI reasoning cycle in the following way:

1. Sensing: We explored how different data types can generate new perceptions, which could be used during the agent's reasoning cycle. Each custom sensor is responsible for integrating specific data into the MCS. In the sensing phase, we also proposed adding bridge-rules that can integrate these perceptions without requiring changes in other contexts or bridge-rules.

2. Planning: In this part of the integration, we showed how custom contexts could be defined and integrated into the agent's planning selection phase. The custom contexts can employ neural networks during the agent's execution. We also simulated a scenario in which only one AI technique could not be sufficient. We presented the required steps to add new contexts to mitigate this limitation. This approach enables us to develop intelligent agents that can combine different methods during the agent's reasoning;

3. A custom context called Neural Network Context (NNC) that can employ the neural network's output;

4. Bridge-rules to handle the information provided by the NNC as part of the rule's body;

6 CASE STUDY

This chapter presents a case study containing two neural-symbolic BDI agents. This chapter explores the required steps of modeling, developing, and deploying these agents in two different scenarios. The case study shows how the agent can handle different types of information and integrate several capabilities during decision-making. Following, we correlate the required steps used in this case study and what was presented in Chapter 5:

1. Define the agent's contexts, such as the beliefs, desires, intentions, communication, planner, and custom contexts (Definition 3);
2. Establish the required custom sensors to handle different information (Section 5.1);
3. Model the Neural Network Context (NNC), defining how the neural network will process the information generated by the agent's sensors (Sections 5.2 and 5.2.1);
4. Use the framework Sigon to model the agent's contexts and bridge-rules (Section 5.3).

Section 6.1 presents the modeling and implementation of a negotiating agent. In Section 6.1, an agent that employs a negotiation strategy to solve the conflict without handling a significant amount of data is implemented. In this scenario, we integrate the neural network's output as part of the agent's reasoning cycle. Section 6.2 presents an agent for malware detection. In Section 6.2, we aim to employ a scenario that can complement the previous one. Considering these, a more complex and closer to a real-world scenario is presented, requiring the agent to process a substantial volume of data. We also explore how the agent can change the neural network's structure and parameters. The agents implemented in this chapter will also be employed during our work's evaluation.

6.1 NEGOTIATING AGENT

This agent's modeling and implementation are divided as follows: (i) defining the agent as a Multi-Context System (i.e., sensors, actuators, contexts, bridge-rules, and plans) following Sigon's syntax; and (ii) implementing the proposed agent. The agent implemented in this case study is similar to the definition presented in Definition 3 in Section 5.3. The main difference is that this agent has a new custom context called Negotiation Context (NC). This context is responsible for modeling the agent's negotiation strategy without changing other contexts or the agent's reasoning cycle. Definition 4 presents the agent's contexts.

Definition 4

$$NA = \langle \{BC, DC, IC, PC, NNC, NC, CC\}, \Delta_{br} \rangle, \tag{11}$$

where BC, DC, IC, PC, NNC, NC, CC are the beliefs, desires, intentions, planner, neural network, negotiation context, and communication contexts; and Δ_{br} are the bridge-rules for exchanging information between contexts.

Since conflicts arise in almost every interaction in our daily tasks, the agents must employ negotiation strategies to establish deals and coordinate their actions to achieve their design goals. Considering this necessity, the agent modeled in this section is a negotiating agent. This agent's version will also be used as a starting point for comparing similar works to ours. We chose the negotiation scenario based on its usage in different works and how it can be compared with ours. The negotiation scenario is based on Rosenfeld and Kraus (2012) work, and it is used as part of the Automated Negotiating Agents Competition (ANAC), which enables the evaluation and benchmark of negotiating agents (JONKER et al., 2017). This domain consists of an employer and employee defining the hiring terms of a job position. In the negotiation session, both the employer and the job candidate wish to formalize the hiring terms and conditions of the job position. The hiring terms in this scenario are the following:

1. Salary: The possible values are (a) \$7,000, (b) \$12,000, or (c) \$20,000;
2. Job description: Responsibilities given to the employer. The possible values are (a) QA, (b) Programmer, (c) Team Manager, or (d) Project Manager;
3. Car benefit: Other job benefits may also be negotiated in addition to the base salary. This term revolves around the possibility that the company will provide a company car for use by the employee. Possible values for this issue are (a) providing a leased company car, (b) no leased car, or (c) no agreement;
4. Pension benefit: The possible value for the percentage of the salary deposited in pension funds are (a) 0%, (b) 10%, (c) 20%, or (d) no agreement;
5. Promotion possibility: This issue describes the commitment by the employer regarding the fast track for promotion for the job candidate. The possible values are (a) fast promotion track (2 years), (b) slow promotion track (4 years), or (c) no agreement;
6. Working hours: This issue describes the number of working hours per day (not including overtime). The possible values are (a) 8 h, (b) 9 h, or (c) 10 h.

6.1.1 Negotiation Context (NC)

Three main areas should be taken into consideration during negotiating agents' development: (i) domain knowledge and preference extraction; (ii) long-term negotiations; (iii) user trust and adoption of the system (BAARSLAG et al., 2017). Considering these areas, the main goal of this custom context is developing the ability to represent preferences based on perceptions and the negotiation domain. To tackle these properties, we modeled a negotiation context responsible for processing, defining, and providing information that can be used during conflict resolution. We justify this decision as follows:

- We can easily adapt the negotiation strategies when the agent is deployed in a different domain. Centralizing the negotiation strategy also removes the need to change other existing contexts and bridge-rules, thus removing the necessity of changing the agent's reasoning cycle;
- Some negotiation scenarios may have different protocols, norms, and rules. Other auctions, such as English, Dutch, and Vickrey, may vary in choosing the winning bid (JENNINGS et al., 2001). However, some rules are the same. Adding custom contexts and sensors that can be easily changed could represent significant progress during intelligent agent development.
- Adding a context responsible for implementing different strategies and protocols may reduce the time to integrate with other architectures. This decision also increases the flexibility of our model.

We define a Negotiation Context (NC) as the context with strategies that can be used during conflict resolution. Each of these strategies can have different complexity, varying between utility functions to more elaborated ones, such as argumentation or an approach integrated with existing contexts. The NC is responsible for providing information that can be used in different contexts. In our work, this information is employed by the Planning Context (PC), more precisely, as the plan's precondition. This approach enables us to add a new NC into the agent reasoning cycle without changing the planning context, making the negotiation process more modular and flexible.

6.1.2 A negotiation strategy based on Aspiration Adaptation Theory

Many real-world problems cannot be easily quantified based on traditional expected utility models (ROSENFELD; KRAUS, 2011; VON NEUMANN; MORGENSTERN, 2007). Creating a utility model may consume a lot of computational resources and time. Utility functions are susceptible to subtle contextual factors. Therefore, agents' developers should consider this aspect during preference elicitation and conflict resolution

(BAARSLAG et al., 2017). Taking these situations into consideration, Selten (1998) proposed a framework called Aspiration Adaptation Theory (AAT) as a bounded rational model of decision-making (ROSENFELD; KRAUS, 2011). Rosenfeld and Kraus (2012) define AATas follows:

Definition 5 *Let G be a complex problem where G cannot be directly solved; therefore, it is required to create m goal variables $G_1,...,G_m$ to solve G, in which m is sorted in order of priority, of urgency.*

Based on Selten (1998) works, Rosenfeld and Kraus (2012) define the following properties in regard of aspiration levels:

- each of the goal variables has a desired value, or its urgency level, that the agent sets for the current period;
- the agent defines the initial aspiration level based on local procedural preferences. The local procedural specifies which aspiration levels are most urgent and how they can be adapted upward or downward;
- which partial aspiration level is retreated from or adapted downward if the current aspiration level is not feasible.

In this case study, we model a negotiation strategy based on key elements of AAT. The key aspects of AAT used in this research consist of the following properties: goal variable, the urgency of each goal variable, and possible actions to accomplish these goal variables.

Definition 6 *A goal variable g_i is defined as:*

$$g_i = (a_i, v_{a_i}) \tag{12}$$

where a_i represents a negotiable attribute, e.g., working hours, salary, promotion possibilities, car benefits, etc. v_{a_i} is a possible value of a_i, such as the preference to work 20 hours a week, or a fast promotion possibilities.

The aspiration levels for a goal variable could be defined based on the agent's beliefs, desires, intentions, or other capabilities. Assuming that an agent has a belief that models the distance between the house of a person and its possible work location. A car may not be needed in this case; therefore, the agent can assign a low aspiration level for the goal variable representing car benefits. Based on the aspiration level of each goal variable, a negotiation strategy defines the actions that can be taken to achieve the goal variable. Figure 13 shows the execution cycle of the negotiation strategy based on the AAT proposed in this work.

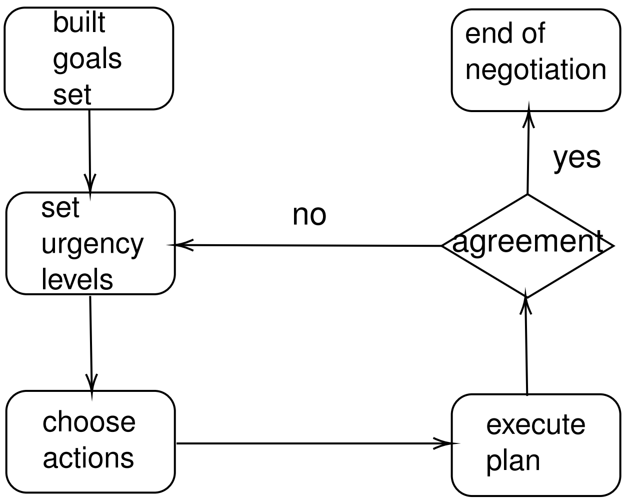


Figure 13 – Cycle of the negotiation strategy based on AAT. Adapted from Selten (1998).

The agent's designer defines the urgency level of a goal variable in the NC via bridge-rules. The possibilities of a goal's urgency level play an essential role during negotiation (SELTEN, 1998). Having a large span of urgency levels increases the chance of changes. This span of possibilities could extend the negotiation. In this scenario, we limit the range of urgency levels between 1 and 10. An example to clarify how this mechanism works is presented. Assuming that an agent represents a person during the definition of the hiring terms of a job proposal, both parties must decide the number of weekly working hours that should be followed. *L* is formed as:

$$\begin{aligned} L = [(\mathit{working\_hours}(40), 2), (\mathit{working\_hours}(30), 8), \\ (\mathit{working\_hours}(20), 10)] \end{aligned} \tag{13}$$

Using the list *L*, an agent could define the number of weekly working hours based on the goal variable with a higher urgent value. In this example, we determine 20 as the proposed value. In subsection 6.1.3, we present how this strategy is added to the agents' reasoning cycle and the required bridge-rule to integrate this strategy during conflict resolution.

6.1.3 Adding a negotiation strategy into the BDI-agent's reasoning cycle

Based on human decision-making, a person's preferences may affect how urgent a goal variable is (ROSENFELD; KRAUS, 2012). We modeled this aspect as part of the negotiation strategy in our work. The Negotiation Context (NC) in Sigon is represented as follows:

$$\mathit{negotiation\_context} : \mathit{urgency}(\varphi, \gamma). \tag{14}$$

in which φ is a goal variable, and γ is a value that represents the urgency of φ. One of the main limitations of the AAT is to provide an effective way of defining the urgency level of a variable goal. In this sense, it could be necessary to have hand-crafted rules, which could increase domain dependence. In our work, we use agents' contexts (i.e., Beliefs Contexts, Desires Contexts, Intentions Contexts, etc.) during the definition of the urgency of a goal variable. The bridge-rule shown in 15 models this interaction between the agents' contexts and the Negotiation Context (NC):

$$\frac{C_i : \alpha \wedge C_j : \beta}{NC : \gamma}, \tag{15}$$

in which, $C_i \in NA$ (Negotiating Agent) and $C_j \in NA$. When α is valid or deduced in C_i, β is valid or deduced in C_j, then γ is added to the Negotiation Context (NC). In this work, the Negotiation Context is responsible for handling the added information and defining a strategy to establish the urgency level of each goal variable.

6.1.4 Implementing the Neural Network Context (NNC)

Since defining the urgency of a particular goal variable requires hard-coded and hand-crafted rules, we explore using an NN to mitigate this limitation. In this sense, we deployed a trained Neural Network (NNC) in the Neural Network Context (NNC). The NN's output is available in the NNC and can be used to provide information to assist with the definition of urgency for the goal variables. The NN project can be accessed in https://colab.research.google.com/drive/1j7vKBwa431nJd2H8NH2J5G3pCqBhaWt2. The data used to train the NN was obtained from job descriptions from Glassdoor.com. Glassdoor.com is a website where employees and former employees anonymously review companies and their management (LUO; ZHOU; SHON, 2016).

Code 6.1 shows the neural network's layers and activation functions. The NN has three layers. The first layer is fully connected, with 174 features in the input and 88 in the output. The second layer is also fully connected, with 88 input and output. The third layer receives 88 features and produces one output. The output models the average salary. The neural network uses ReLU as the activation function. The trained neural network achieved a loss of 2.28224 while running for 1000 epochs. It is worth mentioning that this work did not focus on the NN design. The main goal of this NN is to support the agent's reasoning cycle with reasonable accuracy. In this sense, the NN predicts an average salary.

```

(0): Linear(in_features=174, out_features=88, bias=True)
(1): ReLU()
(2): Linear(in_features=88, out_features=88, bias=True)
(3): ReLU()
```

```
  (4): Linear(in_features=88, out_features=1, bias=True)
```

Code 6.1 – Neural network's layers and activation functions

The agent's execution follows the same approach defined in Gelaim et al. (2019) and Rodrigo Rodrigues Pires de Mello, Silveira, and Santiago (2022), which consists of the agent's sensors processing data and triggering the reasoning cycle. A version of a BDI-agent modeled in Sigon can be found in GitHub's repository in https://github.com/sigon-lang/sigon.

Listing 6.2 shows the initial state of the communication, beliefs, desires, intentions, planner, negotiation, and _neuralNetwork contexts. The Communication Context (CC) models the sensors and actuators. The Communication Context (CC) adds the perceptions these sensors generate. The contractSensor in line 2 generates perceptions based on the job's hiring terms. The cvSensor in line 3 reads the user's resume or CV and generates a perception. The sendMessage actuator in line 4 sends the proposal about the hiring terms defined after the agent's reasoning cycle. The agent's developer should implement both sensors and actuators. In listing 6.3, we present an example of the data processed by the cvSensor and contractSensor.

The contexts of beliefs, desires, and intentions are modeled in lines 6, 8, and 10, respectively. The beliefs context in this scenario is responsible for modeling the user's preferences and information about the negotiation. For instance, the Beliefs Context (BC) holds information about how many hours the represented user has already worked, which can be seen in line 7. The _NeuralNetwork Context (NNC) in line 14 defines the average salary based on the perceptions generated by the *cvSensor*.

Line 15 shows how the neural network's output is modeled in the NNC. The Negotiation Context (NC) in line 18 implements the strategy based on AAT. It uses the definition presented in section 6.1.2 to establish the urgency level. The NC can employ the information provided by different contexts. Lines 19, 20, and 21 show examples of the urgency level in the NC. The planner context is modeled through lines 23 to 29. The planner context holds the agent's predefined plans. The predefined plan represents a proposal containing the most relevant variables.

Bridge-rules are presented in lines 31, 32, 34, and 35 in Listing 6.2 and are responsible for the information exchange between contexts. For instance, after the contract sensor processes the hiring terms and adds the perception into the communication context, the body of the bridge-rules in lines 31 and 32 will become valid. Then, the information will be added to the beliefs and negotiation contexts. The same approach happens for the bridge-rule in line 34, in which the perception about the user's resume will be added into the _NeuralNetwork Context (NNC). When this perception is added to the NNC, it will be used as an input of the trained neural network.

The neural network will process the input and define the average salary based on the user's resume. The last bridge-rule in line 35 adds the average salary into the negotiation context. Based on this information, the Negotiation Context (NC) establishes the urgency level of a particular variable. For this scenario, the NC executes a function responsible for finding the salary ((a) $7,000, (b) $12,000, or (c) $20,000) that is closer to the average salary.

If the Negotiation Context (NC) establishes a goal variable with urgency 10, then the precondition of the plan defined in line 24 will become valid. The plan will be selected since the *negotiateContrat* is true in the Intentions Context (IC). In this current reasoning cycle, the Planner Context (PC) will define the action of sending a proposal with 12000 for the hiring term representing the salary. The PC will add the chosen action into the Communication Context (CC), which triggers the actuator defined in line 4. For the sake of simplicity, we omitted different plans with different urgency levels.

```
 1 communication:
 2     sensor("contractSensor", "sensors.ContractSensor").
 3     sensor("cvSensor", "sensors.CVSensor").
 4     actuator("sendMessage", "actuators.SendMessage").
 5
 6 beliefs :
 7     workingHours(2).
 8 desires:
 9
10 intentions :
11     negotiateContract.
12
13 //defines the avg salary based on the sensors input
14 _nn:
15     avgSalary(11000).
16
17 //implements AAT based on different information
18 _negotiation:
19     urgency(salary, 7000, 5).
20     urgency(salary, 12000, 10).
21     urgency(salary, 20000, 8).
22
23 planner:
24     plan(
25         negotiateContract,
26         [action(sendMessage(X,Y))],
27         [_negotiation:urgency(X, Y, 10)],
28         [_]
29     ).
```

```
// rules responsible to create new perceptions from different knowledge sources
! beliefs X :- communication contractSensor(X).
! _negotiation X :- communication contractSensor(X).

! _nn X :- communication cvSensor(X).
! _negotiation avgSalary(X) :- _nn avgSalary(X).
```

Code 6.2 – Sigon syntax for modeling the proposed agent

```

  job_contract = {
       Salary: The possible values are (a) \$7,000, (b) \$12,000, or (c) \$20,000;
       "salary": [7000, 12000, 20000],
       Responsibilities given to the employer. The possible values are (a) QA, (b)
           Programmer, (c) Team Manager, or (d) Project Manager;
       "jobDescription": ["qa", "programmer", "teamManager", "projectManager"],
       (a) providing a leased company car, (b) no leased car, or (c) no agreement;
       "carBenefits": ["yes", "no", "noAgreement"],
       Pension benefits: The possible value for the percentage of the salary
           deposited in pension funds are (a) 0\%, (b) 10\%, (c) 20\%, or (d) no
           agreement;
       "pensionBenefits": ["0", "10", "20", "noAgreement"],
       The possible values are (a) fast promotion track (2 years), (b) slow
           promotion track (4 years), or (c) no agreement;
       " promotionPossibilities ": ["2", "4", "noAgreement"],
       This issue describes the number of working hours required by the employee
           per day The possible values are (a) 8 h, (b) 9 h, or (c) 10 h.
       "workingHours": [8, 9, 10],
       "negotiate": ["contract"]
   }
   cv_data provides information about the users skills and experience
    cv_data = {"python_yn": 1,
               "spark": 1,
               "aws": 1,
               "excel": 1,
               "job_simp": "data scientist ",
               " seniority ": "senior"
               }
```

Code 6.3 – Hiring terms and CV data

6.2 NEURAL-SYMBOLIC AGENT FOR MALWARE DETECTION

In the previous implementation, the scenario was oriented towards employing a negotiation strategy to solve the conflict without handling a significant amount of data. In this following implementation, we intend to shift towards a more complex and closer to a real-world scenario in which it is required to process a substantial volume of data. The last implementation focused on using the neural network's outputs to mitigate the necessity of hard-coded and hand-crafted rules. In this current implementation, our primary goal is to model a neural-symbolic agent that can change the neural network structure and parameter during its reasoning cycle. This approach enables us to analyze the integration method's flexibility and modularity.

The dataset used in this case study consists of malicious and benign software called Endgame Malware BEnchmark for Research (EMBER) (ANDERSON; ROTH, 2018). EMBER includes real applications from 2018, in which the features extracted from 1.1M binary files contain 900K training samples (300K malicious, 300K benign, 300K unlabeled) and 200K test samples (100K malicious, 100K benign) (ANDERSON; ROTH, 2018). The EMBER dataset consists of a collection of JSON files, where each line contains a single JSON object. The EMBER dataset can be accessed in https://github.com/elastic/ember. Each object includes the following types of data:

- the sha256 hash of the original file as a unique identifier;
- coarse time information (month resolution) that establishes an estimate of when the file was first seen;
- a label, which may be 0 for benign, 1 for malicious, or -1 for unlabeled; and
- eight groups of raw features that include both parsed values and format-agnostic histograms.

Since one of the main objectives of our work is to deploy the agent in a real-world scenario, we are concerned about the necessity of the agent handling new samples from different months in a dynamic environment. To simulate this aspect, we follow the strategy presented in Rahman, Coull, and Wright (2022), which divides and processes the dataset monthly. This decision allows us to explore a scenario with Continual Learning (CL) properties. Continual learning (CL) is a machine learning paradigm where the data distribution and learning objective change over time, or all the training data are never available at once (LESORT et al., 2020). In this sense, CL focuses on mitigating catastrophic forgetting when new data are presented to the neural network (VAN DE VEN; TOLIAS, 2019; LESORT et al., 2020).

We first attempted to employ the Multi-Layer Perceptron presented in Rahman, Coull, and Wright (2022). We were not able to achieve a reasonable accuracy with their

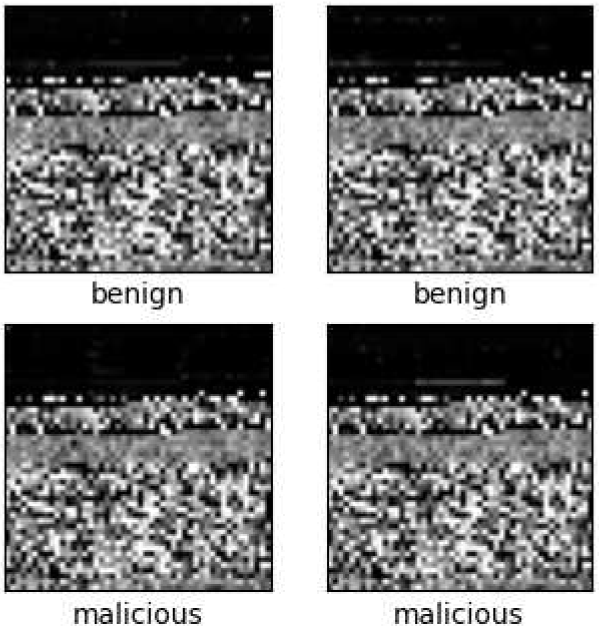


Figure 14 – Example of EMBER's input as images

MLP. Our second approach was to employ a CNN called Malconv (RAFF et al., 2018). However, we faced two limitations: (i) Malconv requires raw-byte sequence inputs, and (ii) the computational resources available to process the EMBER dataset were not adequate. Therefore, we cannot use MalConv with EMBER. To overcome these limitations, we handled the input of the dataset as an image and modeled a simpler CNN that could process the dataset in a reasonable time. Figure 14 presents an example of benign and malicious software as images. The code for handling the EMBER dataset and the CNN we used as a baseline can be accessed in https://www.kaggle.com/code/olufelaa/ml-for-static-malware-detection. Our adopted strategy is similar to the one presented in Ghouti and Imam (2020).

Figure 15 shows the CNN architecture used in this case study. The CNN has three convolutional layers. The first layer specifies the input shape as (48, 48, 1), meaning the input images are expected to have dimensions 48x48 with a single channel (grayscale). Each layer will employ 128 filters to learn different features. The kernel size is set to (3,3). The activation function used was the Rectified Linear Unit (ReLU). A window of 2x2 is used in the Max pooling step. No padding strategy was used in this scenario. After the flatten layers are used to reshape the 3D output from the convolutional layer into a vector with one dimension, there are two fully connected layers. The first one has 400 neurons and employs ReLU as the activation function. The second fully connected layer has a single neuron with a sigmoid activation function. The output models whether the software is benign or malicious.

The optimizer, loss function, and metrics used to train the model were Adam, Binary cross-entropy, and accuracy. The default learning rate and number of epochs

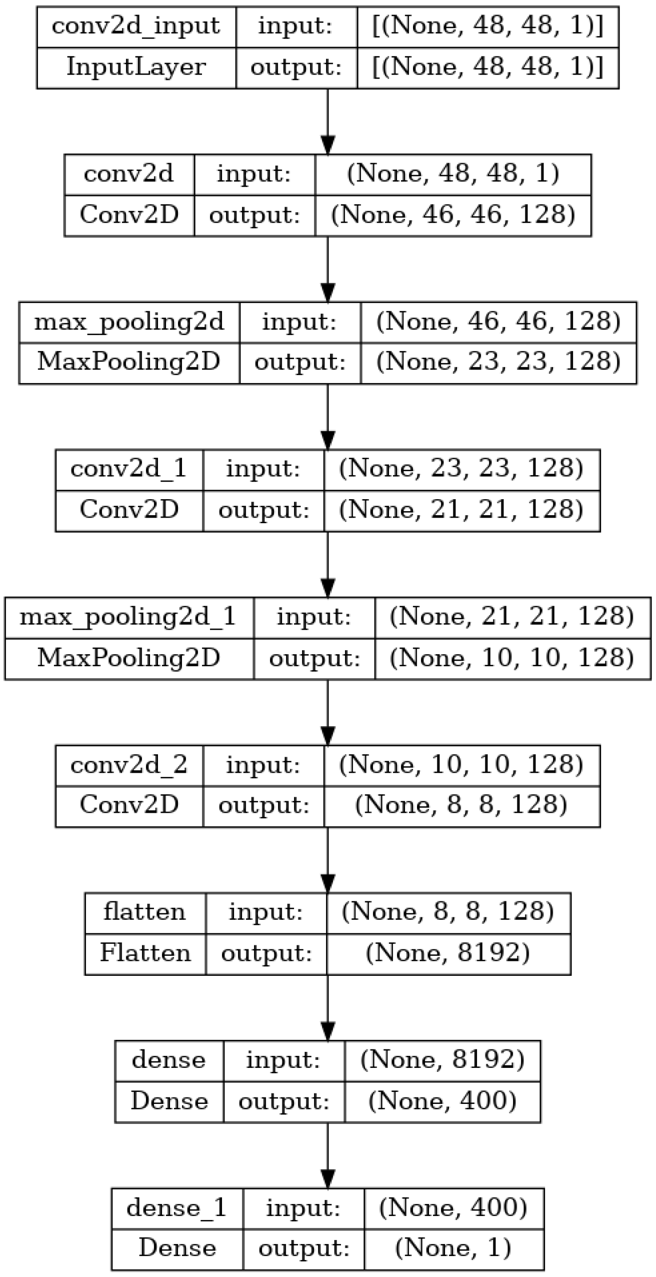

conv2d_input | input: | [(None, 48, 48, 1)]
InputLayer | output: | [(None, 48, 48, 1)]
conv2d | input: | (None, 48, 48, 1)
Conv2D | output: | (None, 46, 46, 128)
max_pooling2d | input: | (None, 46, 46, 128)
MaxPooling2D | output: | (None, 23, 23, 128)
conv2d_1 | input: | (None, 23, 23, 128)
Conv2D | output: | (None, 21, 21, 128)
max_pooling2d_1 | input: | (None, 21, 21, 128)
MaxPooling2D | output: | (None, 10, 10, 128)
conv2d_2 | input: | (None, 10, 10, 128)
Conv2D | output: | (None, 8, 8, 128)
flatten | input: | (None, 8, 8, 128)
Flatten | output: | (None, 8192)
dense | input: | (None, 8192)
Dense | output: | (None, 400)
dense_1 | input: | (None, 400)
Dense | output: | (None, 1)

Figure 15 – CNN’s architecture

were 0.001 and 10. Initially, we added more layers to this CNN and tested different configurations for the number of epochs and learning rate. This setup improved the CNN accuracy. However, it significantly increased the time to train the neural network and process the whole dataset. Since we are interested in analyzing the impact of combining the proposed method with this NN, the results we achieved with the chosen architecture were considered sufficient.

We also applied two techniques related to transfer learning to assist the training on the new data from the whole year. Transfer learning aims to provide a framework to utilize previously acquired knowledge to solve new but similar problems much more quickly and effectively (LU et al., 2015). The approaches employed for transfer learning in this scenario were based on the guidelines from (CHOLLET, François et al., 2015). Following, we present the description of the approaches used in this case study:

1. Train: we trained the model with the data from different months without any specific technique related to fine-tuning;
2. Feature extraction: in this version, we executed the following steps:
   a) Load the model resulted from the training of the previous month;
   b) Set the layers from the base model to be not trainable (i.e., freeze the layers), which enables the CNN to keep some information from previous training;
   c) Add a new trainable layer on top of the base model;
   d) Train the new layer using the data from the current month.
3. Fine-tuning: This version employs the model resulting from the feature extraction; however, it sets the layer to become trainable and then trains the whole model.
4. Neural-symbolic BDI-agent: this version combines the three approaches (train, feature extraction, and fine-tuning). We assume the training method requires more time than fine-tuning and feature extraction. Although this method could require more time, a new layer is added to the NN every time a feature extraction is performed. Considering these, the agent's reasoning cycle is responsible for defining when it is necessary to use one of these strategies. The decisions about which approach to use are based on the CNN's accuracy from previous training. The agent models this strategy through its plans and bridge-rules.

The BDI-agent developed in this case study follows the approach presented in chapter 5. Definition 7 shows the neural-symbolic agents contexts. In this version, we add a new context called Metrics Context (MC). The Metrics Context is responsible for analyzing data about the NN's accuracy and loss. The information about the NN's performance was modeled in the Neural Network Context (NNC). We integrated both contexts through bridge-rules. The action about what approach the agent should employ to train the neural network is modeled in the agent's Planner Context (PC). Based on the information modeled in the Metrics Context (MC), the agent triggers a plan with the defined action and actuator. Each plan models whether the agent should train, execute fine-tuning, or feature extraction.

Definition 7

$$NA = \langle \{BC, DC, IC, PC, NNC, MC, CC\}, \Delta_{br} \rangle, \tag{16}$$

where BC, DC, IC, PC, NNC, NC, CC *are the beliefs, desires, intentions, planner, neural network, metrics context, and communication contexts; and* Δ_{br} *are the bridge-rules for exchanging information between contexts.*

Code 6.4 presents the initial mental state of the neural-symbolic agent for malware detection. The *hardwareSensor* is responsible for gathering information about the machine, such as the CPU and memory usage. The *dataSensor* is responsible for processing data from the EMBER dataset. The actuator *setOperation* defines whether the CNN should execute a new training, a feature extraction, or a fine-tuning. The agent's beliefs contain information about the availability of computational resources integrated via bridge-rule with the *hardwareSensor*. Initially, the agent does not have any intentions or desires. The Metrics Context (MC) is presented in line 14. The MC models a rule that represents the current performance of the NN. The Neural Network Context (NNC) contains the CNN employed and information about the CNN accuracies and losses during the training. Since the NNC has no information about accuracy and loss, the MC starts with performance defined with a low value. The agent's plans are presented from lines 23 to 40. The bridge-rules used to integrate the information between different contexts are given in lines 43, 45, 47, 49, and 51.

```
 1 communication:
 2     sensor("hardwareSensor", "sensors.HardwareSensor"). //information about the
           cpu and mememory usage
 3     sensor("dataSensor", "sensors.DataSensor"). //ember dataset to be processed
           by the agent
 4     actuator("setOperation", "actuators.SetOperation"). //defines the next operation
           and creates a new perception of these operations
 5
 6 beliefs :
 7     time(available ) .
 8
 9 desires:
10
11 intentions :
12
13 //provides information about nn performance
14 _metrics:
15     performance(low).
16     //performance(low). < 60
17     //performance(medium). >= 60 < 92
18     //performance(high).  >= 92
19
20 _nn:
21
22 planner:
23     plan(
24         improveAccuracy,
25         [action(setOperation(train)) ],
26         [_metrics:performance(low), beliefs:time(available),
               beliefs :resources(high)],
27         [_]
28     ).
29     plan(
30         improveAccuracy,
31         [action(setOperation(fineTuning))],
32         [_metrics:performance(medium)],
33         [_]
34     ).
35     plan(
36         improveAccuracy,
37         [action(setOperation(featureExtraction)) ],
38         [_metrics:performance(high)],
39         [_]
40     ).
41
42
43 ! beliefs X :- communication hardwareSensor(X). // adds colected data about the
       HW to the beliefs context.
44
45 ! _nn setOperation(X) :- communication sense(operation(X)). // defines an operation:
       train , feature extraction or fine-tuning.
46
47 ! _nn execute(X) :- communication dataSensor(X). // executes an operation with the
       current available data.
48
49 ! _metrics history_trainings (X,Y) :- _nn history_trainings (X, Y). // X = acc value
       and Y = loss value
50
51 ! desires improveAccuracy :- _metrics performance(low).
```

Code 6.4 – Initial mental state of the neural-symbolic agent for malware detection

In the first reasoning cycle, we simulated the step in which the *hardwareSensor* gathers data about the machine and perceives the resources' availability. The first execution generates the perception *hardwareSensor*(*resources*(*high*)). This perception represents a situation in which the computational resources are highly available, meaning that the agent can use these resources. This perception triggers the bridge-rule in line 43, resulting in a new belief. After the execution of the last bridge-rule, the only bridge-rule that is valid is the one presented in line 51. This bridge-rule adds a new desire to improve the NN's accuracy. This desire also triggers an internal bridge-rule, adding a new intention to improve the accuracy. This internal bridge-rule followed the Definition 6 presented in Section 3.5. With this new intention, the plan defined in line 23 becomes valid, triggering the action of training the neural network. Code 6.5 shows the mental state of the neural-symbolic agent after the first reasoning cycle. The actuator *setOperation* creates a new perception and adds to the Communication Context (CC). In Code 6.5, we presented only the relevant parts of the agent's contexts and bridge-rules.

```
 1 communication:
 2     sensor("hardwareSensor", "sensors.HardwareSensor"). //information about the
           cpu and mememory usage
 3     sensor("dataSensor", "sensors.DataSensor"). //ember dataset to be processed
           by the agent
 4     actuator("setOperation", "actuators.SetOperation"). //defines the next operation
           and creates a new perception of these operations
 5     operation( train ) .
 6
 7  beliefs :
 8     time(available ) .
 9     resources(high).
10
11 desires:
12     improveAccuracy.
13
14  intentions :
15     improveAccuracy.
16
17  //provides information about nn performance
18 _metrics:
19     performance(low).
20 _nn:
21
22 planner:
23     plan(
24         improveAccuracy,
25         [action(setOperation(train)) ],
26         [_metrics:performance(low), beliefs :time(available ) ,
               beliefs :resources(high)],
27         [_]
28     ) .
```

```

! beliefs resources(high) :– communication hardwareSensor(resources(high)).

! desires improveAccuracy :– _metrics performance(low).
```

Code 6.5 – Mental state of the neural-symbolic agent after the first reasoning cycle

The next reasoning cycle is initiated when the *dataSensor* sensor starts to process the data from January. After the *dataSensor* generates a perception, the bridge-rules in lines 44 and 46 of Code 6.6 become valid. The bridge-rule in line 44 defines which method should be used to process the EMBER dataset. In this reasoning cycle, the neural network will employ the train method to process the data from January. After the bridge-rule in line 46 finishes its execution, the Neural Network Context starts the execution of the CNN based on the perception added from the Communication Context. After the neural network finishes to process the data, the information about the NN's accuracy and loss are available in the NNC. For instance, the NNC persists the following values: history_trainings(0.94, 0.14), in which the first value is the accuracy and the second value is the loss. When these information becomes available, the bridge-rule in line 48 becomes valid. The Metrics Context define the NN's performance based on the following rules: (i) *performance*(*low*) if accuracy is less then 60; (ii) *performance*(*medium*) if accuracy is greater equals then 60 and less then 92; and (iii) *performance*(*medium*) if accuracy is greater equals then 92. Based on the values for the accuracy and loss, the MC sets the NN's performance to high.

```
 1
 2 communication:
 3     sensor("hardwareSensor", "sensors.HardwareSensor").
 4     sensor("dataSensor", "sensors.DataSensor").
 5     actuator("setOperation", "actuators.SetOperation").
 6     operation( train ) .
 7
 8 beliefs :
 9     time( available ) .
10     resources(high).
11
12
13 desires:
14     improveAccuracy.
15
16 intentions :
17     improveAccuracy.
18
```

```
19
20 _metrics:
21     performance(high).
22 _nn:
23
24 planner:
25     plan(
26         improveAccuracy,
27         [action(setOperation(train)) ],
28         [_metrics:performance(low), beliefs:time(available),
               beliefs :resources(high)],
29         [_]
30     ).
31     plan(
32         improveAccuracy,
33         [action(setOperation(fineTuning))],
34         [_metrics:performance(medium)],
35         [_]
36     ).
37     plan(
38         improveAccuracy,
39         [action(setOperation(featureExtraction)) ],
40         [_metrics:performance(high)],
41         [_]
42     ).
43
44 ! _nn setOperation(train)  :- communication sense(operation(train)).
45
46 ! _nn execute(X) :- communication dataSensor(X). // executes an operation with the
       current available  data.
47
48 ! _metrics  history_trainings (X,Y) :-  _nn  history_trainings (X,  Y).  // X = acc value
       and Y = loss value
```

Code 6.6 – Mental state of the neural-symbolic agent during the second reasoning cycle

When the NN's performance information is available in the MC, the plan in line 37 of Code 6.6 becomes valid. The selected action in this reasoning cycle is *setOperation*(*featureExtraction*). This operation will replace the last one (*operation*(*train*)) in the CC. In this sense, the agent's decision-making establishes the operation (training, fine-tuning, or feature extraction) employed during the NN training in the next reasoning cycle. Since we are concerned about exploring how the agent changes the neural network structure, we did not implement a mechanism to evaluate whether a specific software is benign or malign.

6.3 DISCUSSION

This chapter presented a case study of two neural-symbolic agents, which showed the required steps for modeling and implementing these agents. Each agent followed the steps shown in Chapter 5. The first agent is a negotiating agent responsible for defining the most relevant terms of a job contract. In the first implementation, we employed the neural network's output during the agent's reasoning cycle. The implemented agent in the first scenario uses a negotiation strategy to solve the conflict without handling a significant amount of data. The second agent assists during the training of a NN for malware detection. The second scenario complements the previous one by requiring processing a real-world dataset with 900K training samples (300K malicious, 300K benign, 300K unlabeled) and 200K test samples (100K malicious, 100K benign). Different from the previous agent, this agent changes the neural network's structure by employing different fine-tuning and feature extraction.

A negotiating agent was presented in the first case study presented in Section 6.1. The main goal of this case study is to employ a neural network's output to mitigate the necessity of using hand-crafted and hard-coded rules. Two custom sensors were modeled to process different information in this scenario. These sensors process information about the job contract and the user's CV. We added two new custom contexts: Neural Network Context (NNC) and Negotiation Context (NC). The NNC used a trained NN that establishes an average salary based on the user's CV. The Negotiation Context (NC) modeled the strategy employed by the agent. The integration and information flow between the agent's sensors and contexts were achieved via the bride-rules.

The second case study in Section 6.2 showed an agent for malware detection. This agent follows the same approach as the previous one, in which custom contexts, sensors, and bridge-rules are modeled. Unlike the previous case study, in which the NN's output is used to mitigate the necessity of hard-coded rules, in this case study, our goal is to employ the agent's decision-making to change the NN's structure and also establish how the NN can process a dataset. The method used by the NN revolves around defining when to use a transfer learning technique (fine-tuning and feature extraction) or retrain the whole model. The main goal of transfer learning is to provide a framework to utilize previously acquired knowledge to solve new but similar problems much more quickly and effectively (LU et al., 2015).

The agent monitors the NN's performance and establishes whether training or executing a transfer learning technique (fine-tuning or feature extraction) would be appropriate. The decision about the operation is based on the NN's accuracy and the available computational resources. For instance, only re-training the model would require more time and could result in catastrophic forgetting. Although we assumed that training could require more time, it is important to notice that every time a feature extraction and fine-tuning are performed, a new layer is added to the previous NN, which could increase the time to process the dataset. We believe the agent should use the accuracy from previous training to define which method would be adequate. The agent models this strategy through its plans and bridge-rules.

This chapter showed how we could incorporate different types of perceptions with custom contexts and how the information provided can be used during the agent's reasoning cycle without changing other existing contexts and bridge-rules. These steps focused on integrating the neural network in the sense and planning phase of the BDI cycle. The agents implemented in this case study showed that the integration method provided enough flexibility and modularity to employ the neural network's outputs or change its structure during the agent's decision-making.

7 EVALUATION OF THE MODEL

In Chapter 6, two neural-symbolic agents were modeled, implemented, and deployed. Considering these, in this chapter, our primary goal is to evaluate the trade-off of employing the implemented agents. Section 7.1 evaluates the negotiating agent, focusing on analyzing the impacts of using the neural network's output to mitigate the necessity of hand-crafted and hard-coded rules. The negotiation scenario also enabled us to compare the neural-symbolic agent with several works from the literature. Section 7.2 presents an evaluation of the agent for malware detection. This evaluation compares the agent, which can employ a training method and a fine-tuning technique, with two versions that only use one of these methods during the dataset processing. This evaluation aims to establish whether the agent can improve the neural network's performance (accuracy and time) by changing its structure and parameters.

7.1 NEGOTIATING AGENTS FOR JOB CONTRACT

In this section, we evaluate two aspects of the agent proposed in this research. We want to establish how our agent can adapt to different situations and the impact of using the neural network's output as part of the agent's reasoning cycle. In this sense, two experiments are presented: (i) Experiment 1's primary goal is to analyze the effectiveness of the proposed agent; (ii) Experiment 2 focuses on evaluating the efficiency of using a neural network during the agent's decision-making.

In these experiments, we employed the negotiation scenario presented in Chapter 6. This decision enables us to compare our work with the negotiation agents available in the GENIUS framework (General Environment for Negotiation with Intelligent multipurpose Usage Simulation) (LIN et al., 2014). GENIUS is the official tool used in the Automated Negotiating Agents Competition (ANAC), which helps the research community benchmark and evaluate its work (JONKER et al., 2017). GENIUS provides a set of utility functions that can be used to evaluate different negotiating agents. With this scenario, we can also compare our agent with the agent based on the Multi-Context System and Adaption Aspiration Theory (AAT).

The experiments were performed using the following specifications:

- CPU: Intel(R) Core(TM) i7-7700;
- GPU: GeForce GTX 1060 6GB;
- Memory: 16 GB DD4;
- Python 3.6;
- Sigon framework: Github repository branches aat and aat_v2 can be accessed in https://github.com/sigon-lang/sigon;

- The neural network was implemented using PyTorch 1.13.1+cu116.

7.1.1 Experiment 1

In this experiment, our primary goal is to compare the proposed agent with other implementations available in the GENIUS framework. Since GENIUS provides a negotiation scenario similar to the one presented in Chapter 6, we were able to extract 136 negotiating agent's results and compared them with our work. We used the employer/employee scenario to compare our work with these negotiating agents. Since the description of the hiring term called *working from home* was unclear, we opted to remove the variable during the benchmark. The hiring term working from home has three options. However, the meaning of each option was not described. Even though we removed this variable, GENIUS allows us to execute a benchmark without affecting the results achieved by the agents. Following, we listed the hiring terms used in this scenario:

1. Salary: (a) $2000, (b) $2500, (c) $3000, (d) $3500 or (e) $4000;
2. Car benefit: (a) yes or (b) no;
3. Permanent contract: (a) yes or (b) no;
4. Career possibilities: (a) low, (b) medium, or (c) high;
5. Full-time equivalent (FTE): (a) 24, (b) 32, or (c) 40.

In Code 7.1, we presented parts of the negotiating agent used in this scenario. In this version, we employed a negotiation strategy based on the Aspiration Adaption Theory (AAT). The neural network's output is part of a bridge-rule's body, which reduces the necessity of defining hand-crafted bridge-rules. The bridge-rules presented in Code 7.1 are the ones that will be valid and be used to determine the hiring term's urgency or aspiration levels. The contractSensor and cvSensor process the data shown in Code 7.2. The variable job contract represents the hiring terms, and the CV data represents the user's skills and experiences. The CV data will be used as an input of the trained NN deployed in the Neural Network Context (NNC). As previously mentioned, the NN predicts the average salary. We also simulated some of the user's preferences in the contexts of beliefs, desires, and intentions. This information is relevant to define some of the hiring term's urgency levels.

```

communication:
    sensor("contractSensor", "sensors.ContractSensor").
    sensor("cvSensor", "sensors.CVSensor").
    actuator("sendMessage", "actuators.SendMessage").
```

```

beliefs :
    workingHours(2).
    jobDistance(10).

desires:
    busyDailyHours(12).
    workDistance(near).
    job(programmer).

intentions :
    negotiateContract.

_nn:

_negotiation:

planner:
    plan(
        negotiateContract,
        [action(sendMessage(X,Y))],
        [_negotiation:urgency(X, Y, 10)],
        [_]
    ).

! beliefs X :- communication contractSensor(X).
! _nn X :- communication cvSensor(X).
! beliefs experience(X) :- _nn seniority(X).

! _negotiation X :- communication contractSensor(X).
! _negotiation avgSalary(X) :- _nn avgSalary(X).

! _negotiation urgency(careerPossibilities, high, 10) :- beliefs experience(senior).
! _negotiation urgency(car, no, 10) :- desires workDistance(near) & beliefs
    jobDistance(3).
! _negotiation urgency(car, yes, 10) :- beliefs jobDistance(10).

! _negotiation urgency(workingHours, 40, 10) :- desires busyDailyHours(12) &
    beliefs workingHours(2).

! _negotiation urgency(permanentContract, yes, 10) :- desires busyDailyHours(10).
```

Code 7.1 – Negotiating agent of Scenario 2

```

job_contract = {
    "salary": [2000, 2500, 3000, 3500, 4000],
    "carBenefits": ["yes", "no"],
    "permanentContract": ["yes", "no"],
    " careerPossibilities ": ["low", "medium", "high"],
    "workingHours": [8, 9, 10], represents the fte hiring term
    "negotiate": ["contract"]
}

cv_data = {'python_yn': 1,
           'spark': 1,
           'aws': 1,
           'excel': 1,
           'job_simp': 'data scientist ',
           ' seniority ': 'senior'
}
```

Code 7.2 – Hiring terms for the second experiment

Initially, the sensors create perceptions, and the NNC uses the deployed NN to predict the average salary based on the perception created by the cvSensor. After the perceptions and the average salary are available, the bridge-rules become valid, and multiple urgency levels are defined for different hiring terms. The main difference in this scenario is that the Negotiation Context (NC) can determine the urgency level with the following two strategies:

1. Directly via bridge-rules: the NC can use the information provided by different contexts, which are modeled in lines 38, 39, 40, 42, and 44;

2. Execute the strategy presented in Chapter 6 to establish the value of a hiring term and its urgency level.

After the end of the reasoning cycle, the proposal defined by the negotiating agent is the following:

1. Salary: $4000;

2. Career possibilities: high;

3. Car: yes;

4. Working Hours: 40;
5. Permanent contract: yes

To compare the proposed agent with different works, we used the 136 agents available in GENIUS and extracted the achieved utility function. Each utility function focuses on other hiring terms of this negotiation scenario, which enables us to analyze how our agent handles different situations.

7.1.1.1 Utility function 1 (mid level and senior job position)

With this utility function extracted from the GENIUS framework, our goal is to explore how the agent will handle a negotiation over a job position that requires some experience in the field. Following, we describe the first utility function and its values for this experiment:

1. S - Salary: (a) 2000 = 0, (b) 2500 = 0.25, (c) 3000 = 0.5, (d) 3500 = 0.75 or (e) 4000 = 1.0;
2. Cb - Car benefits: (a) yes = 1.0 or (b) no = 0.0;
3. Pc - Permanent contract: (a) yes = 1.0 or (b) no = 0.0;
4. Cp - Career possibilities: (a) low = 0.0, (b) medium = 0.5 or (c) high = 1.0;
5. Fte - Full-time equivalent: (a) 24 = 0.25, (b) 32 = 0.5 or (c) 40 = 1.0.

Following, we present the importance of each hiring term of the utility function:

$$\begin{aligned} U(S,Cb,Pc,Cp,Fte) = & \\ & 0.2900838579284962 * S + \\ & 0.07628234656996354 * Cb + \\ & 0.19444527477051546 * Pc + \\ & 0.05411415124288346 * Cp + \\ & 0.38507247973666225 * Fte \end{aligned} \tag{17}$$

For example, if the agent chose the following hiring terms:

- S - Salary: (d) 3500 = 0.75;
- Cb - Car benefits: (a) yes = 1.0;
- Pc - Permanent contract: (a) yes = 1.0;

- Cp - Career possibilities: (a) low = 0.0;
- Fte - Full-time equivalent: (c) 40 = 1.0.

Based on these hiring terms' values, the $U(S, Cb, Pc, Cp, Fte) = 0.873362995$ is defined as follows:

$$U(S, Cb, Pc, Cp, Fte) = \begin{array}{c} 0.2900838579284962 * 0.75 + \\ 0.07628234656996354 * 1.0 + \\ 0.19444527477051546 * 1.0 + \\ 0.05411415124288346 * 0.0 + \\ 0.38507247973666225 * 1.0 \end{array} \tag{18}$$

For this negotiation experiment and utility function, 96 agents achieved the maximum utility (1), and 25 agents did not reach the total value. 15 agents were not able to solve the negotiation. Of these 15 agents, 8 could not follow this scenario's protocol, and the remaining 7 could not construct the agent properly inside GENIUS. The results extracted from GENIUS can be accessed in https://drive.google.com/file/d/1vLe0CL0Fl7AWbA27HS69RQ2j3-AvM3N9/view?usp=share_link. The agent modeling and the utility functions used during this benchmark can be accessed https://github.com/sigon-lang/sigon/tree/aat_v2.

7.1.1.2 Utility function 2 (entry-level job position)

For the second part of experiment 1, we used other utility functions to model a different situation. The utility function used in this experiment tries to model a situation of an entry-level job position. In this case, proposing a high value for the salary is not an appropriate option. In this utility function, we removed the hiring terms unrelated to the average salary defined by the Neural Network Context (NNC). This approach allows us to explore the impact of using a neural network to process the user's skills and preferences and analyze the impact of the achieved utility function. Following, we describe the utility function we used for this last part of our first experiment:

1. S - Salary: (a) 2000 = 1, (b) 2500 = 1, (c) 3000 = 1, (d) 3500 = 0 or (e) 4000 = 0;
2. Cp - Career possibilities: (a) low = 1, (b) medium = 1 or (c) high = 0.

The second utility function is defined as follows:

$$U(S, Cp) = 0.64 * S + 0.36 * Cp \tag{19}$$

The main goal of this utility function is to model an agent to represent a user negotiating his first job contract. For this benchmark, we considered that the agent would use the cv_data presented in Code 7.3. We have two reasoning cycles based on the Code 7.1. The first one is started by the cvSensor, creating a new perception of the user's skills and experience. Bridge-rules in lines 32 and 33 are executed. In the next reasoning cycle, the contractSensor processes the data about the hiring terms, and the bridge-rules in lines 31, 35, and 36 become valid. When these bridge-rules are executed, the Negotiation Context (NC) defines the urgency level for salary and career possibilities. Based on these urgency levels, the Planner Context (PC) creates the following proposal message:

1. Salary = 2000;

2. Career possibilities = low.

```
cv_data = {
    'Industry': 'Architectural & Engineering Services',
    'Sector': 'Business Services',
    'num_comp': 0,
    'hourly': 0,
    'employer_provided': 0,
    'job_state': 'AL',
    'same_state': 0,
    'age': 20,
    'aws': 0,
    'job_simp': 'na',
    'seniority': 'na'
}
```

Code 7.3 – Example of some of the hiring terms for the third experiment

We followed the same strategy applied in the previous benchmark for this second utility function. We extracted the utility function of the 136 negotiating agents available in GENIUS. The same 15 agents did not solve the negotiation scenario. 80 agents achieved the maximum utility (1), and the remaining 41 achieved a utility lower than 1. Our approach also achieved maximum utility. The implementation for this second part can be accessed in our GitHub repository in https://github.com/sigon-lang/sigon/blob/aat_v2/experiment3-nn.py. The results extracted from GENIUS for this experiment can be accessed in https://drive.google.com/file/d/1gkeiuZ03JmBTvXel0Z19Wx6X94to6bvO/view?usp=share_link. Table 3 summarizes the results of this experiment. Even though the second utility function has fewer hiring terms than the first, the results presented in

Table 3 – Results from experiment 1

Agent	Results	Utility function 1	Utility function 2
Neural-Symbolic BDI-agent	1	1	1
Negotiating agents from GENIUS	1	96	80
	0,9 - 0,99	9	26
	0,80 - 0,89	3	7
	0,70 - 0,79	4	0
	0,60 - 0,69	3	1
	0,50 - 0,59	2	0
	0,40 - 0,49	2	5
	0,30 - 0,39	2	1
	0,20 - 0,29	0	0
	0,01 - 0,19	0	1
	0	15	15
Total		137	137

Table 3 showed that not every negotiating agent could adapt and propose the salary and career possibility that would result in the maximum utility.

7.1.2 Experiment 2

In this subsection, we compared two negotiation agents that use the Sigon framework (GELAIM et al., 2019). The first one is based on our proposed approach in Rodrigo Rodrigues Pires de Mello, Gelaim, and Silveira (2018). The second one is an implementation of the agent proposed in this research. Using the agent we presented in Rodrigo Rodrigues Pires de Mello, Gelaim, and Silveira (2018) enables us to compare our approach in the same scenario. Both versions employ Multi-Context Systems, BDI architecture, and Aspiration Adaption Theory (AAT). The main difference between these versions is how the aspiration levels or urgency goals are defined. In the first version, the agent's beliefs, desires, and intentions define the urgency goals in the Negotiation Context (NC). In the second version, we employ the Neural Network Context (NNC) to assist in determining urgency levels.

With these two approaches implemented in Sigon, our primary goal is to compare the impact of using an auxiliary context, the Neural Network Context (NNC), during the agent's decision-making and how they impact the agent's modeling, reasoning time, and conflict resolution. Since we are trying to isolate the impact of using the NNC during the agent's decision-making, we only considered the hiring term about the salary in this scenario.

Code 7.4 presents the version of the negotiating agent without the NNC. The

Communication Context (CC) is defined from lines 1 to 3, containing one contractSensor, responsible for processing information about the hiring terms, and one actuator, called sendMessage, responsible for sending the proposal of the chosen hiring terms. The contexts of beliefs, desires, and intentions are defined in lines 5, 15, and 21, respectively. The Planner Context (PC) is described in lines 26 to 31. We also omitted several other plans for variables with lower salaries. The information these contexts provide will be used to define the salary's urgency level. The bridge-rules responsible for determining the information exchange between contexts are presented in lines 33, 35, and 36.

Code 7.4 provides the initial mental state of the negotiating agent before the negotiation begins. Initially, the Negotiation Context (NC) does not have any urgency defined; therefore, its initial state is empty. After the negotiation starts, the contractSensor will process the data provided in Code 7.5. The contractSensor handles the processed data and creates a perception with the predicate *contractSensor(X)* to be added in the CC. This step validates the bridge-rule in line 31, adding X to the Beliefs Context (BC). In this reasoning cycle, *X* represents the hiring terms and their possibilities. Following, we present the job description and its options that are now in the BC:

- jobDescription(qa);
- jobDescription(programmer);
- jobDescription(teamManager);
- jobDescription(projectManager).

After the bridge-rule in line 33 finishes its execution, the body of the bridge-rule in line 35 becomes valid. This bridge-rule adds in the NC that the hiring term salary has 10 as the urgency level urgency(salary, 7000). Since the bridge-rule's body in line 36 is not valid, it will not be executed. With the urgency value of salary defined in NC, the precondition of the plan specified in line 27 becomes true. Since the intention of the negotiateContract is also valid in the Intentions Context (IC), the Planner Context (PC) can execute the plan of the action sendMessage(salary, 7000) to be performed by the actuator sendMessage. Code 7.7 presents the most relevant information on the mental state of the negotiating agent after executing the bridge-rules.

```
1 communication:
2     sensor("contractSensor", "sensors.ContractSensor").
3     actuator("sendMessage", "actuators.SendMessage").
4
5 beliefs :
6     workingHours(2).
```

```
    experience(programmer).
    experience(teamManager).
    ageGroup(young).
    yearsOfExperience(programmer, 3).
    yearsOfExperience(teamManager, 0).
    jobDistance(3).
    has(car).

desires:
    busyDailyHours(10).
    distanceFromWork(5). //wants to work close to its home
    job(programmer).
    salary(7000).

intentions :
    negotiateContract.

_negotiation:

planner:
    plan(negotiateContract,
        [action(sendMessage(X,Y))],
        [_negotiation:urgency(X, Y, 10)],
        [_]
    ).

! beliefs X :- communication contractSensor(X).

! _negotiation urgency(salary, 7000, 10) :- desires salary(7000) & beliefs
    jobDescription(programmer) & desires job(programmer).
! _negotiation urgency(salary, 12000, 10) :- desires salary(12000) & beliefs
    jobDescription(projectManager) & desires job(projectManager).
```

Code 7.4 – Initial mental state of the negotiating agent proposed in (MELLO, R. R. P. d.; GELAIM; SILVEIRA, 2018)

```

job_contract = {
      Salary: The possible values are (a) \$7,000, (b) \$12,000, or (c) \$20,000;
     "salary": [7000, 12000, 20000],
      Responsibilities given to the employer. The possible values are (a) QA, (b)
          Programmer, (c) Team Manager, or (d) Project Manager;
```

```
        "jobDescription": ["qa", "programmer", "teamManager", "projectManager"],
         (a) providing a leased company car, (b) no leased car, or (c) no
             agreement;
        "carBenefits": ["yes", "no", "noAgreement"],
         Pension benefits: The possible value for the percentage of the salary
             deposited in pension funds are (a) 0\%, (b) 10\%, (c) 20\%, or (d) no
             agreement;
        "pensionBenefits": ["0", "10", "20", "noAgreement"],
         The possible values are (a) fast promotion track (2 years), (b) slow
             promotion track (4 years), or (c) no agreement;
        " promotionPossibilities ": ["2", "4", "noAgreement"],
         This issue describes the number of working hours required by the employee
             per day The possible values are (a) 8 h, (b) 9 h, or (c) 10 h.
        "workingHours": [8, 9, 10],
        "negotiate": ["contract"]
    }
```

Code 7.5 – Hiring terms

```

beliefs :
    jobDescription(qa).
    jobDescription(programmer).
    jobDescription(teamManager).
    jobDescription(projectManager).

desires:
    job(programmer).
    salary(7000).

intentions :
    negotiateContract.

_negotiation:
    urgency(salary, 7000, 10).

planner:
    plan(negotiateContract,
        [action(sendMessage(X,Y))],
        [_negotiation:urgency(X, Y, 10)],
        [_]
    ).
```

```

! beliefs X :- communication contractSensor(X).

! _negotiation urgency(salary, 7000, 10) :- desires salary(7000) & beliefs
    jobDescription(programmer) & desires job(programmer).
! _negotiation urgency(salary, 12000, 10) :- desires salary(12000) & beliefs
    jobDescription(projectManager) & desires job(projectManager).
```

Code 7.6 – Second reasoning cycle of the negotiating agent proposed in Rodrigo Rodrigues Pires de Mello, Gelaim, and Silveira (2018)

Code 7.7 presents the agent proposed in this work. The contexts and bridge rules are similar to those shown in Chapter 6 and used in the experiment presented in 7.1.1. Our main goal with this version is to reduce the necessity of hand-crafted rules while defining the most important hiring terms. cvSensor processes the data provided in Code 7.8, which simulates the user's skills and experience. Now, the cvSensor starts processing the user's skills and experience, and then the contractSensor begins processing the data about the hiring terms and their possibilities.

To integrate the new custom sensor into different contexts, we added a new bridge-rule defined in line 29. This bridge-rule adds X in the Neural Network Context (NNC). X will be used as an input to the trained neural network in the NNC. The details about the neural network implementation are provided in Chapter 6. The neural network's primary goal is to define the average salary based on the user's skills and experience. With this result, other contexts can retrieve this information via bridge-rules. Code 7.9 presents the agent's neural network context (NNC) after the execution of this bridge-rule.

After detecting the average salary based on the perception created by the cvSensor, the negotiating agent can start processing the data of the hiring terms and their possibilities. The next steps of the reasoning cycle are presented as follows:

1. contractSensor handles the data shown in Code 7.5 and creates a new perception to be added in the Communication Context (CC);

2. The bridge-rules in lines 26 and 27 are executed. The hiring terms and their possibilities are added in the Beliefs Context (BC) and Negotiation Context (NC);

3. Since the average salary was already defined in the previous reasoning cycle, the bridge-rule in line 30 is also executed;

4. Based on the result of the bridge-rule 30, the Negotiation Context (NC) executes the implemented strategy based on AAT and detects which salary would be more relevant to the user;

This approach's details and implementation are presented in Chapter 6. Code 7.10 shows the relevant part of the agent's contexts after executing the reasoning cycle. Since the urgency of the hiring term salary is defined in the NC in line 25, the Planner Context (PC) selects the predefined plan provided in line 28. The actuator in line 4 sends the following message: *sendMessage(salary, 7000)*.

```
 1 communication:
 2     sensor("contractSensor", "sensors.ContractSensor").
 3     sensor("cvSensor", "sensors.CVSensor").
 4     actuator("sendMessage", "actuators.SendMessage").
 5
 6 beliefs :
 7     workingHours(2).
 8
 9 desires:
10     negotiateContract.
11
12 intentions :
13     negotiateContract.
14
15 // defines the avg salary based on the sensors input
16 _nn:
17
18 // implements AAT based on different information
19 _negotiation:
20
21 planner:
22     plan(
23         negotiateContract,
24         [action(sendMessage(X,Y))],
25         [_negotiation:urgency(X, Y, 10)],
26         [_]
27     ).
28 // rules responsible to create new perceptions from different knowledge sources
29 ! beliefs X :- communication contractSensor(X).
30 ! _negotiation X :- communication contractSensor(X).
31
32 ! _nn X :- communication cvSensor(X).
33 ! _negotiation avgSalary(X) :- _nn avgSalary(X).
```

Code 7.7 – Initial mental state of the proposed negotiating agent

```
    cv_data provides information about the user's  skills  and experience
    cv_data = {'python_yn': 1,
               'spark': 1,
               'aws': 1,
               'excel': 1,
               'job_simp': 'data scientist',
               'seniority': 'senior'
               }
```

Code 7.8 – Hiring terms and CV data

```
_nn:
    avgSalary(7222.35).

! _nn X :- communication cvSensor(X).
```

Code 7.9 – Second version of the negotiating agent

```
communication:
    sensor("contractSensor", "sensors.ContractSensor").
    sensor("cvSensor", "sensors.CVSensor").
    actuator("sendMessage", "actuators.SendMessage").

beliefs:
    workingHours(2).
    salary(7000).
    salary(12000).
    salary(20000).
    jobDescription(programmer).
    jobDescription(teamManager).
    jobDescription(projectManager).

desires:

intentions:
    negotiateContract.

_nn:
    avgSalary(7222.35).

// implements AAT based on different information
_negotiation:
    urgency(salary, 7000, 10).
```

```

planner:
    plan(
        negotiateContract,
        [action(sendMessage(X,Y))],
        [_negotiation:urgency(X, Y, 10)],
        [_]
    ).
// rules responsible to create new perceptions from different knowledge sources
! beliefs X :- communication contractSensor(X).
! _negotiation X :- communication contractSensor(X).

! _nn X :- communication cvSensor(X).
! _negotiation avgSalary(X) :- _nn avgSalary(X).
```

Code 7.10 – Second reasoning cycle of the proposed negotiating agent

In this second experiment, our main goal is to analyze the impact of adding a neural network execution as part of the agent's reasoning cycle. To establish the impact of the proposed approach, we performed two comparisons. Our quantitative comparison mainly revolves around two properties:

1. The amount of time used to perform a reasoning cycle;
2. The decision provided by each agent.

We executed the first comparison 50 times and collected: (i) the total amount of time to perform a reasoning cycle and the decision performed by each agent. Figure 16 presents the elapsed time for each execution. Following, we define the main feature of each version:

1. Version 1 (red triangle) only uses the Negotiation Context (NC) to solve the conflict;
2. Version 2 (green square) employs the Negotiation Context (NC) and Neural Network Context (NNC);
3. Version 3 (blue circle) also employs the NC and NNC. However, this version uses the neural network in every reasoning cycle.

This figure shows that version 2 improves its execution time after the first execution. In version 2, the information about the CV does not change during negotiation. Therefore, it can be executed before the negotiation starts. Thus, the neural network is triggered only one time. The average time and standard deviation for version 1 are

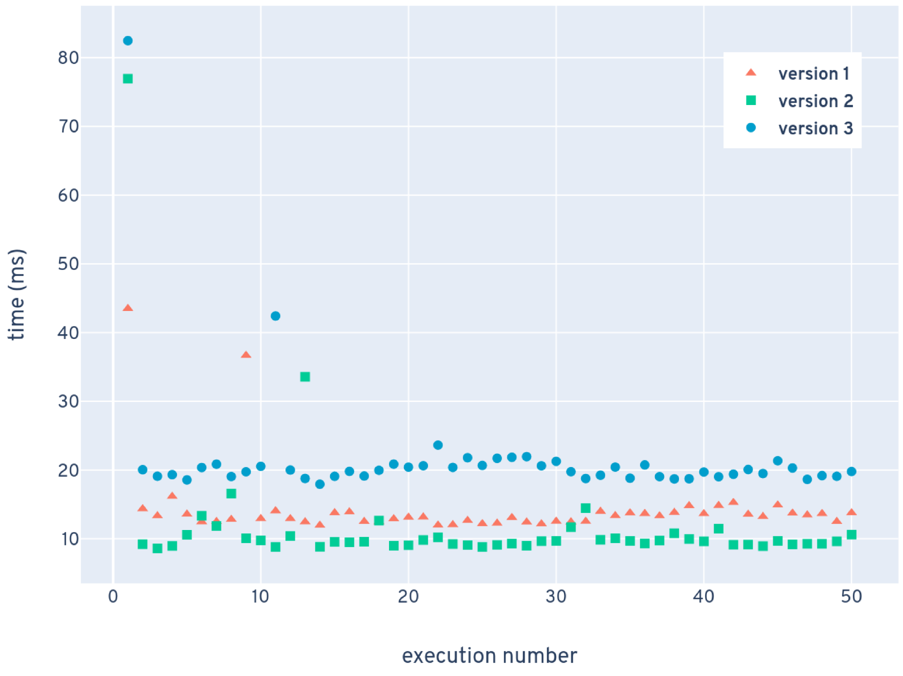


Figure 16 – Comparison between version 1, version 2, and version 3.

14.35 and 5.43, respectively. For version 2, the average time and standard deviation are 11.85 and 10.08, respectively. In summary, the second version performed better in 44 executions. Since adding more bridge-rules adds the necessity of checking whether the information is valid in the context, it is reasonable to state that this is one of the leading causes of the higher reasoning cycle time achieved in version 1.

Based on the implementation of the agents in Experiment 2, we also explored how using the neural network in every reasoning cycle would affect the execution time. Figure 16 shows the execution time of version 1 (with the NC), version 2 (with NC and NNC), and version 3 (NC and NNC), but using the neural network in every reasoning cycle. It is possible to notice that version 1 and version 2 performed better than version 3. The average time and standard deviation for version 3 are 21.67 and 9.40.

Even though every version provides the same output, defining when the neural network should execute is crucial. The neural network is executed when a bridge-rule that contains the Neural Network Context (NNC) as the head of the rule becomes valid. In versions 2 and 3, the neural network is activated when the custom sensor processes new information and creates a perception. For instance, since the user's skills did not change during the negotiation, triggering the neural network in every reasoning cycle was unnecessary. However, in scenarios with limited time in which the NNC is part of the head of a bridge-rule, it is necessary to control when the bridge-rule becomes valid. In this sense, it is crucial to establish the trade-off between obtaining a new output from the neural network and the elapsed time to get this new output.

7.1.3 Discussion

We divided the negotiating agents' experiments activity into two parts: (i) a comparison with 136 negotiating agents available in the GENIUS framework; and (ii) a comparison with the theoretical version presented in our previous work (MELLO, R. R. P. d.; GELAIM; SILVEIRA, 2018). In the first part of these experiments, we focused on evaluating the utility functions achieved in two situations. The main goal was to analyze how the agent adapted when different conditions were presented. For instance, in the first situation was required to propose a high salary to achieve a high utility value. On the other hand, the second situation represented an entry-level job position, which requires the agent to propose a low salary. In the last experiment, we compared three versions of the negotiating agent. The first version was implemented based on our previous work (MELLO, R. R. P. d.; GELAIM; SILVEIRA, 2018), which also uses MCS and Aspiration Adaption Theory (AAT). The second version is an implementation of the agent proposed in this work. This version is also based on MCS and AAT; however, it uses the information provided by the NNC to define the most urgent negotiation variable. The third version is similar to the second one, and the main difference is that the neural network is executed in every reasoning cycle.

Our results in the first experiment showed that the negotiating agent adapted to different situations and achieved the maximum utility value in both cases, matching the best agents competing in the ANAC competition. The results of the second experiment showed that the second version performed better (lower time to solve the conflict) and provided the same decision as versions 1 and 3. The last experiment also showed that executing the neural network in every reasoning cycle could not be adequate in scenarios that require rapid responses. In this sense, it is necessary to establish the trade-off between obtaining a new output from the neural network and the elapsed time to get this new output. Considering these results, the proposed agent is suitable for solving conflicts while adapting to different situations without adding more hand-crafted rules.

7.2 EVALUATION OF THE NEURAL-SYMBOLIC AGENT FOR MALWARE DETECTION

In the previous evaluation, we analyzed and evaluated the impact of employing a neural network's outputs in the agent's decision-making. We focused on finding whether a neural network could mitigate some of the symbolic method's limitations. In this current Section, we aim to explore how the agent's decision-making can improve the neural network performance in a real-world scenario. One of our primary goals is to analyze the integration method's impact and how it can affect the neural network's performance.

In these experiments, we employed a similar agent to the one presented in

Section 6.2. The agent for malware detection uses the same CNN and processes the EMBER dataset. The agent also establishes which training method the CNN should employ: conventional training, fine-tuning, or feature extraction. In these experiments, we assume that employing fine-tuning and feature extraction results in better accuracy. Although the training method could require more time, a new layer is added to the NN every time a feature extraction is performed. Considering these, the agent's reasoning cycle establishes when it is necessary to use one of these strategies. In Section 6.2, the agent employed a custom sensor and contexts to establish the computational resources available and define desires and intentions. In these experiments, the agent already has the required desires and intentions to train the neural network. Code 7.11 recapitulates the agent's contexts and bridge-rules employed during these experiments.

```

communication:
    sensor("dataSensor", "sensors.DataSensor").
    actuator("setOperation", "actuators.SetOperation").

beliefs :
    time(available ) .
    resources(high).

desires:
    improveAccuracy.

intentions :
    improveAccuracy.


_metrics:
    performance(high).
_nn:

planner:
    plan(
        improveAccuracy,
        [action(setOperation(train)) ],
        [_metrics:performance(low), beliefs :time(available ) ,
            beliefs :resources(high)],
        [_]
    ) .
    plan(
        improveAccuracy,
        [action(setOperation(fineTuning))],
        [_metrics:performance(medium)],
        [_]
    ).
    plan(
        improveAccuracy,
        [action(setOperation(featureExtraction))],
        [_metrics:performance(high)],
        [_]
    ).

! _nn setOperation(train) :- communication sense(operation(train)).

! _nn execute(X) :- communication dataSensor(X). // executes an operation with the
    current available data.

! _metrics history_trainings (X,Y) :- _nn history_trainings (X, Y). // X = acc value
    and Y = loss value
```

Code 7.11 – Neural-symbolic agent used for the experiments

As mentioned in Section 6.2, we added Continual Learning (CL) properties by processing the dataset monthly. We believe this approach also enables us to model a scenario with dynamic properties. Considering these, we start the CNN training with data from January, and after the training is complete, we process the data from February and so on. After the CNN is trained with the data from the whole year, we test the CNN with the data from the entire year. This approach enables us to analyze how the transfer learning techniques and the proposed agent improve the results. This experiment is divided into two different scenarios. Both scenarios compare the proposed neural-symbolic agent with other approaches that only use a neural network. Following, we define these two scenarios:

1. Scenario 1: uses 10 epochs to train the neural network and execute fine-tuning. It uses 15 epochs to perform feature extraction.

2. Scenario 2: uses 30 epochs for training, fine-tuning, and feature extraction. The main difference is that we use an approach called early stopping. This approach establishes when the training, fine-tuning, and feature extraction should stop, which mitigates overfitting.

The experiments were performed using the following specifications:

- CPU: AMD Ryzen 9 7900;

- GPU: GeForce GTX 4060 8GB;
- Memory: 32 GB DDR5;
- Python 3.11.5;
- Sigon framework: https://github.com/sigon-lang/sigon branches malware;
- The neural network was implemented using KERAS API (CHOLLET, François et al., 2015).

7.2.1 Scenario 1

In this scenario, we developed three versions to detect malicious software. These versions are trained using the EMBER dataset. The first version trains the model without any fine-tuning technique during the processing of each month. The second version uses feature extraction and fine-tuning to process the data from each month. Since we assumed that employing fine-tuning and feature extraction could result in better accuracy, we chose version 2 as the baseline version in this scenario. The third version, which represents the neural-symbolic BDI-agent proposed in this work, combines the previous versions. The main difference is that the neural-symbolic agent uses its reasoning cycle to define which technique should be employed. We executed each version 10 times and gathered data about the NN's accuracy and loss during training and testing and the required time to finish the training.

Figure 17 presents the mean accuracy achieved in the test dataset from processing the twelve months of EMBER. We executed each version 10 times. In Figure 17, it is possible to notice that the neural-symbolic agent (version 3) achieved a median of 0.814. The fine-tuning and feature extraction (version 2) reached 0.839. Version 1, which only uses training during the experiment, achieved the lowest value with the highest interquartile range. In this case, we believe that the catastrophic forgetting impacted the results of version 1. The interquartile range of the neural-symbolic version is greater than version 2. We claim this result is due to the rule we employed to define when the neural network's performance is considered high, medium, or low. We also noticed that not every execution among the ten executions achieved the same sequence of action. For instance, in a particular execution, the agent used feature extraction more times than fine-tuning.

Since we are interested in data that arrives in different months, we analyzed the accuracy achieved in the test dataset of each month separately and the time elapsed to train each version. In this sense, Figure 18 presents the results for each month of the ten executions. It is possible to notice that the NN's performance, when only the train method was employed, was lower in the first months. This information corroborates our previous claim that this method was more impacted by catastrophic forgetting.

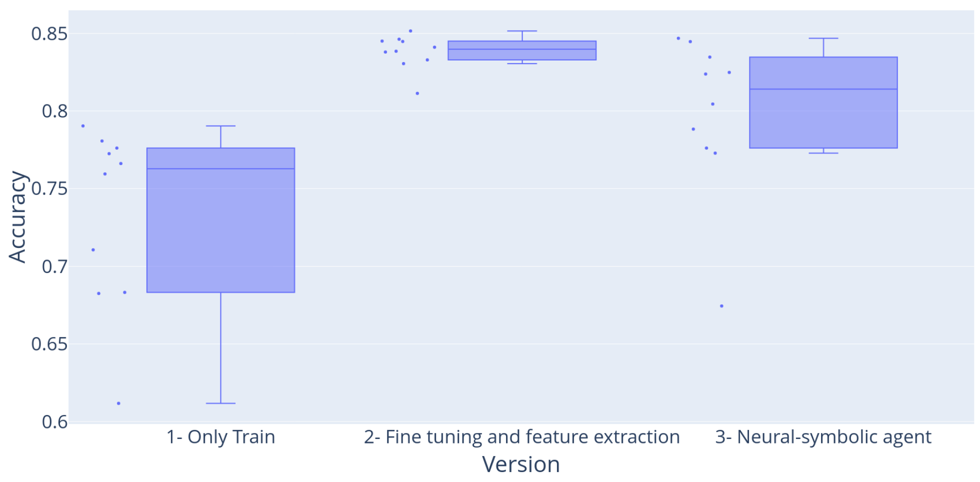


Figure 17 – Scenario 1: Mean accuracy from 12 months of 10 executions

For instance, when we tested the trained CNN after processing January, we noticed that the CNN achieved an accuracy of 93% in the test dataset. However, Figure 18 showed that the accuracy dropped to 65.1% when we executed the resulting CNN after processing twelve months of the EMBER dataset. When considering the results from the three versions with software from December, we believe they achieved similar results because the agents processed the training dataset of software from December in the last iteration. Another relevant aspect is that despite the neural-symbolic agent achieving a similar result in the mean accuracy, the neural-symbolic agent performance was worse in 10 months.

Figure 19 presents the time elapsed to complete the training. Even though the neural-symbolic agent accuracy was worse than version 2, the time to execute the training was lower than version 2. It is also possible to notice that version that only used the training method was the fastest version. Although we assumed that training could require more time, it is important to notice that every time a feature extraction is performed, a new layer is added into the previous NN.

7.2.2 Scenario 2

In the second scenario of this experiment, our goal is to analyze the impact of adding a new strategy during the model's training and establish how it can affect the accuracy. This strategy revolves around a mechanism provided by the KERAS API called Early Stopping. KERAS provides a parameter called patience, which controls when the training should stop. The parameter *patience* sets the number of epochs with no improvement, after which training will be stopped. In this scenario, we followed the same approach as the previous scenario and compared the neural-symbolic agent with the two versions that only use training or a transfer learning technique.

The value for the *patience* of versions 1 and 2 is always three. With the neural-

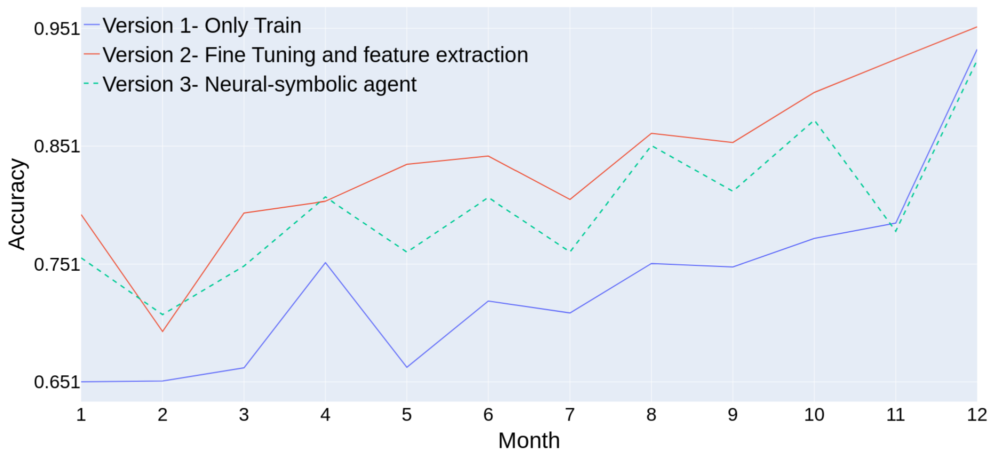


Figure 18 – Scenario 1: Mean accuracy for each months of 10 executions

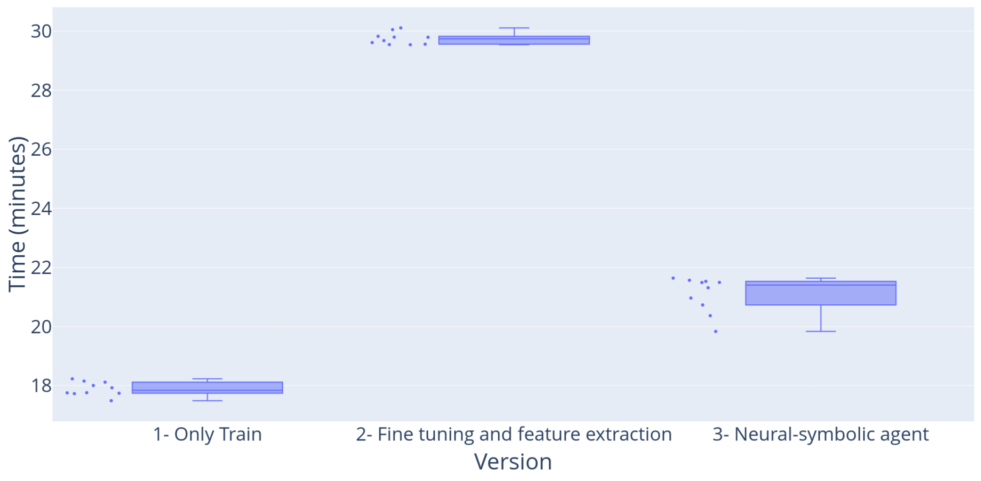


Figure 19 – Scenario 1: Elapsed time to train

symbolic BDI-agent, we define a new bridge-rule to set a new value for *patience* dynamically. This strategy is based on the accuracy achieved when the data from the previous month was processed. The strategy is modeled in the Metrics Context (MC) as follows:

1. Decrement by two the *patience* when accuracy is high;
2. Increment *patience* when the accuracy from the previous training is defined as medium;
3. Increment by two the *patience* when accuracy is low;
4. The maximum value of the *patience* parameter is 8. Thus, the parameter can only increase until it reaches the value 0 or decrease until it reaches zero.

The neural-symbolic agent used in this second scenario is similar to the one presented in scenario 1; the only difference is the addition of a new bridge rule. Listing

7.12 shows the new bridge-rule added in this version. This bridge-rule is responsible for integrating the information about the *patience*. Based on CNN's accuracy information, the MC applies the previously presented strategy and integrates this decision with the Neural Network Context (NNC). It is essential to state that this bridge-rule is always executed before the bridge-rule responsible for starting the training.

```
! _nn setParameters(X) :– _metrics findParameters(X).
```

Code 7.12 – Bridge-rule added in the second scenario of Experiment 1

Figure 20 presents the mean accuracy achieved in scenario 2. The neural-symbolic agent (version 3) achieved a median of 0.844, similar to the one that employs fine-tuning and feature extraction (version 2) in every training, which achieved 0.846. Figure 21 shows the mean accuracy from the ten executions of each month. In this version, the neural-symbolic agent achieved results closer to those of the version that uses fine-tuning and feature extraction. Figure 22 presents the elapsed time to train these versions. In Figure 22, it is possible to notice the impact of the new bridge-rule, which plays a vital role in reducing the required time to train the CNN and improving the accuracy. Another significant result is that using the *patience* parameter negatively affected version 1. We believe this could be because more epochs were required to achieve accuracy, with no improvement in version 1.

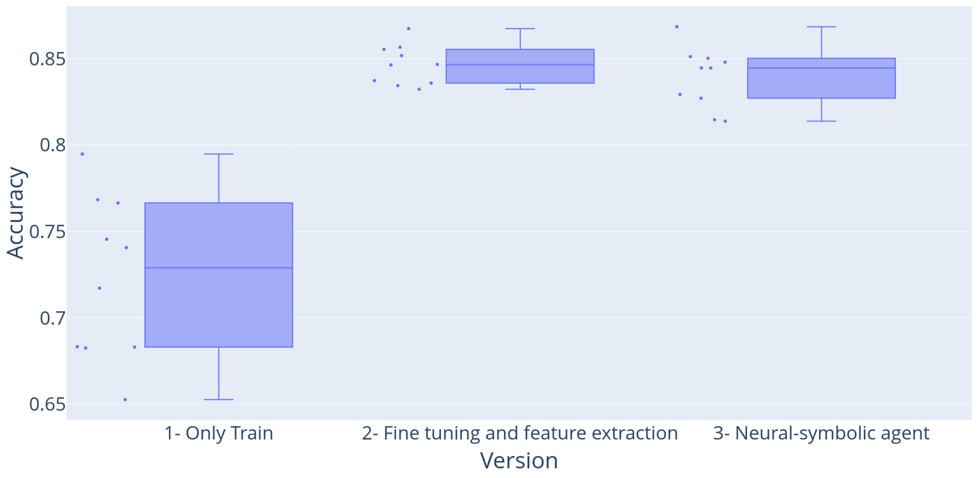


Figure 20 – Scenario 2: Mean accuracy from 12 months of 10 executions

7.2.3 Discussion

This section presented an experiment with two scenarios with agents and neural networks to detect malicious software. We used a dataset called EMBER (ANDERSON; ROTH, 2018), which includes real applications from 2018. We also modeled this scenario employing Continual Learning (CL) properties, in which all the data was not

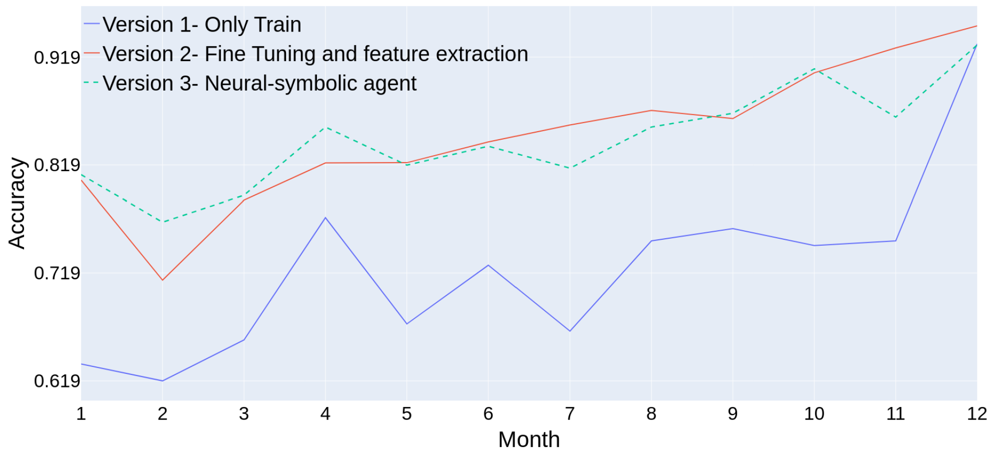


Figure 21 – Scenario 2: Mean accuracy for each months of 10 executions

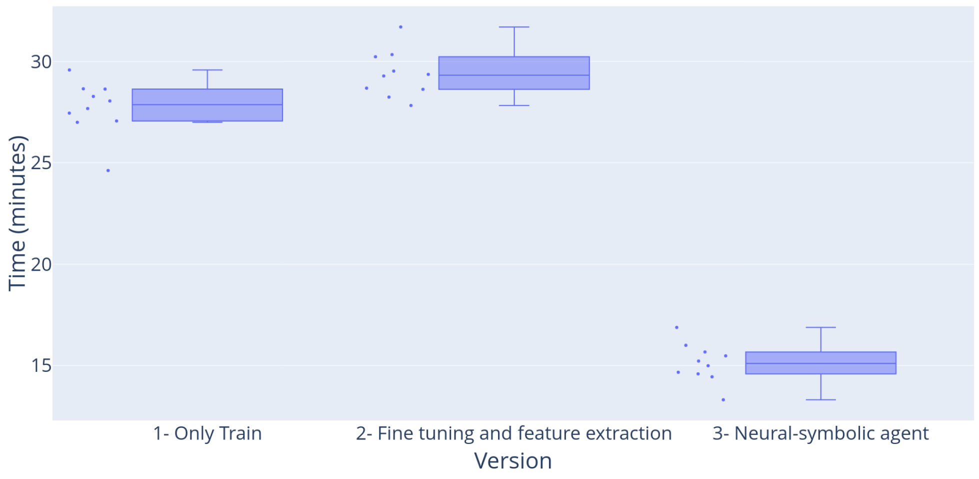


Figure 22 – Scenario 2: Elapsed time to train

available at the same time. In this sense, we train the model with data from different months, simulating a real-world setting in which new data arrives at the end of each month. Our main goal was to analyze whether the proposed neural-symbolic agent can improve the neural network's performance. In this experiment, we focused on exploring the model's accuracy achieved with the EMBER dataset and the time elapsed to train the model.

We separated this experiment into scenarios 1 and 2. This decision enabled us to analyze the impact of adding new bridge-rules in the agent's reasoning cycle. Each scenario contained three different versions to detect malicious software. The first two versions used training and combined feature extraction with fine-tuning. The third version was the neural-symbolic BDI-agent proposed in this research. The agent's reasoning cycle is responsible for defining when it is necessary to use one of these strategies. The decisions about which approach to use are based on the CNN's ac-

Table 4 – Main results from Scenario 1 and 2.

	Scenario 1		Scenario 2	
Version	time (min)	accuracy	time (min)	accuracy (%)
Train	17.89	76.2	27.70	72.8
Fine-tuning and feature extraction	29.75	83.9	29.38	84.6
Neural-symbolic agent	21.09	81.4	15.12	84.4

curacy from previous training. Each strategy has a different time to process the data. In this sense, the neural-symbolic agent's strategy determines when each approach should be used.

In scenario 1, the neural-symbolic agent employed a bridge-rule that defines whether the model achieved a high, medium, or low accuracy. Based on the results of this bridge-rule, the agent triggers a plan that establishes which strategy it will use to process the data from the next month. Scenario 2 follows a similar setting. The main difference is that we added a new bridge-rule to control the amount of epochs the model should wait until it finishes the training phase. Considering these settings, we analyzed how these rules impacted the resulting model and compared them with versions that only use the training, feature extraction, and fine-tuning methods.

Table 4 summarizes the main results from both scenarios. This table presents the mean value from time and accuracy from the ten executions. The accuracy in this table is defined by the mean from the total accuracy achieved with the test dataset from the 12 months. Even though the neural-symbolic agent achieved lower results than the version that uses fine-tuning and feature extraction, it required 29% less time to train the model. The version that only uses the train approach did not achieve relevant results. We believe that the reason is that this version was more affected by the catastrophic forgetting. Scenario 2 showed that the new bridge-rule added to control when the model should stop the training phase positively affected the neural-symbolic agent. We were able to improve the model accuracy and reduce the required amount of time to finish the training phase. The neural-symbolic agent was 48% faster than the version using fine-tuning and feature extraction. The accuracy achieved each month in the third version was similar to that achieved in the version with fine-tuning and feature extraction.

The scenarios modeled in this Chapter showed that the proposed agent can execute in a real-world and dynamic scenario where data arrives at different moments. Even though we implemented straightforward mechanisms to control the strategy used during training, the results achieved in scenarios 1 and 2 showed that the neural-symbolic agent improved the neural network's performance, more precisely, accuracy and time. The new bridge-rule added in scenario 2 also positively impacted the results without affecting the required time to finish the training. We believe a promising way is to combine more complex strategies to train the model, which could lead to better results in the model's performance.

7.3 VALIDITY THREATS AND LIMITATIONS

The evaluation presented in this chapter has threats and limitations that could impact the results of this research. Our results could be affected by some decisions and how we implemented some core aspects of the experiments and scenarios.

In the evaluations presented in Section 7.1, we believe the scenario employed did not represent a complex environment. In both experiments, the negotiation agent solved the conflict without the necessity of learning or adapting during the conflict resolution or after a certain amount of encounters. These characteristics affect how the agent adapts to different and complex situations. Another relevant threat is that we compared agents implemented with different frameworks. For instance, our agent was developed in Sigon, and the other agents were implemented in GENIUS. To mitigate this threat, we compared these agents' achieved utility values. In this sense, we believe it is still necessary to analyze other relevant aspects, such as the time required to reach an agreement and the computational resources these agents use.

To mitigate the issues of scenario complexity, we performed a new experiment focusing on a real-world scenario for malware detection. The dataset employed is based on real-world applications. The validity threats of the experiment presented in Section 7.2 can be divided into two decisions. The first relates to the approach we adopted regarding the neural network model. We handle the model's input as an image. Even though this approach was employed in Ghouti and Imam (2020), the results we achieved with the neural-symbolic agent and CNN may need to be reproducible or be affected when different architectures are used. The second decision is related to the rules and plans we employed in the neural-symbolic agent. The values to define whether the NN's performance was high, medium, or low and the patience variable could lead to some bias.

Even though we focused on exploring the impact of using the proposed agent model, we believe it is necessary to collect more metrics of the employed neural network. In the experiments of Section 7.2, collecting more metrics (i.e., precision, recall, Area Under the Curve (AUC), and confusion matrix) could increase the validity of our results. In the last experiment, we did not fully explore the impact of catastrophic forgetting. For instance, we could have extracted the accuracy from the resulting NNs of each month, tested them, and compared them with the resulting NN after processing the data from the twelve months. Another crucial aspect is that we did not consider the required time to model the neural network and integrate it with the neural-symbolic agent proposed in our work.

8 ADDITIONAL RELATED WORKS

In Chapter 4, we presented a Systematic Literature Mapping (SLM) about how studies combine neural networks and intelligent agents to solve different problems. We believe exploring how recent works model, develop, evaluate, and deploy neural-symbolic agents in real-world scenarios is necessary. In this sense, we perform an SLM focusing on neural-symbolic agents and how our work's proposed neural-symbolic agent and experiments are related to the most recent studies. Section 8.1 presents the protocol, research questions, search strings, and main findings of the SLM. Section 8.3 presents related works that directly impacted our research.

8.1 SYSTEMATIC LITERATURE MAPPING (SLM) OF NEURAL-SYMBOLIC AGENTS

In this section, we followed a similar approach presented in Chapter 4. This chapter described an SLM we previously executed in our research. The SLM presented in Chapter 4 was the starting point of our research, which focused on exploring how different works combine neural networks and intelligent agents. We proposed a neural-symbolic agent and executed two experiments based on these findings. Considering that, this current SLM presents how different works model and develop neural-symbolic agents. The main goal of this SLM is to find recent works and how they are related to our research.

The methodology employed in this SLM is the same as presented in our previous SLM in Chapter 4. We followed the Kitchenham and Charters (2007) guidelines, formed by the following discrete activities: planning, conducting, and reporting. We used the five criteria Population, Intervention, Comparison, Outcomes, and Context (PICOC) described in Petticrew and Roberts (2008) to establish the research questions. Following, we present the four criteria:

- P (population or problem): which classes of agents were employed;
- I (intervention or interest): how neural-symbolic and agents are combined;
- O (Outcome/results): how these works evaluate the proposed agents;
- C (Context): how these works implement the proposed agents and which tools are used.

The research questions are defined as follows:

- RQ1: How neural-symbolic systems are modeled?
- RQ2: Which field of study are these works focused?

Table 5 – Inclusion and exclusion criteria

Inclusion (I)	Exclusion (E)
published between 2019 to 2024	published before 2019
written in English	not written in English
available to download	unavailable to be read
proposes an agent based on neural-symbolic properties	does not use agents
present a qualitative or quantitative evaluation	does not use any neural-symbolic property
published in conference or journal	do not present quantitative or qualitative evaluation
primary studies	secondary or tertiary studies

- RQ3: Which tools are being used to develop these agents?
- RQ4: How do these works evaluate the proposed agents?

The search string used in this SLM was based on two words: neural-symbolic and agent. Initially, we executed the search with these two words in the ACM library. However, we noticed that the search did not return any results. To overcome this limitation, we removed the part referencing the agent and executed the search string using only words similar to neural-symbolic. Following, we present the final version of the search strings:

- Scopus: TITLE-ABS-KEY (("neural-symbolic" OR "neuro-symbolic" OR "neural symbolic" OR "neuro symbolic") AND ("agent"))
- ACM: [[Title: "neural-symbolic"] OR [Title: "neuro-symbolic"] OR [Title: "neural symbolic"] OR [Title: "neuro symbolic"]] AND [[Abstract: "neural-symbolic"] OR [Abstract: "neuro-symbolic"] OR [Abstract: "neural symbolic"] OR [Abstract: "neuro symbolic"]] AND [[Keywords: "neural-symbolic"] OR [Keywords: "neuro-symbolic"] OR [Keywords: "neural symbolic"] OR [Keywords: "neuro symbolic"]]

Figure 23 presents the steps to execute this SLM. Since we removed the word related to the agent in the ACM search string, many works were removed during step 2. We encountered a few challenges that we faced in our previous SLM: some works were unavailable, and we also noticed that when analyzing the whole paper, some of these works did not fit the criteria.

To access the Data extraction form, the reader could https://docs.google.com/spreadsheets/d/1bc9vt00pOewaQI19E1igRNuCMpkwu1OgaHGMerg2-Tk/edit?usp=sharing.

8.2 RESULTS FROM THE DATA ANALYSES

In this section, we present the main findings from the SLM. The findings were based on answering the research questions previously presented. To answer these research questions, we created groups for each research question. This approach

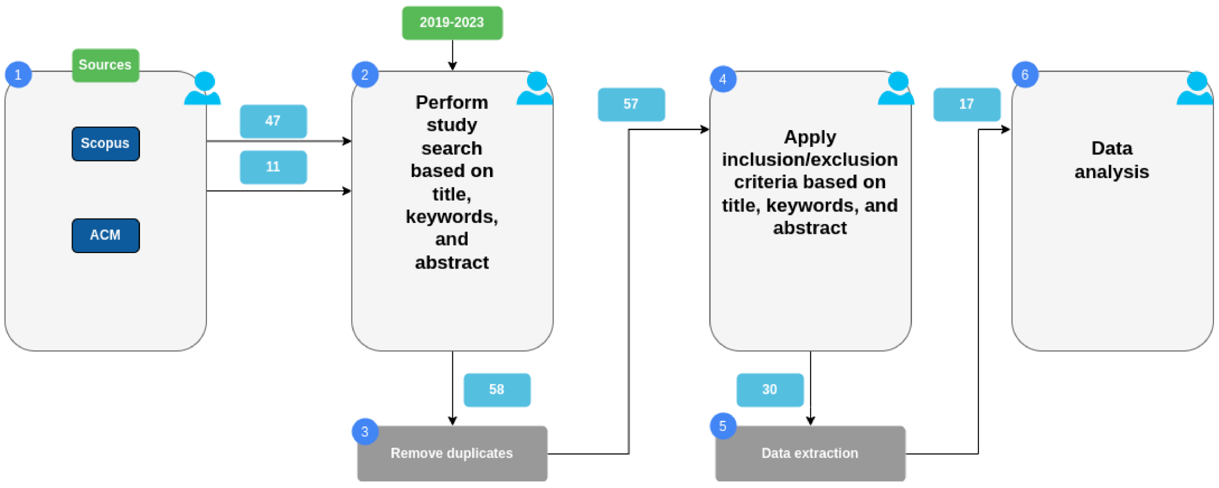

2019-2023
1
Sources
Scopus
ACM
47
11
2
Perform study search based on title, keywords, and abstract
57
4
Apply inclusion/exclusion criteria based on title, keywords, and abstract
17
6
Data analysis
58
3
Remove duplicates
30
5
Data extraction

Figure 23 – Neural-symbolic agents Systematic literature mapping executed steps.

enables us to search for patterns of how these works model, develop and evaluate the proposed neural-symbolic systems. Based on these patterns, we can compare our research with similar works.

RQ1 explores how these works model the interaction between different components of these neural-symbolic systems. For instance, some neural-symbolic systems employ the connectionist method's output as part of the agent's planner. In this RQ1, we employed the taxonomy presented in Section 3.6. This taxonomy describes a set of methodologies for developing neural-symbolic agents. Based on these properties, we defined the groups of RQ1 as follows:

- learning for reasoning: main goal is to integrate the benefits of the neural networks to assist in finding solutions;
- reasoning for learning: uses neural systems' learning capabilities to map functions and integrates the advantages of symbolic systems (symbolic knowledge) into the learning process to enhance the learning ability of neural systems;
- learning-reasoning: both neural and symbolic systems play equal roles and work together in a mutually beneficial way.

Based on the findings of RQ1, we can classify these works with the neural-symbolic taxonomy presented in Section 3.6. This taxonomy classifies works based on how they integrate both methods. In this sense, we classify these works as follows: (i) learning for reasoning: 7; (ii) reasoning for learning: 3; and (iii) learning-reasoning: 7. In our work, we claim that we explored the learning for reasoning and reasoning for learning. In Section 7.1, we explored how to employ the NN's output to mitigate the necessity of hard-coded and hand-crafted rules. In Section 7.2, we analyzed whether the proposed agent improves the neural network's performance.

Table 6 presents in which field of study these systems are deployed. Planning, natural language processing, and reinforcement learning represent the field of study with the most work. The focus on reinforcement learning follows one of the results of our first SLM, in which many studies explore the integration of neural networks and learning agents. We believe that natural language processing interest is due to the success of GPT models in the last few years. In the planning field of study, we believe the main goal is to improve the modeling of dynamic plans. Asai and Muise (2020) defend that obtaining the descriptive action models from the raw observations with minimal human interference is the next key milestone for expanding the scope of applying Automated Planning to the raw unstructured inputs.

Field of study	Quantity
concurrent stochastic games	1
ethical	1
learn policy rules	4
planning	6
natural language processing	5

Table 6 – Distribution of the field of study.

Figure 24 shows the tools for developing and evaluating the proposed neural-symbolic agents. Unlike the previous results, we could not notice any pattern related to which tools are the most used. These results show that the community could benefit from a framework for developing neural-symbolic systems. Table 7 presents the methods employed in the evaluation. We were able to group this evaluation into three main categories: (i) only executes: represents the studies that only present an execution of the proposed agent; (ii) different versions: represents studies that implemented different versions of the agent and compare each other, without referencing different studies from the literature; and (iii) different researches compare the proposed agent with several studies from the literature. The results presented in Table 7 also follow the approach we adopted in the executed experiments, in which we compare our agent in different scenarios with works from the literature.

In this SLM, we faced similar validity threats from the previous SLM. Since we tried to classify these works in groups, some classifications could be incorrect. We

Table 7 – Results from the method used to evaluate the proposed agents.

Evaluation	Quantity
Only executes	2
Different versions	8
Different researches	7

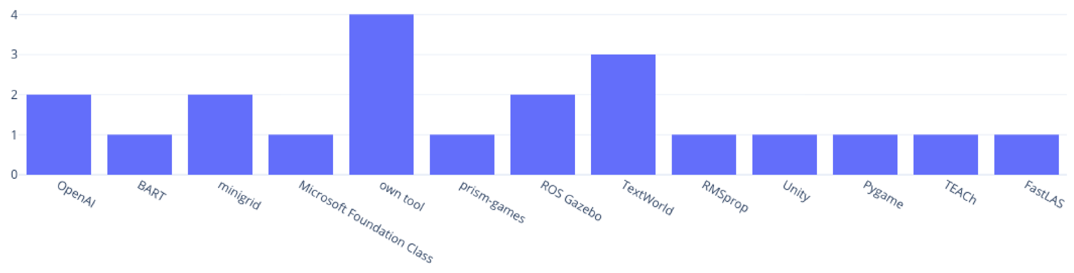


Figure 24 – Tools employed to develop neural-symbolic agents.

executed this SLM with one researcher, which could lead to incorrect interpretation and bias. Some of the works returned in the search strings were also unavailable, reducing the total amount of papers we analyzed. Based on the results presented in this section, we define how our research is situated in the previously presented research questions. With the proposed neural-symbolic agent, scenario, and experiment executed in this research, we summarize our work as follows:

1. RQ1:
    - learning for reasoning: Uses the Neural Network Context (NNC) as part of the agent's plans preconditions and to model custom bridge-rules;
    - reasoning for learning: Uses the neural-symbolic agent's decision-making to change the neural network's parameters.
2. RQ2: even though we did not explicitly focus on planning, the scenarios about negotiation and malware detection integrate the neural network's output with the agent's plans preconditions;
3. RQ3: uses own tool to model the agent;
4. RQ5: evaluates the proposed agent with different works and versions.

8.2.1 Discussion

In this subsection, we explore in more depth three studies that complement our work and indicate future research we could pursue. We then present a more detailed review of the selected studies and compare them with our research.

Chaput et al. (2021) present a novel approach to how different types of agents could employ symbolic and connectionist methods to improve their ability to behave ethically. This approach proposes using symbolic judging agents to evaluate the ethics of learning agents' behavior and improve how they act in dynamic multi-agent environments. Judging agents are given a set of symbolic rules to reason over actions undertaken in the environment. Judging agents use these rules to compute a judgment (e.g., "moral" or "immoral") and provide it as feedback for learning agents. Learning agents learn a policy to solve the task, which may imply an ethical dilemma, by learning to select the correct action and perform it in the environment according to this feedback. The learned policy reflects the ethical considerations embedded in the symbolic rules. The proposed approach was applied to an energy distribution problem in a Smart Grid simulator context. The experiments and results show the ability of learning agents to correctly adapt their behaviors to comply with the judging gents' rules, including when rules evolve.

Moon (2021) proposes a plugin framework for neural-symbolic agents in which a multi-agent system can be involved in task planning in a broad range of areas by combining symbolic and connectionist approaches. A planning domain definition language-based planning algorithm, a symbolic approach, and the cooperative–competitive reinforcement learning algorithm, a connectionist approach, were utilized. The proposed plugin framework integrates middle-ware between a hierarchical task planning framework and cooperative–competitive reinforcement learning. Actions generated through a planner are delivered to the action controller through plan execution. Agents are removed through the reinforcement learning system if actions requiring a quick response are received. All other actions are applied with motion planning based on motion algorithms. The proposed approach simulated an experiment using ten unmanned surface vehicles in which the given tasks were successfully executed.

Basu et al. (2022) present a hybrid neural-symbolic architecture for Text-based games (TBGs) that uses symbolic reasoning along with the neural Reinforcement Learning (RL) model. TBGs provide a challenging environment where an agent can observe the game's current state and act in the world using only the modality of text (BASU et al., 2022). The authors defend that TBGs require reinforcement learning (RL) agents to combine natural language understanding with reasoning. The proposed architecture employs inductive logic programming (ILP) to learn the symbolic rules (policies) as default theory with exceptions and is represented in the form of an Answer-Set Program (ASP) that allows performing non-monotonic reasoning in the partially observable game environment. The approach employs WordNet as an external knowledge source to lift the learned rules to their generalized versions. These rules are learned online and applied with an ASP solver to predict an action for the agent. The neural part is responsible for exploring the environment and is used in scenarios where a symbolic agent fails to provide an action (due to a lack of learned rules). The architecture was tested on a TextWorld-Commonsense framework (MURUGESAN et al., 2021; BASU et al., 2022). The authors presented experiments in which the proposed approach outperforms state-of-the-art agents.

Chaput et al. (2021) and Moon (2021) focused on exploring how the output of symbolic methods could be employed as an input of connectionist methods. Based on the taxonomy presented in Section 3.6, these works could be classified in the reasoning for learning category. Basu et al. (2022) proposed a hybrid approach that could be classified as learning-reasoning, in which the agent chose between symbolic and connectionist methods during decision-making. As previously mentioned, we focused on the learning for reasoning and reasoning for learning categories. However, we believe it is possible to model a similar approach as the one proposed in Basu et al. (2022). The agent's designer could employ the plan's preconditions to verify whether information is valid in the agent's symbolic or connectionist contexts. Definition 10 in Section 5.2 showed how this bridge-rule is implemented.

Regarding the integration method, Chaput et al. (2021) defend that using agents with different capabilities (symbolic or connectionist) could enable these agents to evolve or update separately, benefiting from co-construction processes. Moon (2021) also proposed an approach of neural-symbolic agents for Multi-Agent Systems (MAS). However, the approach is similar to the one proposed in Basu et al. (2022), in which every agent employs the same decision-making. Even though we focused on only one agent, the proposed integration method enables us to model agents in which the symbolic and connectionist capabilities could also evolve separately. To be more precise, Multi-Context Systems facilitate the modeling of custom contexts without changing the existing ones. We believe the proposed integration method provides relevant features, such as modularity and flexibility, enabling the development of generic neural-symbolic agents.

The previously presented works also expanded on future works we could pursue. Moon (2021) and Basu et al. (2022) simulated the proposed approaches with several relevant tools, such as Robot Operating System(ROS), OpenAI GYM, and TextWorld-Commonsense framework. Considering this, we aim to use these tools to compare our work with different studies. Another path that we intend to follow is related to dynamic planning. Similar to what was presented in Moon (2021) and Basu et al. (2022), we intend to explore how the connectionist context could create plans during the agent's execution.

8.3 FURTHER RELATED WORKS

In this section, we present several works that were not retrieved during the Systematic Literature Mappings' execution but were crucial during the development of our research. We show which studies were the baseline of our research, how we adapted some approaches, and which approaches are similar to ours.

The neural-symbolic agent proposed in this research is implemented using the

Sigon framework (GELAIM et al., 2019). Sigon was employed to model agents in scenarios about situation awareness, perceptions, and negotiation (GELAIM, 2021). Sigon's most relevant details are presented in Chapter 3, and its implementation in JAVA can be accessed in https://github.com/sigon-lang/sigon-lang. Even though the Sigon framework already has a JAVA version, we implemented a new version using Python. This decision facilitates the usage of the most relevant Machine Learning libraries and frameworks. We also focused on improving support for defining customs sensors, contexts, and bridge-rules without changing existing contexts and bridge-rules. Chapter 6 presents how these customizations can be accomplished. This Sigon version can be accessed https://github.com/sigon-lang/sigon.

Even though the Multi-Context System was not present in the Systematic Literature Mapping, Besold (2009) and Besold and Mandl (2010b,a) generalize concepts of the MCS showed in Brewka and Eiter (2007) to be applicable for both logical and sub-symbolic reasoners. Besold (2009) and Besold and Mandl (2010b,a) focused on formally redefining the required concepts to integrate a sub-symbolic reasoner. A proof of concept and several examples were presented. However, the implementation is left as future work. We believe the main difference between these works and ours is that we employed a BDI agent to model the MCS and focused on the implementation rather than formal modeling an MCS with a connectionist context.

Several works have already explored the usage of neural networks with BDI agents in different scenarios (LOKUGE; ALAHAKOON, 2004; LOKUGE; ALAHAKOON; DISSANAYAKE, 2004; HERRERO et al., 2007; SUBAGDJA; TAN, 2008, 2009; DE GREGORIO, 2008; HONARVAR; GHASEM-AGHAEE, 2009; CHEN; LONG; JIANG, 2015). Lokuge and Alahakoon (2004), Lokuge, Alahakoon, and Dissanayake (2004) and Subagdja and Tan (2008, 2009) present intelligent systems architectures in which the agent's beliefs, desires, intentions, and plans are used as input to a neural network to identify a set of plans for achieving the desires and goals in the system. Herrero et al. (2007) present BDI agents that use the artificial neural network to identify intrusions in computer networks. De Gregorio (2008) combines virtual neural sensors with a BDI agent to model and evaluate an active video surveillance system in complex scenarios. Honarvar and Ghasem-Aghaee (2009) propose an ethical BDI agent that employs a neural network to determine if a behavior or action is ethically right or wrong. Chen, Long, and Jiang (2015) propose a new BDI model to improve the overall utility of the neural network.

Our research followed an approach similar to the previous studies in which a neural network is employed during the sensing, planning, and acting phases of the agent's reasoning cycle. However, it is possible to notice that these works used the neural network only in one step of the agent's reasoning cycle. De Gregorio (2008) focused on the sensing phase, Lokuge and Alahakoon (2004), Lokuge, Alahakoon, and Dissanayake (2004) and Subagdja and Tan (2008, 2009) on the planning phase, Honarvar and Ghasem-Aghaee (2009) in the action phase, and Chen, Long, and Jiang (2015) focused in using the model to enhance the neural network and optimize the agent's goals. We believe the main advantage of our work is the flexibility and modularity achieved by combining Multi-Context Systems (MCS), custom contexts and sensors, and a high-level language to develop these intelligent systems. For instance, custom sensors can handle different data types, and custom contexts can be modeled to integrate this information through bridge-rules. These characteristics enable the developer to employ the neural network or other AI methods in different steps of the agent's reasoning cycle. Erduran (2022) presents a literature review about how several works handle the integration between Machine Learning algorithms and software agents. This work proposes a multidimensional ML-COG Cube to illustrate the integration perspectives for both paradigms and present relevant research questions and open issues. Erduran (2022) classifies and compares our integration method presented in Rodrigo Rodrigues Pires de Mello, Silveira, and Santiago (2022) with several studies integrating different learning methods with BDI agents.

Based on the previously presented related works, we believe that our work contributions are: (i) a neural-symbolic agent and an integration method inspired by Multi-Context Systems, in which different AI methods are employed in the agent's reasoning cycle and (ii) design a novel approach for developing modular intelligent agents with the possibility of employing different Machine Learning techniques. With these contributions, we intend to help the community while shifting the paradigm of building a programming-based model to a trained-based model (BORDINI et al., 2020).

9 CONTRIBUTIONS

This chapter summarizes our work's contributions and limitations. In Section 9.1, we present how we answered the research questions and the results from this work. In Section 9.2, the analyses of the objectives are described. Finally, in Section 9.3, the limitations of our research are discussed.

9.1 RESPONSE TO THE RESEARCH QUESTION

During this work, we built and evaluated three artifacts to answer the research question of how to enhance intelligent systems by integrating connectionist methods into the agent's reasoning cycle. In the first artifact, we modeled and implemented the first version of the neural-symbolic agent and the integration method. In the second artifact, we improved the integration method by using the neural network's output as part of a bridge-rule. The experiments executed in the second artifact showed that the agent adapted to different situations without requiring adding new rules and matched the best negotiating agents available in the framework GENIUS. GENIUS is the official tool used in the Automated Negotiating Agents Competition (ANAC), which helps the research community benchmark and evaluate its work (JONKER et al., 2017). In the third artifact, we explored how a neural-symbolic agent can improve the neural network's performance. The experiment's results showed that compared to the baseline version of this experiment, which achieved an accuracy of 84.6%, the neural-symbolic agent achieved 84.4% and was 48% faster during the training phase. Considering these results, the proposed agent and integration method mitigates the necessity of hand-crafted rules and improves the neural network's performance, thus enhancing the intelligent systems' decision-making in these scenarios.

9.2 ANALYSIS OF OBJECTIVES

Following, we address how we achieved the specific goals of this research:

1. Specify the model of an intelligent agent following a Multi-Context System and BDI-like approach: in Chapter 5, we present the details about the neural-symbolic properties and how the proposed agent is modeled and implemented;

2. Investigate ways of integrating neural networks in different phases of the agent's reasoning cycle: We integrated the connectionist methods into the agent's sensing and planning. We also employed the neural network's output as part of a bridge-rule responsible for information exchange integration. The integration's details and a case study with the agent's reasoning cycle were presented in Chapters 5 and 6. In Section 8.1, a Systematic Literature Mapping (SLM) is presented.

This SLM focuses on how different works model, develop, evaluate, and deploy neural-symbolic agents;

3. Deploy the agent in a real-world scenario: In Chapters 6, a A case study of two neural-symbolic agents was presented. Section 6.1 presented a negotiating agent that employs the NN's output to mitigate the necessity of hard-coded and hand-crafted rules. Section 6.2 presented an agent for malware detection. This agent establishes when to use the training method, fine-tuning, or feature extraction. The agent changes the NN's structure and parameters to execute this method.
4. Evaluate and validate the proposed model according to the deployed scenarios: In Section 7.1, we compared the proposed agent with different negotiating agents and evaluated the impacts of employing the NN's output during decision-making. In Section 7.2, we explored a more complex, dynamic, and closer to a real-world scenario. In this scenario, we analyzed whether the agent's decision-making can improve the NN's performance (accuracy and time to process the entire dataset) by changing the NN's structure and parameters.

9.3 LIMITATIONS

The results of this research help us to evaluate and find the limitations of the proposed agent. In this sense, our work's limitations can be divided as follows: (i) Systematic Literature Mapping (SLM) validity threats, (ii) scenarios employed in the case study and experiments, and (iii) employing different neural network architectures. From the SLMs' perspective, researchers' bias and expertise are the most relevant validity threats. In the second perspective, it is possible to notice that the scenario used during the case study and experiments presented in Chapter 7.1 did not represent a complex environment. For instance, the negotiation agent solved the conflict without needing to learn or adapt during the conflict resolution or after a certain amount of encounters. Considering this scenario's complexity, the results achieved could be limited. In the third perspective, we explored only two types of neural network architecture: Multilayer Perceptron and Convolutional. In this sense, we believe studying and evaluating how different architectures impact the agent's decision-making is necessary.

9.4 SCIENTIFIC PUBLICATIONS

In this work, we executed three iterations of the Design Science Research Methodology (DSRM). During each iteration, it is necessary to execute a communication activity. In the first two iterations, we published a paper about the main findings of the SLM presented in Chapter 4, and another paper on the first version of the proposed model and case study presented in Chapters 5 and 6. In the second iteration,

Table 8 – Scientific publications

Reference	Qualis	Category
de Mello, R. R. P., Silveira, R. A., & de Santiago, R. (2021). Integrating neural networks into the agent's decision-making: A systematic literature mapping. In 15th Workshop-School on Agents, Environments, and Applications.	NA	Conference
de Mello, R. R. P., Silveira, R. A., & de Santiago, R. (2022). A Mediator Agent based on Multi-Context System and Information Retrieval. In ICAART (2) (pp. 78-87).	A4	Conference
de Mello, R. R. P., de Santiago, R., Silveira, R. A., & Gelaim, T. Â. (2024). Neural-symbolic BDI-Agent as a Multi-Context System: A case study with negotiating agent. Expert Systems with Applications, 238, 121656.	A1	Journal

we published a paper about the improved version of the neural-symbolic agent and the evaluation presented in 7.1. Considering these activities, Table 8 presents the information of the published papers. Qualis assesses the quality of articles and other types of production based on the analysis of the quality of the dissemination vehicles, that is, scientific journals (CAPES, 2024). The information about the qualis of the journals and conference was extracted from the official document of the Graduate Program in Computer Science (PPGCC) of UFSC (PPGCC, 2024). This information is based on the evaluation provided by CAPES (2024).

10 CONCLUSION

Using intelligent agents to assist humans or systems during daily tasks is a challenging and relevant problem in AI. Most of these tasks require handling different types of information, and employing only one AI (i.e., symbolic or connectionist) method could not suffice. Symbolic and connectionist methods diverge in terms of their data representations and problem-solving approaches. Symbolic methods rely on discrete symbolic representations and traditional search algorithms to discover solutions, while connectionist systems employ continuous feature vector representations and neural cells to learn mapping functions. Consequently, a significant challenge lies in designing a unified framework that seamlessly integrates both symbolic and neural components (YU et al., 2021). Considering these, we proposed a neural-symbolic BDI-agent modeled as a Multi-Context System (MCS). We divided this research into modeling, reasoning, and experiments.

Initially, we executed a Systematic Literature Mapping (SLM) to establish the problem and motivation of this research. Our SLM focused on finding evidence about how studies combine neural networks and intelligent agents to solve problems. We analyzed over 1000 papers, and after the execution of inclusion/exclusion criteria, only 110 papers remained. The most relevant finding of this SLM was that only 5% of the studies explored the integration of neural networks as part of the agent's reasoning cycle.

To design the integration between symbolic and connectionist methods, we proposed a neural-symbolic BDI-agent based Multi-Context Systems (MCS). The proposed integration method models custom sensors and contexts to handle different data types and AI methods. The implementations used the Sigon framework (GELAIM et al., 2019). We also implemented some changes in the Sigon grammar and developed a new version in Python. This decision enabled us to accommodate the proposed model and facilitate the integration between the most relevant Machine Learning libraries (Scikit learn, Pytorch, Tensorflow, and Keras). The reasoning cycle starts with a perception processed by a sensor, which could change the agent's mental state and trigger multiple bridge-rules. We integrated the decision provided by the neural network as information of the Neural Network Context (NNC), which can be combined with different contexts via bridge-rules.

We presented a case study with a negotiating agent and an agent for malware detection. We also presented the agents' reasoning cycle and how they can be implemented in Sigon. The negotiating agent employs the NN's output during its decision-making. The agent for malware detection establishes which method to use to process the dataset (conventional training method or a transfer learning technique). To achieve this task, the agent monitors the NN's accuracy. Unlike the negotiation agent, which only integrates the NN's outputs, this agent changes the NN's structure and parameters. We claim that these agents are complementary. The negotiating agent employs a strategy to solve the conflict without handling a significant amount of data. The second one acts in a scenario closer to the real world, in which it is required to process a substantial volume of data that arrives at different moments.

We divided the evaluation into two parts. The first evaluation focused on analyzing whether the neural-symbolic agent can reduce the necessity of hand-crafted and hard-coded rules. The scenario in this evaluation enabled us to compare the proposed agent with 136 negotiating agents available in the GENIUS framework. In the second evaluation, we analyzed whether the proposed agent can improve the neural network's performance. Our results in the first evaluation showed that the agent adapted to different situations and achieved the maximum utility value in both situations, matching the best agents competing in the ANAC competition. The evaluation also showed that executing the neural network in every reasoning cycle could not be adequate in scenarios that require rapid responses. In the second evaluation, as opposed to the baseline version, which attained an accuracy of 84.6%, the neural-symbolic agent achieved 84.4% accuracy while also completing the training phase 48% faster.

We executed other Systematic Literature Mapping (SLM), which enabled us to analyze how different studies models, design, evaluate, and deploy neural-symbolic agents. We analyzed 58 papers, and the main findings from this SLM were the following: (i) the methodologies of learning for reasoning and reasoning for learning were present in 59% of the retrieved works; (ii) 88,2% of the works focused on planning, natural language processing, and reinforcement learning as the field of study; and (iii) 88% compare the implemented neural-symbolic agent with different works or with different versions implemented on its research. In our work, we employed the learning for reasoning methodology, in which NN's output is used as part of the agent's decision-making. We also explored the reasoning for learning, in which the agent changes the neural network's parameters during its execution. To analyze the proposed agent, we compared our research with several works and approaches that only used one AI method.

Considering our research results, we believe the results of our work can improve the agent's decision-making by raising the level of abstraction and providing a flexible and modular integration. This result is achieved by combining Multi-Context Systems (MCS), custom contexts and sensors, and Sigon as a high-level language. These characteristics enable the developer to employ the neural network or other AI methods in different steps of the agent's reasoning cycle. The proposed agent could represent an improved agent architecture that integrates and employs several AI methods during the agent's decision-making. We claim that this approach increases the development of modular intelligent systems capable of using different AI techniques. We believe the new version of Sigon could fit as a relevant framework to model neural-symbolic intelligent systems. With these contributions, we intend to help the community while shifting the paradigm of building a programming-based model to a trained-based model (BORDINI et al., 2020).

Based on the modeling of the proposed agent and the taxonomy of neural-symbolic systems presented in Section 3.6, we claim that our research tackled the following two integration approaches: (i) learning for reasoning: the negotiating agent employed the neural network's output during its decision-making; and (ii) the reasoning for learning: the decision-making of the neural-symbolic agent for malware detection changed some of the neural network's parameter, which improved the NN's accuracy and the time to finish the training phase. In this work, we did not evaluate whether the proposed method could tackle the learning-reasoning, in which both neural and symbolic systems play equal roles and work together in a mutually beneficial way. To mitigate this research's limitations and improve the proposed model in future works, we intend to focus on the following aspects:

1. Bordini et al. (2020) defend that integrating AI techniques with BDI agents can occur in the sensing, planning, and acting phases. This research focused on employing only a Machine Learning method (neural networks) as part of the agent's sensing and planning phase. Considering this limitation, we aim to evolve the proposed agent to a more general architecture, exploring the acting phase and evaluating the integration of different Machine Learning methods during the agent's decision-making;

2. In this work, we did not focus on whether the integration method can mitigate some of the connectionist limitations, such as requiring a substantial amount of data and explainability. In future work, we intend to explore whether different explainability techniques could be modeled as a custom context and integrated via bridge-rules;

3. We intend to keep improving the proposed integration method and the Sigon framework. The current version was implemented to accommodate the proposed agent. However, it is required to improve its performance and provide adequate documentation to the community;

4. The evaluations presented in our work did not cover whether the neural-symbolic agent reduces the required time to develop the neural network and integrate it with the proposed model. In this sense, in future works, we intend to explore whether using the proposed neural-symbolic agent reduces the required time and Lines of Code (LOC) to solve a real-world problem;

5. We should deploy the proposed agent in several scenarios. This step is crucial to explore how the agent would adapt and analyze whether the neural-symbolic

agent could execute without human intervention. For instance, in this research, we did not explore any scenario of natural language processing. The SLM about neural-symbolic agents showed that this scenario has been gaining attention in recent years.

BIBLIOGRAPHY

ADADI, Amina; BERRADA, Mohammed. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). **IEEE Access**, IEEE, v. 6, p. 52138–52160, 2018.

AMRANI, N.E.A. et al. A new interpretation technique of traffic signs, based on Deep Learning and Semantic Web. In: cited By 0. DOI: 10.1109/ICDS47004.2019.8942319. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85078283281&doi=10.1109%2fICDS47004.2019.8942319&partnerID=40&md5=3531a1cb17db65229a2630a1effe472e.

ANDERSON, Hyrum S; ROTH, Phil. Ember: an open dataset for training static pe malware machine learning models. **arXiv preprint arXiv:1804.04637**, 2018.

ANJOMSHOAE, Sule et al. Explainable agents and robots: Results from a systematic literature review. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS and MULTIAGENT SYSTEMS. PROCEEDINGS of the 18th International Conference on Autonomous Agents and MultiAgent Systems. [S.l.: s.n.], 2019. P. 1078–1088.

ARRIETA, Alejandro Barredo et al. **Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI**. [S.l.: s.n.], 2019. arXiv: 1910.10045 [cs.AI].

ASAI, Masataro; MUISE, Christian. Learning neural-symbolic descriptive planning models via cube-space priors: The voyage home (to STRIPS). **arXiv preprint arXiv:2004.12850**, 2020.

BAARSLAG, Tim et al. When will negotiation agents be able to represent us? The challenges and opportunities for autonomous negotiators. In: INTERNATIONAL JOINT CONFERENCES ON ARTIFICIAL INTELLIGENCE.

BALTRUŠAITIS, Tadas; ROBINSON, Peter; MORENCY, Louis-Philippe. Openface: an open source facial behavior analysis toolkit. In: IEEE. 2016 IEEE Winter Conference on Applications of Computer Vision (WACV). [S.l.: s.n.], 2016. P. 1–10.

BASU, Kinjal et al. A hybrid neuro-symbolic approach for text-based games using inductive logic programming. In: COMBINING Learning and Reasoning: Programming Languages, Formalisms, and Representations. [S.l.: s.n.], 2022.

BENNETOT, Adrien et al. Towards explainable neural-symbolic visual reasoning. In:

BESOLD, Tarek R. **Theory and Implementation of Multi-Context Systems Containing Logical and Sub-Symbolic Contexts of Reasoning**. 2009. PhD thesis – Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU).

BESOLD, Tarek R; GARCEZ, Artur d'Avila, et al. Neural-symbolic learning and reasoning: A survey and interpretation. **arXiv preprint arXiv:1711.03902**, 2017.

BESOLD, Tarek R; MANDL, Stefan. Integrating Logical and Sub-symbolic Contexts of Reasoning. In: ICAART (1). [S.l.: s.n.], 2010. P. 494–497.

BESOLD, Tarek R; MANDL, Stefan. Towards an implementation of a multi-context system framework. **MRC 2010**, p. 13, 2010.

BORDINI, Rafael H et al. Agent programming in the cognitive era. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 34, n. 2, p. 1–31, 2020.

BRATMAN, Michael. Intention, plans, and practical reason, 1987.

BREWKA, Gerhard; EITER, Thomas. Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI. [S.l.: s.n.], 2007. P. 385–390.

BREWKA, Gerhard; EITER, Thomas; FINK, Michael. Nonmonotonic multi-context systems: A flexible approach for integrating heterogeneous knowledge sources. In: LOGIC programming, knowledge representation, and nonmonotonic reasoning. [S.l.]: Springer, 2011. P. 233–258.

BREWKA, Gerhard; ELLMAUTHALER, Stefan; PÜHRER, Jörg. Multi-Context Systems for Reactive Reasoning in Dynamic Environments. In: ECAI. [S.l.: s.n.], 2014. P. 159–164.

CABALAR, Pedro et al. Multi-context systems in dynamic environments. **Annals of Mathematics and Artificial Intelligence**, Springer, v. 86, n. 1-3, p. 87–120, 2019.

CAPES, MEC. **Qualis**. [S.l.: s.n.], 2024. Available from: https://sucupira.capes.gov.br/sucupira/. Visited on: 30 Jan. 2024.

CASALI, A.; GODO, L.; SIERRA, C. Graded BDI models for agent architectures. English. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, 3487 LNAI, p. 126–143, 2005. ISSN 03029743.

CHAPUT, Rémy et al. A multi-agent approach to combine reasoning and learning for an ethical behavior. In: PROCEEDINGS of the 2021 AAAI/ACM Conference on AI, Ethics, and Society. [S.l.: s.n.], 2021. P. 13–23.

CHEN, Huang; LONG, Chen; JIANG, Hao-Bin. Building a Belief–Desire–Intention Agent for Modeling Neural Networks. **Applied Artificial Intelligence**, Taylor & Francis, v. 29, n. 8, p. 753–765, 2015.

CHEN, I.-M.; ZHAO, C.; CHAN, C.-Y. A Deep Reinforcement Learning-Based Approach to Intelligent Powertrain Control for Automated Vehicles. In: cited By 0. DOI: 10.1109/ITSC.2019.8917076. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85076813029&doi=10.1109%2fITSC.2019.8917076&partnerID=40&md5=05162af6864be9085ad374c0283d3df7.

CHENG, Qixiang et al. Silicon Photonics Codesign for Deep Learning. **Proceedings of the IEEE**, PP, p. 1–22, Feb. 2020. DOI: 10.1109/JPROC.2020.2968184.

CHOLLET, François et al. **Keras**. [S.l.: s.n.], 2015. https://keras.io.

CHOLLET, Francois et al. **Deep learning with Python**. [S.l.]: Manning New York, 2018. v. 361.

CHONG, Hui-Qing; TAN, Ah-Hwee; NG, Gee-Wah. Integrated cognitive architectures: a survey. **Artificial Intelligence Review**, Springer, v. 28, n. 2, p. 103–130, 2007.

DE GREGORIO, Massimo. An intelligent active video surveillance system based on the integration of virtual neural sensors and BDI agents. **IEICE TRANSACTIONS on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, v. 91, n. 7, p. 1914–1921, 2008.

DE PAULA, S.M.; GUDWIN, R.R. Evolving conceptual spaces for symbol grounding in language games. **Biologically Inspired Cognitive Architectures**, 2015. cited By 5. DOI: 10.1016/j.bica.2015.09.006. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84945901227&doi=10.1016%2fj.bica.2015.09.006&partnerID=40&md5=cdea2523134ad8af41688b10bed26325.

DIALLO, Elhadji Amadou Oury; SUGIYAMA, Ayumi; SUGAWARA, Toshiharu. Learning to coordinate with deep reinforcement learning in doubles pong game. In: IEEE. 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). [S.l.: s.n.], 2017. P. 14–19.

DIXON-WOODS, Mary et al. Conducting a critical interpretive synthesis of the literature on access to healthcare by vulnerable groups. **BMC medical research methodology**, BioMed Central, v. 6, n. 1, p. 1–13, 2006.

EITER, Thomas et al. Finding explanations of inconsistency in multi-context systems. **Artificial Intelligence**, v. 216, p. 233–274, 2014. ISSN 0004-3702. DOI: https://doi.org/10.1016/j.artint.2014.07.008. Available from: https://www.sciencedirect.com/science/article/pii/S0004370214000915.

ELFENBEIN, Hillary Anger et al. Reading your counterpart: The benefit of emotion recognition accuracy for effectiveness in negotiation. **Journal of Nonverbal Behavior**, Springer, v. 31, n. 4, p. 205–223, 2007.

ERDURAN, Ömer Ibrahim. Machine Learning Algorithms for Cognitive and Autonomous BDI Agents, 2022.

EUGSTER, Patrick Th et al. The many faces of publish/subscribe. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 35, n. 2, p. 114–131, 2003.

GARCEZ, Artur; BESOLD, Tarek R, et al. Neural-symbolic learning and reasoning: contributions and challenges, 2015.

GARCEZ, Artur d´’vila; GORI, Marco, et al. **Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning**. [S.l.: s.n.], 2019. arXiv: 1905.06088 `[cs.AI]`.

GARCEZ, Artur d'Avila et al. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. **arXiv preprint arXiv:1905.06088**, 2019.

GARCEZ, Artur d'Avila et al. Neural-symbolic learning and reasoning: A survey and interpretation. **Neuro-Symbolic Artificial Intelligence: The State of the Art**, IOS press, v. 342, n. 1, p. 327, 2022.

GARG, D.; CHLI, M.; VOGIATZIS, G. A Deep Reinforcement Learning Agent for Traffic Intersection Control Optimization. In: cited By 0. DOI: 10.1109/ITSC.2019.8917361. Available from:
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85076820366&doi=10.1109%2fITSC.2019.8917361&partnerID=40&md5=e1f11b8301d133545733109bf381ee37.

GARNELO, Marta; ARULKUMARAN, Kai; SHANAHAN, Murray. Towards deep symbolic reinforcement learning. **arXiv preprint arXiv:1609.05518**, 2016.

GARNELO, Marta; SHANAHAN, Murray. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. **Current Opinion in Behavioral Sciences**, Elsevier, v. 29, p. 17–23, 2019.

GARNELO, Marta; SHANAHAN, Murray. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. **Current Opinion in Behavioral Sciences**, v. 29, p. 17–23, 2019. SI: 29: Artificial Intelligence (2019). ISSN 2352-1546. DOI: https://doi.org/10.1016/j.cobeha.2018.12.010. Available from: http://www.sciencedirect.com/science/article/pii/S2352154618301943.

GELAIM, Thiago Angelo. **Situation awareness and practical reasoning in dynamic environments**. 2021. PhD thesis – Federal University of Santa Catarina.

GELAIM, Thiago Ângelo et al. Sigon: A multi-context system framework for intelligent agents. **Expert Systems with Applications**, Elsevier, v. 119, p. 51–60, 2019.

GHOUTI, Lahouari; IMAM, Muhammad. Malware classification using compact image features and multiclass support vector machines. **IET Information Security**, Wiley Online Library, v. 14, n. 4, p. 419–429, 2020.

GU, Jiuxiang et al. Recent advances in convolutional neural networks. **Pattern recognition**, Elsevier, v. 77, p. 354–377, 2018.

HECHT-NIELSEN, ROBERT. III.3 - Theory of the Backpropagation Neural Network**Based on “nonindent” by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE. In: WECHSLER, Harry (Ed.). **Neural Networks for Perception**. [S.l.]: Academic Press, 1992. P. 65–93. ISBN 978-0-12-741252-8. DOI: https://doi.org/10.1016/B978-0-12-741252-8.50010-8. Available from: https://www.sciencedirect.com/science/article/pii/B9780127412528500108.

HERRERO, Álvaro et al. Hybrid multi agent-neural network intrusion detection with mobile visualization. **Innovations in Hybrid Intelligent Systems**, Springer, p. 320–328, 2007.

HITZLER, Pascal et al. Neural-symbolic integration and the semantic web. **Semantic Web**, IOS Press, v. 11, n. 1, p. 3–11, 2020.

HONARVAR, Ali Reza; GHASEM-AGHAEE, Nasser. An artificial neural network approach for creating an ethical artificial agent. In: IEEE. 2009 IEEE international symposium on computational intelligence in robotics and automation-(CIRA). [S.l.: s.n.], 2009. P. 290–295.

HOU, Yaqing; FENG, Liang; ONG, Yew-Soon. Creating human-like non-player game characters using a memetic multi-agent system. In: IEEE. 2016 International Joint Conference on Neural Networks (IJCNN). [S.l.: s.n.], 2016. P. 177–184.

ILKOU, Eleni; KOUTRAKI, Maria. Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies? In: CIKM (Workshops). [S.l.: s.n.], 2020.

JEDRZEJOWICZ, Piotr. Machine learning and agents. In: SPRINGER. KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications. [S.l.: s.n.], 2011. P. 2–15.

JENNINGS, Nicholas R et al. Automated negotiation: prospects, methods and challenges. **Group Decision and Negotiation**, Springer, v. 10, n. 2, p. 199–215, 2001.

JONKER, Catholijn et al. Automated negotiating agents competition (ANAC). In: 1. PROCEEDINGS of the AAAI conference on artificial intelligence. [S.l.: s.n.], 2017.

KASSAB, Mohamad et al. Software architectural patterns in practice: an empirical study. **Innovations in Systems and Software Engineering**, Springer, v. 14, n. 4, p. 263–271, 2018.

KING, Davis E. Dlib-ml: A machine learning toolkit. **The Journal of Machine Learning Research**, JMLR. org, v. 10, p. 1755–1758, 2009.

KITCHENHAM, Barbara; CHARTERS, Stuart. Guidelines for performing systematic literature reviews in software engineering. Citeseer, 2007.

KLOSE, Patrick; MESTER, Rudolf. Simulated autonomous driving in a realistic driving environment using deep reinforcement learning and a deterministic finite state machine. In: PROCEEDINGS of the 2nd International Conference on Applications of Intelligent Systems. [S.l.: s.n.], 2019. P. 1–6.

KOTYAN, S.; VARGAS, D.V.; VENKANNA, U. Self Training Autonomous Driving Agent. In: cited By 0. DOI: 10.23919/SICE.2019.8859883. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85073874223&doi=10.23919%2fSICE.2019.8859883&partnerID=40&md5=9ced2c365ba6564c4bdab494b71e8043.

KRIESEL, David. A brief introduction on neural networks. Citeseer, 2007.

LAMOUIK, I.; YAHYAOUY, A.; SABRI, M.A. Smart multi-agent traffic coordinator for autonomous vehicles at intersections. In: cited By 6. DOI: 10.1109/ATSIP.2017.8075564. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85035344396&doi=10.1109%2fATSIP.2017.8075564&partnerID=40&md5=fc48fcfa1fb5d664f002a7fa0874032c.

LESORT, Timothée et al. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. **Information fusion**, Elsevier, v. 58, p. 52–68, 2020.

LI, Zewen et al. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. **IEEE Transactions on Neural Networks and Learning Systems**, v. 33, n. 12, p. 6999–7019, 2022. DOI: 10.1109/TNNLS.2021.3084827.

LIN, Raz et al. Genius: An integrated environment for supporting the design of generic automated negotiators. **Computational Intelligence**, Wiley Online Library, v. 30, n. 1, p. 48–70, 2014.

LOKUGE, Prasanna; ALAHAKOON, Damminda. Hybrid BDI agents with improved learning capabilities for adaptive planning in a container terminal application. In: IEEE. PROCEEDINGS. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004.(IAT 2004). [S.l.: s.n.], 2004. P. 120–126.

LOKUGE, Prasanna; ALAHAKOON, Damminda; DISSANAYAKE, Parakrama. Collaborative neuro-BDI agents in container terminals. In: IEEE. 18TH International Conference on Advanced Information Networking and Applications, 2004. AINA 2004. [S.l.: s.n.], 2004. P. 155–158.

LOUMIOTIS, I. et al. Road Traffic Prediction Using Artificial Neural Networks. In: cited By 3. DOI: 10.23919/SEEDA-CECNSM.2018.8544943. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85059781489&doi=10.23919%2fSEEDA-CECNSM.2018.8544943&partnerID=40&md5=51fa25d8728563f7741a244c35258628.

LU, Jie et al. Transfer learning using computational intelligence: A survey. **Knowledge-Based Systems**, v. 80, p. 14–23, 2015. 25th anniversary of Knowledge-Based Systems. ISSN 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2015.01.010. Available from: https://www.sciencedirect.com/science/article/pii/S0950705115000179.

LUO, Ning; ZHOU, Yilu; SHON, John. Employee satisfaction and corporate performance: Mining employee reviews on glassdoor. com, 2016.

LUO, T.; SUBAGDJA, B., et al. Multi-Agent Collaborative Exploration through Graph-based Deep Reinforcement Learning. In: cited By 0. DOI: 10.1109/AGENTS.2019.8929168. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85077821596&doi=10.1109%2fAGENTS.2019.8929168&partnerID=40&md5=be37a0c29bb33942027a05aa2dd06981.

MAJUMDAR, A.; BENAVIDEZ, P.; JAMSHIDI, M. Multi-Agent Exploration for Faster and Reliable Deep Q-Learning Convergence in Reinforcement Learning. In: cited By 2. DOI: 10.23919/WAC.2018.8430409. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85052097888&doi=10.23919%2fWAC.2018.8430409&partnerID=40&md5=767adbb830941940860604336055a3a3.

MARCUS, Gary. Deep learning: A critical appraisal. **arXiv preprint arXiv:1801.00631**, 2018.

MARRA, Giuseppe et al. Integrating Learning and Reasoning with Deep Logic Models. **arXiv preprint arXiv:1901.04195**, 2019.

MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **Bulletin of mathematical biology**, Springer, v. 52, n. 1-2, p. 99–115, 1990.

MELLO, Rodrigo Rodrigues Pires de; GELAIM, Thiago Ângelo; SILVEIRA, Ricardo Azambuja. Negotiating Agents: A Model Based on BDI Architecture and Multi-Context Systems Using Aspiration Adaptation Theory as a

Negotiation Strategy. In: SPRINGER. CONFERENCE on Complex, Intelligent, and Software Intensive Systems. [S.l.: s.n.], 2018. P. 351–362.

MELLO, Rodrigo Rodrigues Pires de; SILVEIRA, Ricardo Azambuja; SANTIAGO, Rafael de. **A Mediator Agent based on Multi-Context System and Information Retrieval**. [S.l.]: ICAART, 2022.

MELLO, Rodrigo Rodrigues Pires de; GELAIM, Thiago Ângelo; SILVEIRA, Ricardo Azambuja. Negotiating Agents: A Model Based on BDI Architecture and Multi-Context Systems Using Aspiration Adaptation Theory as a Negotiation Strategy. In: SPRINGER. CONFERENCE on Complex, Intelligent, and Software Intensive Systems. [S.l.: s.n.], 2018. P. 351–362.

MELLO, Rodrigo Rodrigues Pires de; SILVEIRA, Ricardo Azambuja; SANTIAGO, Rafael de. Integrating neural networks into the agent's decision-making: A Systematic Literature Mapping. In: PROCEEDINGS for 15th Workshop-School on Agents, Environments, and Applications. [S.l.: s.n.], 2021.

MELLO, Rodrigo Rodrigues Pires de et al. Neural-symbolic BDI-Agent as a Multi-Context System: A case study with negotiating agent. **Expert Systems with Applications**, Elsevier, v. 238, p. 121656, 2024.

MINSKY, Marvin L. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. **AI magazine**, v. 12, n. 2, p. 34–34, 1991.

MNIH, Volodymyr; KAVUKCUOGLU, Koray; SILVER, David; GRAVES, Alex, et al. **Playing Atari with Deep Reinforcement Learning**. [S.l.: s.n.], 2013. arXiv: 1312.5602 [cs.LG].

MNIH, Volodymyr; KAVUKCUOGLU, Koray; SILVER, David; RUSU, Andrei A, et al. Human-level control through deep reinforcement learning. **nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

MOON, Jiyoun. Plugin Framework-Based Neuro-Symbolic Grounded Task Planning for Multi-Agent System. **Sensors**, MDPI, v. 21, n. 23, p. 7896, 2021.

MURUGESAN, Keerthiram et al. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines. In: 10. PROCEEDINGS of the AAAI Conference on Artificial Intelligence. [S.l.: s.n.], 2021. P. 9018–9027.

OZAKI, Ana. Learning Description Logic Ontologies: Five Approaches. Where Do They Stand? **KI-Künstliche Intelligenz**, Springer, v. 34, n. 3, p. 317–327, 2020.

PARISOTTO, Emilio et al. Neuro-symbolic program synthesis. **arXiv preprint arXiv:1611.01855**, 2016.

PARKHI, Omkar M; VEDALDI, Andrea; ZISSERMAN, Andrew. Deep face recognition. British Machine Vision Association, 2015.

PARSONS, S.; SIERRA, C.; JENNINGS, N. Agents that reason and negotiate by arguing. English. **Journal of Logic and Computation**, v. 8, n. 3, p. 261–292, 1998. ISSN 0955792X.

PEFFERS, Ken et al. A design science research methodology for information systems research. **Journal of management information systems**, Taylor & Francis, v. 24, n. 3, p. 45–77, 2007.

PENNING, Leo de et al. A neural-symbolic cognitive agent for online learning and reasoning. In: INTERNATIONAL JOINT CONFERENCES ON ARTIFICIAL INTELLIGENCE. PROCEEDINGS of the Twenty-Second international joint conference on Artificial Intelligence. [S.l.: s.n.], 2011. P. 1653–1658.

PETTICREW, Mark; ROBERTS, Helen. **Systematic reviews in the social sciences: A practical guide**. [S.l.]: John Wiley & Sons, 2008.

PPGCC, UFSC. **Qualis - SICLAP**. [S.l.: s.n.], 2024. Available from: https://docs.google.com/spreadsheets/d/1wwuBauzM7OGXKNMI0WVNJbeStI6mFhmo_p5tNR3yawk/edit#gid=0. Visited on: 30 Jan. 2024.

RAFF, Edward et al. Malware detection by eating a whole exe. In: WORKSHOPS at the thirty-second AAAI conference on artificial intelligence. [S.l.: s.n.], 2018.

RAHMAN, Mohammad Saidur; COULL, Scott; WRIGHT, Matthew. On the Limitations of Continual Learning for Malware Classification. In: PMLR. CONFERENCE on Lifelong Learning Agents. [S.l.: s.n.], 2022. P. 564–582.

RAMEZANI DOORAKI, Amir; LEE, Deok-Jin. An end-to-end deep reinforcement learning-based intelligent agent capable of autonomous exploration in unknown environments. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 18, n. 10, p. 3575, 2018.

RAO, Anand S; GEORGEFF, Michael P, et al. BDI agents: From theory to practice. In: ICMAS. [S.l.: s.n.], 1995. P. 312–319.

RIZK, Y.; AWAD, M.; TUNSTEL, E.W. Decision Making in Multiagent Systems: A Survey. **IEEE Transactions on Cognitive and Developmental Systems**, v. 10, n. 3, p. 514–529, 2018. cited By 10. DOI: 10.1109/TCDS.2018.2840971. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85047620369&doi=10.1109%2fTCDS.2018.2840971&partnerID=40&md5=59e75032980c32a945618bff61a6725d.

ROSENFELD, Avi; KRAUS, Sarit. Modeling agents based on aspiration adaptation theory. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 24, n. 2, p. 221–254, 2012.

ROSENFELD, Avi; KRAUS, Sarit. Using aspiration adaptation theory to improve learning. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS and MULTIAGENT SYSTEMS. THE 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. [S.l.: s.n.], 2011. P. 423–430.

RUSSELL, Stuart; NORVIG, Peter. Artificial intelligence: a modern approach, 2002.

SARKER, Md Kamruzzaman et al. Neuro-symbolic artificial intelligence. **AI Communications**, IOS Press, v. 34, n. 3, p. 197–209, 2021.

SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2015. P. 815–823.

SELTEN, Reinhard. Aspiration adaptation theory. **Journal of mathematical psychology**, Elsevier, v. 42, n. 2-3, p. 191–214, 1998.

SERAFIM, Paulo et al. On the development of an autonomous agent for a 3d first-person shooter game using deep reinforcement learning. In: IEEE. 2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames). [S.l.: s.n.], 2017. P. 155–163.

SERENGIL, Sefik Ilkin; OZPINAR, Alper. LightFace: A Hybrid Deep Face Recognition Framework. In: IEEE. 2020 Innovations in Intelligent Systems and Applications Conference (ASYU). [S.l.: s.n.], 2020. P. 23–27. DOI: 10.1109/ASYU50717.2020.9259802.

SHARMA, S. et al. Modeling human behavior during emergency evacuation using intelligent agents: A multi-agent simulation approach. **Information Systems Frontiers**, 2018. cited By 6. DOI: 10.1007/s10796-017-9791-x. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85028361895&doi=10.1007%2fs10796-017-9791-x&partnerID=40&md5=b9ec7725675f7ec1fe271c543408a590.

SOTNIKOV, O.M.; MAZURENKO, V.V. Neural network agent playing spin Hamiltonian games on a quantum computer. **Journal of Physics A: Mathematical and Theoretical**, 2020. cited By 0. DOI: 10.1088/1751-8121/ab73ad. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85082738741&doi=10.1088%2f1751-8121%2fab73ad&partnerID=40&md5=43f8afbf53fc225de7c64f60410d86a5.

SUBAGDJA, Budhitama; TAN, Ah-Hwee. A self-organizing neural network architecture for intentional planning agents. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2009.

SUBAGDJA, Budhitama; TAN, Ah-Hwee. Planning with iFALCON: towards a neural-network-based BDI agent architecture. In: IEEE. 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. [S.l.: s.n.], 2008. P. 231–237.

SUN, Ron. Artificial intelligence: Connectionist and symbolic approaches. Citeseer, 1999.

SUN, Yi. **Deep learning face representation by joint identification-verification**. [S.l.]: The Chinese University of Hong Kong (Hong Kong), 2015.

SUN, Yi; WANG, Xiaogang; TANG, Xiaoou. Deep learning face representation from predicting 10,000 classes. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2014. P. 1891–1898.

SVOZIL, Daniel; KVASNICKA, Vladimír; POSPICHAL, Jirí. Introduction to multi-layer feed-forward neural networks. **Chemometrics and Intelligent Laboratory Systems**, v. 39, n. 1, p. 43–62, 1997. ISSN 0169-7439. DOI: https://doi.org/10.1016/S0169-7439(97)00061-0. Available from: https://www.sciencedirect.com/science/article/pii/S0169743997000610.

TAIGMAN, Yaniv et al. Deepface: Closing the gap to human-level performance in face verification. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2014. P. 1701–1708.

TRESCAK, Tomas et al. Dispute resolution using argumentation-based mediation. **arXiv preprint arXiv:1409.4164**, 2014.

UMILI, Elena et al. Learning a symbolic planning domain through the interaction with continuous environments. In: WORKSHOP on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL), workshop at ICAPS 2021. [S.l.: s.n.], 2021.

VALIANT, Leslie G. Three Problems in Computer Science. **J. ACM**, Association for Computing Machinery, New York, NY, USA, v. 50, n. 1, p. 96–99, Jan. 2003. ISSN 0004-5411. DOI: 10.1145/602382.602410. Available from: https://doi.org/10.1145/602382.602410.

VAN DE VEN, Gido M; TOLIAS, Andreas S. Three scenarios for continual learning. **arXiv preprint arXiv:1904.07734**, 2019.

VON NEUMANN, John; MORGENSTERN, Oskar. **Theory of games and economic behavior (commemorative edition)**. [S.l.]: Princeton university press, 2007.

WANG, D.; TAN, A.-H. Creating Autonomous Adaptive Agents in a Real-Time First-Person Shooter Computer Game. **IEEE Transactions on Computational Intelligence and AI in Games**, 2015. cited By 27. DOI: 10.1109/TCIAIG.2014.2336702. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84933053464&doi=10.1109%2fTCIAIG.2014.2336702&partnerID=40&md5=3359a3acc4ebf49d373c4b4aa31645a2.

WANG, Sun-Chong. Artificial Neural Network. In: INTERDISCIPLINARY Computing in Java Programming. Boston, MA: Springer US, 2003. P. 81–100. ISBN 978-1-4615-0377-4. DOI: 10.1007/978-1-4615-0377-4_5. Available from: https://doi.org/10.1007/978-1-4615-0377-4_5.

WILAMOWSKI, Bogdan M. Neural network architectures and learning algorithms. **IEEE Industrial Electronics Magazine**, v. 3, n. 4, p. 56–63, 2009. DOI: 10.1109/MIE.2009.934790.

WOOLDRIDGE, Michael; JENNINGS, Nicholas R, et al. Intelligent agents: Theory and practice. **Knowledge engineering review**, Cambridge Univ Press, v. 10, n. 2, p. 115–152, 1995.

WOOLDRIDGE, Michael J; JENNINGS, Nicholas R. Intelligent agents: Theory and practice. **The knowledge engineering review**, v. 10, n. 2, p. 115–152, 1995.

YANG, Wen-Chi. When the selfish herd is too crowded to enter. In: IEEE. 2017 IEEE Symposium Series on Computational Intelligence (SSCI). [S.l.: s.n.], 2017. P. 1–8.

YU, Dongran et al. A survey on neural-symbolic systems. **arXiv preprint arXiv:2111.08164**, 2021.

YUKSEL, M.E. Agent-based evacuation modeling with multiple exits using NeuroEvolution of Augmenting Topologies. **Advanced Engineering Informatics**, 2018. cited By 8. DOI: 10.1016/j.aei.2017.11.003. Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85037683029&doi=10.1016%2fj.aei.2017.11.003&partnerID=40&md5=b4d8148d101975ec0eaca0ab7829ac59.

ZHU, Jiangcheng et al. Hierarchical Decision and Control for Continuous Multitarget Problem: Policy Evaluation With Action Delay. **IEEE transactions on neural networks and learning systems**, IEEE, v. 30, n. 2, p. 464–473, 2018.

APPENDIX A – MEDIATOR AGENT

In this chapter, we present a mediator agent that is responsible for solving conflicts during negotiation. The mediation process simulates a real-world case in which the mediator is trustworthy that can employ different resources, and provides new information (TRESCAK et al., 2014). Our main goal is to explore how different types of information (i.e., text and image) can be employed during the agent's decision-making. The mediator agent uses two main strategies during conflict resolution: facial expression recognition and information retrieval. This scenario allows us to explore how different custom contexts interact with other agents' contexts during the reasoning cycle.

Facial expression recognition is a relevant tool for the study of Emotion Recognition Accuracy (ERA). Its usage enables to estimate the impact on objective outcomes in negotiation, a setting that can be highly emotional and in which real-life stakes can be high (ELFENBEIN et al., 2007). Information retrieval will be employed to expand the agent's knowledge about the negotiation item. During this work, the information added into the agent's knowledge base will be used during the planning phase, more precisely, in situations where facial expressions recognition does not provide an output with the required precision or matches the precondition of an existent plan.

A.1 NEGOTIATION SCENARIO DEFINITION

In this case study, we modeled a scenario where a person tries to sell an item to another person. At the beginning of the negotiation, the seller proposes an initial price, and the buyer can accept or propose a new value. Our scenario is inspired in the home improvements negotiation scenario from Parsons, Sierra, and Jennings (1998) and Trescak et al. (2014), in which agents must solve conflicts to reach its design goals. We use a mediator agent to provide a fair negotiation, in which the mediator can advise about the price of the item, trying to satisfy both parties. Since the main objective of this case study is to explore the integration of different information types, the negotiation protocol employed in this scenario is simplified. The following Section shows how we implemented the mediator agent with its two main negotiation strategies.

Section A.2 presents how we modeled a web-scraper and added it to the agent's actuators. We also provide tests regarding similarities functions and several approaches to clean the data that could affect the agent's negotiation strategy. Section A.3 presents the details of the mediator agent implemented in the Sigon framework.

A.2 WEB-SCRAPER IMPLEMENTATION

We focused on extracting information from an e-commerce platform called MercadoLivre. This approach enables us to create new perceptions about the information gathered during this process, improving the agent's decision-making by retrieving new information about an item that is being negotiated. The following Code A.1 shows an example of the output generated by the web-scraper. After this step, the agent can use text sensors to process these perceptions and generate new information about a specific item.

```

[
 {
  " Title ":  "Celular  16gb 2gb Ram LG K7i Mosquito Away 4g Igual K10 11",
  "Price":  698.58,
  "User":  "J.F.IMPORTACAO",
  "Amount": 91,
  "New": false
 }
]
```

Code A.1 – Perception example

During this implementation, we faced a few challenges during the information extraction. The first one is that in some cases, the details about an item are presented in different sections in the platform, affecting the quality of the retrieved data. The second one was related to defining strategies to remove entries that did not represent the item. For instance, we tried retrieving information about a specific smartphone, and the platform returns information about this smartphone's accessories, such as charger and screen protection. In this sense, the value of this entry did not represent an accurate value for this item, affecting the agent's new perceptions.

To mitigate these two limitations, we employed the following strategies: remove the fields **amount** and **new** from an entry. The main reason is that, in some cases, the information was not filled on the platform. In the second limitation, we modeled the following strategies in the agent's auxiliary context:

1. Executing similarity functions: for this strategy, we executed the following similarity functions: Hamming, Levenshtein, Jaro, Jaro-Winkler, Smith-Waterman-Gotoh, Sorensen-Dice, Jaccard Overlap Coefficient. To execute these functions, we used a library called **strutil**. This library implementation can be accessed in GitHub repository https://github.com/adrg/strutil#hamming. We noticed that these functions could not be detected whether a certain entry was not related to the item. Taking that into consideration, we removed this strategy from this implementation;

2. Removing the mild and extreme quartiles: we tried to compute these quartiles, however in some experiments, we noticed that some obvious items were not

removed, or no mild and extreme were detected, even though it was clear that some entries did not represent the searched item;

3. Removing outliers based on the standard deviation: we removed the items that did not match the following criteria:
$price > mean - deviation$ and $price < mean + deviation$. Using this strategy enabled us to remove the entries related to the item accessories, such as screen protectors and chargers. After this step, the agent calculates the mean value of the remaining items, which provides more accurate results about the searched item.

A.3 DESIGNING A MEDIATOR AGENT IN THE SIGON FRAMEWORK

This section presents how to integrate the web-scraper and the trained neural network into an agent developed in Sigon. First, we start by showing how to use the web-scraper as an actuator. Second, we modeled custom context and added the trained neural network into it. Moreover, third, we present the mediator agent modeled as a Multi-Context System. We also provide some reasoning cycles of the mediator during conflict resolution.

In this work we used a framework called Deepface for facial expression recognition. Deepface is a lightweight hybrid high performance face recognition framework, which wraps the most popular face recognition models: VGG-Face (PARKHI; VEDALDI; ZISSERMAN, 2015), FaceNet (SCHROFF; KALENICHENKO; PHILBIN, 2015), OpenFace (BALTRUŠAITIS; ROBINSON; MORENCY, 2016), DeepFace (TAIGMAN et al., 2014), DeepID (SUN; WANG; TANG, 2014; SUN, Y., 2015) and Dlib (KING, 2009) (SERENGIL; OZPINAR, 2020). Those models already reached and passed the human level accuracy of 97.53% (SERAFIM et al., 2017; TAIGMAN et al., 2014).

In listening A.2, we present an initial version of the mediator agent. This agent has three sensors and two actuators. We define the agent's sensors and actuators as follows:

1. Sensor *textSensor* handles perception about the negotiation item and the information exchange between involved parties;

2. Sensor *negotiationPerception* is responsible for handling perceptions about the information retrieved by the actuator;

3. Sensor *imageSensor* handles images containing pictures of seller or buyer;

4. Actuator *defineNextValue* can inform the advice created in the current reasoning cycle, which consists of increasing, decreasing, or keeping the same value during the negotiation phase;

5. Actuator *findInformation* can retrieve information about the negotiation item, such as price, amount, whether it is a brand new product or not.

```
communication:
     sensor("textSensor", "integration . TextSensor").
     sensor("negotiationPerception", " integration .WebScraperPerception").
     sensor("imageSensor", "integration.ImageSensor").
     actuator("defineNextValue", "integration . TextActuator") .
     actuator("findInformation" ,  " integration .WebScraper").

beliefs :
 assistHuman.
 item(LgK10). //smartphone Lgk10
 negotiating .

_neuralNetwork:
 detecEmotion.
 currentEmotion(seller,  neutral) .
 currentEmotion(buyer, neutral) .

_auxiliary :
 retrievedPrice (X).  // proposes a new value based on the information retrieved

desires:
 updateDecision.

intentions :
 updateDecision.

planner:
     plan(
     updateDecision,
     [action(findInformation () ) ],
     [_neuralNetworks: currentEmotion(buyer, neutral),
     _neuralNetworks: currentEmotion(seller, neutral) ], _) .
     plan(
     updateDecision,
     [action(defineNextValue(decrease)],
     [_neuralNetworks: currentEmotion(buyer, happy),
     neuralNetworks: currentEmotion(seller, sad) ], _) .
     plan(
     updateDecision,
40      [action(defineNextValue(increase)],
41      [_neuralNetworks: currentEmotion(buyer, happy)],_).
42
43 ! _neuralNetwork X :- communication imageSensor(X).
44 ! _auxiliary  X :- communication negotiationPerception(X).
45 ! beliefs  X :- communication textSensor(X).
```

Code A.2 – The initial mental state of the mediator agent implemented in the Sigon framework

In listing A.2, the beliefs context has information about the current state of the negotiation. The desires and intentions contexts define which goal the agent will try to achieve in the current reasoning. It is worth mentioning that an agent can have different desires and intentions. However, for the sake of simplicity, we omit this process. The neural network context has a strategy that can detect emotions based on facial expressions pictures. In this case study, we focused on the phase when the negotiation is stalled, and the mediator agent should provide a new proposal. Taking that into consideration, the mediator agent detects that both parties have neutral emotions. To provide a new proposal, the agent uses its actuator to retrieve information about the item.

Code A.3 presents the next cycle of the mediator's reasoning. In this cycle, the agent will process the information retrieved by its web-scraper actuator and update the knowledge of the auxiliary context. Firstly, the bridge-rule in line 25 will be activated, adding the perception processed by the *negotiationPerception* sensor into the auxiliary context. The auxiliary context uses the strategies presented in Section A.2, where the agent removes the outliers and uses the mean value of the retrieved item. Based on the current state of the contexts, the agent then executes the plan of proposing a new deal, presented in line 18.

```
 1 communication:
 2     sensor("textSensor", "integration.TextSensor").
 3     sensor("negotiationPerception", "integration.WebScraperPerception").
 4     sensor("imageSensor", "integration.ImageSensor").
 5     actuator("defineNextValue", "integration.TextActuator").
 6     actuator("findInformation", "integration.WebScraper").
 7
 8 _auxiliary:
 9  retrievedPrice(659).  // agent proposes a new value based on the retrieved
10
11 desires:
12  updateDecision.
13
```

```
intentions :
 updateDecision.

planner:
     plan(
     updateDecision,
     [action(defineNextValue(create)],
     [_neuralNetworks: currentEmotion(buyer, sad),
     _neuralNetworks: currentEmotion(seller, sad),
     _auxiliary :  retrievedPrice (_) ], _).

! _neuralNetwork X :– communication imageSensor(X).
!  _auxiliary  X :–  communication negotiationPerception(X).
!  beliefs  X :–  communication textSensor(X).
```

Code A.3 – The mental state of the mediator agent during the next reasoning cycle

In the next cycle, the agent's neural network context detects that both parties are happy with the previous proposed value. The mediator sends a message offering to keep the current value and ends the negotiation. Since the primary goal of this case study is to present how an agent can use a different type of information during decision-making, we decided to omit some steps of negotiation protocol and information about the item.

In this case study, we showed some relevant steps of the mediator's reasoning cycle. We employed information retrieval, how to process different data types, integration with other existing contexts, and how advice can be created and proposed to the involved parties. We also investigated situations in which using only a single resource or reasoning, such as NN for facial expression recognition, is insufficient. Our agent can use a different mechanism, such as information retrieval, to acquire new knowledge about the environment to mitigate this limitation.