MSc in Data Science
Project Report
2022/23

"Logical Rule Integration into Neural Networks"

Author – Aarti Suryawanshi
Supervisor – Dr. Tillman Weyde

Submission Date – 24th December 2023

Declaration

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work, I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.
Signed – Aarti Suryawanshi

Abstract

This dissertation introduces an innovative approach to augment neural networks by integrating logical rules, aiming to enhance their decision-making and interpretability. Leveraging the zoo dataset, the study developed two novel models: the Partial Logic Neural Network and the Full Logic Network. These models demonstrate the effective integration of empirical data processing with logical reasoning, offering new insights into neural network optimization.The Partial Logic Neural Network exemplifies the balance between empirical data analysis and logical reasoning, achieving an integration of these elements with MSE and L1 loss functions. In contrast, the Full Logic Network is designed to adhere strictly to pre-defined logical rules, emphasizing consistency in the network's inferences. Its design prioritizes logical consistency, making it particularly suitable for tasks requiring clear rule-based decision-making. This model employs ReLU activation functions and a unique logic loss function during training, ensuring fidelity to the encoded logical rules.The research represents a significant advancement in the field of neural-symbolic AI, showcasing the potential applications of these enhanced models in sectors like healthcare and finance, where logical decision-making is vital. The findings suggest new research avenues in diverse datasets and neural architectures, proposing significant implications for future AI advancements. By integrating logical reasoning into neural networks, the study opens up possibilities for developing AI systems that are not only efficient and adaptable but also transparent and interpretable.The implications of this research are far-reaching, promising to elevate the role and impact of AI in various complex decision-making scenarios.

Keywords : Logical Integration, Neural Network Optimization, Decision-Making AI, Data Interpretability, Neural-Symbolic Integration

Table of Contents:

1. Introduction

In our dissertation, we embark on an exploratory journey into the intricacies of neural networks within machine learning. These networks have been instrumental in driving advancements in various applications like image processing, speech recognition, and financial forecasting. Their unparalleled ability to parse through and make sense of voluminous datasets has positioned them at the forefront of technological innovation.

Despite their impressive capabilities, neural networks grapple with several challenges. A primary limitation lies in their heavy dependence on extensive, well-labeled datasets. This reliance poses substantial challenges, mainly when data is scarce, expensive to obtain, or difficult to annotate accurately. Such issues are prevalent in sensitive areas where data privacy and security are crucial. This dependence on large volumes of data can hinder the versatility and deployment of neural networks in diverse settings.

Furthermore, neural networks often struggle with tasks requiring logical reasoning and abstract decision-making. While these networks excel in identifying patterns and making predictions based on historical data, their proficiency in applying abstract concepts and rules is markedly less developed. This limitation is acutely evident in situations that demand an understanding of causality or decision-making based on complex rule sets. Hence, neural networks often falter in scenarios where a deeper comprehension of underlying principles or logical structures is essential.

Furthermore, neural networks often need help with tasks requiring logical reasoning and abstract decision-making. While these networks excel in identifying patterns and making predictions based on historical data, their proficiency in applying abstract concepts and rules needs to be more developed. This limitation is acutely evident in situations that demand understanding causality or decision-making based on complex rule sets. Hence, neural networks often falter in scenarios where a more profound comprehension of underlying principles or logical structures is essential.

In addition, a significant issue with neural networks, especially those based on deep learning models, is their lack of interpretability. These 'black box' models, though efficient in producing results, often provide little clarity on how specific conclusions or decisions are made. This opacity can be a major barrier in domains where understanding the decision-making process is as crucial as the outcomes, such as in medical diagnostics, legal decision-making, and financial risk assessment.

In addition, a significant issue with neural networks, especially those based on deep learning models, is their lack of interpretability. Though efficient in producing results, these 'black box' models often provide little clarity on how specific conclusions or decisions are made. This opacity can be a significant barrier in domains where understanding the decision-making process is as crucial as the outcomes, such as medical diagnostics, legal decision-making, and financial risk assessment.

To surmount these challenges, our dissertation puts forth an innovative approach: integrating prior knowledge and logical rules into feedforward neural networks. This strategy aims to enhance the robustness and generalization capabilities of neural networks, making them more adaptable and efficient in situations where data availability is limited, or the application of logical reasoning is paramount.

To surmount these challenges, our dissertation proposes an innovative approach: integrating prior knowledge and logical rules into feedforward neural networks. This strategy aims to enhance neural networks' robustness and generalization capabilities, making them more adaptable and efficient in situations where data availability is limited or applying logical reasoning is paramount.

We hypothesize that combining prior knowledge and logical rules into neural networks could offer several advantages. Firstly, it could significantly reduce the network's reliance on large datasets. By incorporating existing knowledge and logical structures, the network could utilize this information to make informed decisions, even with limited data. This method could be especially beneficial in areas where data collection is arduous or privacy concerns limit data availability.

Using prior knowledge and logical rules in neural networks could offer several advantages. Firstly, it could significantly reduce the network's reliance on large datasets. The network could utilize this information to make informed decisions by incorporating existing knowledge and logical structures, even with limited data. This method could be especially beneficial when data collection is arduous, or privacy concerns limit data availability.

Secondly, the incorporation of logical reasoning into neural networks has the potential to bridge gaps where traditional models fall short. By embedding structured rules and reasoning processes into the network, it could be enabled to perform tasks that require logical operations, abstract thinking, and rule-based decision-making. This enhancement could considerably expand the scope of neural networks, allowing them to tackle problems involving intricate decision-making processes and causal reasoning more effectively.

Moreover, this approach could significantly improve the interpretability and transparency of neural networks. By integrating logical rules that are comprehensible and explainable, the decision-making process of these networks could become more transparent. This improvement is vital in applications where understanding the rationale behind decisions is essential for trust, compliance, and ethical considerations.

To empirically validate our hypothesis and demonstrate the practicality of our proposed approach, we utilize the zoo dataset from the Zoo - UCI Machine Learning Repository as a case study. With its diverse and complex nature, this dataset mirrors the challenges often encountered in real-world data. It provides an ideal scenario to test the effectiveness of integrating logical rules and prior knowledge into neural network frameworks. While adept at pattern recognition, this dataset will illustrate how traditional neural networks might struggle with nuanced categorization and prediction tasks that require a blend of data processing and logical reasoning.

As we advance through our research, we anticipate facing challenges and learning opportunities. These will encompass the technical aspects of integrating logical reasoning into neural networks and exploring the ethical and practical implications of such integration. Our aim is to document these experiences thoroughly, contributing valuable insights to the broader discourse on the future of AI and neural networks.

We anticipate facing challenges and learning opportunities as we advance through our research. These will encompass the technical aspects of integrating logical reasoning into neural networks and exploring such integration's ethical and practical implications. We aim to document these experiences thoroughly, contributing valuable insights to the broader discourse on the future of AI and neural networks.

In conclusion, our dissertation represents a significant stride towards enhancing the field of neural networks. By bridging the gap between traditional data-driven models and the necessity for logical reasoning, we struggle to develop neural networks that are not only more efficient and adaptable but also transparent and trustworthy. This research is expected to unlock new avenues for the application of AI, making neural networks more relevant and applicable in a variety of complex and nuanced domains.

1.1. Problem Background

Our research examines the limitations of neural networks within machine learning, particularly their reliance on extensive, labeled datasets and their struggle with tasks requiring logical reasoning and complex decision-making. At the same time, neural networks excel in areas like image recognition and predictive analytics, their dependency on large datasets and difficulties in applying rule-based logic present significant challenges, especially in fields where nuanced decisions are critical.

Our dissertation addresses these challenges by proposing a novel approach: integrating logical rules and prior knowledge into feedforward neural network structures. This integration reduces reliance on vast datasets and enhances the networks' ability to perform complex reasoning. This approach will lead to efficient training, improved decision-making capabilities, and broader applicability of neural networks in various sectors. Our goal is to expand the functional scope of neural networks, making them more adaptable and effective for diverse real-world applications.

1.2. Research Questions

Our study's primary goal is "Is it possible to successfully embed logical rules into the learning model of neural networks?”

1.3. Objectives

1. Select and Adapt a Scheme for Implementing Logic Rules in Neural Network Weights : The primary objective of our study is to identify and adapt an appropriate scheme that enables the integration of logic rules into the weights of a neural network. This process includes a thorough review and assessment of existing methodologies that facilitate embedding of logical structures within neural networks. The criterion for success in this objective is the identification of a suitable scheme and its successful adaptation to the neural network for the selected task and dataset.

2. Implement the Scheme in PyTorch : Once the scheme is selected, the next objective is to implement it using PyTorch, a popular open-source machine learning library. The implementation must be functional, efficient, and compatible with the chosen neural network model. This objective's criterion is the development of functional PyTorch code that effectively integrates the chosen scheme into the neural network.

3. Select a Suitable Dataset Associated with Logic Rules : The third objective is to select an appropriate dataset inherently associated with logical rules. This dataset should be relevant to the task at hand and allow for the practical application and evaluation of the integrated logic rules within the neural network. The criterion for this objective is the identification and selection of an appropriate dataset along with its associated logic rules.

4. Apply the Scheme and Evaluate the Results : The final objective involves the application of the selected scheme to the chosen dataset using the developed PyTorch implementation. This step includes conducting experiments to assess the effectiveness of the logic rule integration in the neural network. The criterion for this objective is the comprehensive documentation and evaluation of the experiments conducted, focusing on the impact of logic rules on the neural network's performance.

1.4 Beneficiaries

The outputs from this project are poised to benefit a wide array of stakeholders across various sectors, underlining the far-reaching impact of our work.

1. Academic Researchers: Scholars and researchers in machine learning and AI will find our dissertation and accompanying resources immensely valuable for furthering their research. The in-depth exploration of logical rule integration into neural networks is likely to spur new investigative and innovative directions. The availability of open-source code and datasets enables other researchers to build upon our findings, enriching the collective knowledge base in the field.

2. Industry Practitioners: Professionals in data-driven industries stand to gain significantly from the advanced neural network models developed in our project. In sectors like finance, healthcare, and technology, where the accuracy and interpretability of AI models are paramount, our enhanced models offer notable advantages. Fusing logical rules into neural networks is set to bolster these models' performance, reliability, and transparency, leading to more informed and effective decision-making.

3. Educators and Students: Our project's educational materials and interactive tools are invaluable for educators and students alike. These resources demystify complex machine-learning concepts, making them more accessible and comprehensible. They offer practical experience, enhancing students' skill sets and preparing them for future challenges in the field.

4. General Public: The broader implications of our research extend to the general public, especially as AI increasingly permeates various facets of everyday life. The advancements in neural network technology and the enhanced transparency and reliability of these models are expected to foster greater trust and acceptance of AI in societal applications.

In conclusion, our project's outputs represent a significant contribution to the realms of machine learning and AI. By offering a comprehensive array of resources, from in-depth research papers to practical tools and educational materials, we aim to propel the understanding and application of logically enhanced neural networks. The broad spectrum of beneficiaries, ranging from academic circles to industry professionals, educators, policymakers, and the general public, underscores the extensive impact and relevance of our research efforts.

1.5 Intended Method

In our dissertation, we explore techniques to enhance interpretability in neural networks by assimilating symbolic knowledge representations. Specifically, we integrate structured logical rules from the zoo animal traits dataset into model parameters and training processes using weight initialization and logic loss functions. Comparative assessments measure resulting improvements in transparency, generalization capability, and efficiency against standard neural counterparts. The research aims to advance model accountability through explicitly embedding domain expertise into data-driven learning for trustworthy AI.

1.Data Selection
- We have selected the zoo dataset for its structured categorization of animals, making it ideal for applying logical rules within a neural network framework.

2.Encoding Logic in Neural Networks

- Our approach involves converting the zoo dataset's categorical attributes into a format that feedforward neural networks can process, ensuring logical reasoning is encoded into the network's decision-making process.

3.Model Selection
- We chose a feedforward neural network due to its simplicity, effectiveness in pattern recognition, and suitability for integrating logical rules. This model aligns with our objective to enhance decision-making processes in AI systems using structured logical reasoning.

4.Training and Evaluation
- The NN will be trained on the preprocessed zoo dataset, with an emphasis on accurately applying the encoded logical rules. The evaluation will focus on the network's ability to correctly classify animals based on these rules, assessing its accuracy and effectiveness in logical reasoning.

1.6 Project Report Outline

Here is an outline for the project report chapters adapted to our dissertation context:

Chapter 1: Provides background on neural networks, defines limitations addressed, and establishes research questions, objectives and beneficiaries.

Chapter 2: Surveys prior work on integrating logic in neural networks, relevant techniques like fuzzy rules and probabilistic logic, and benchmarks.

Chapter 3: Details the stepwise methodology including zoo dataset selection, logical rule creation, propositional logic translation, network configuration with weight initialization and loss functions, training procedures and evaluation metrics.

Chapter 4: Presents results of model transparency analyses, reasoning evaluation, accuracy and efficiency comparisons to standard neural networks, along with performance benchmarking.

Chapter 5: Discusses interpretation of results, reflecting on techniques that worked well versus limitations encountered, and implications for future research.

Chapter 6: Summarizes contributions made in addressing objectives, proposes potential pathways to further advance logical rule integration, and concludes with final remarks.

2. Background and Context

In the field of ML, neural networks have established themselves as pivotal pattern recognition algorithms. This dissertation focuses on their application in analyzing the intricate patterns present in zoo animal data. Specifically, the study employs feedforward neural networks (FFNNs) due to their structured processing of data, which aligns well with the project's requirements.

2.1. Feed Forward Neural Networks

This paper (Seiya Satoh, Yamagishi and Takahashi, 2023) is highly pertinent to our dissertation, which investigates the application and analysis of Feed-Forward Neural Networks (FFNNs) in logic modeling. The innovative approach of using Independent Component Analysis (ICA) on FFNNs' hidden layers, as presented in the paper, offers a unique perspective on network comparison and analysis. This methodology is particularly relevant to our work, as it provides a deeper understanding of the internal dynamics of FFNNs. By applying these insights, we can better comprehend how FFNNs process and model logical functions, enhancing our ability to assess their performance and potential for complex logical operations. This aligns seamlessly with our dissertation's objectives of exploring the capabilities and limitations of FFNNs in intricate logical computations.

In our dissertation, we explore the integration of logic rules into FFNNs. The work of (Rajasegaram D. 2023), provides valuable context for this. (Rajasegaram D. 2023) approach to embedding logical structures into neural networks, specifically in the context of weights and biases, is highly relevant. It gives us insights into the practical application of logic rules in FFNNs and the potential impact on network performance. This alignment offers a foundation for our research, enriching our understanding of how logical rules can enhance neural network efficiency.

We aim to integrate logic rules into the weights of neural networks and evaluate their performance. The paper by (Daróczy, B., Alekszíev, R. and Benczúr, A., 2018) provides valuable insights into the hierarchical structure and sparse representations of feedforward neural networks. This research aids our understanding of network architectures and optimization, critical for embedding logic rules efficiently. It informs our approach, particularly in adapting methodologies for logic rules integration and optimizing them in PyTorch, enhancing the practicality and effectiveness of our work.

The research from these papers contributes to our dissertation's goal of integrating logical rules into neural networks using Feed-Forward Neural Networks (FFNNs).These studies provide critical insights into the architecture and application of FFNNs, guiding us in refining neural network applications for more efficient and adaptive AI solutions.

2.1.1. Architecture

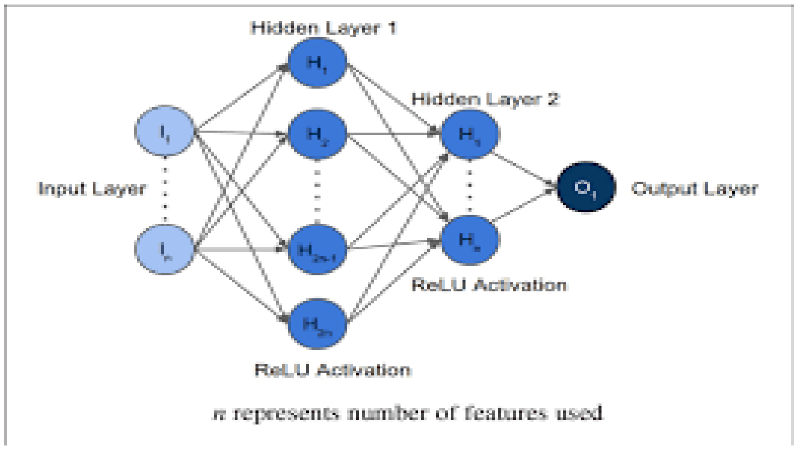


Fig 1. : Simple Feedforward Neural Network with ReLU Activation

In exploring neural network architectures, the feedforward neural network (FFNN) serves as a foundational model for understanding complex data processing. Figure 1 in the report illustrates a typical FFNN structure, which is composed of several distinct layers: an input layer, multiple hidden layers, and an output layer.

At the beginning of the forward pass in the neural network, the input layer receives the initial data. This data is represented by 'n', indicating the number of features each input possesses. From here, the data is propagated forward, first through Hidden Layer 1, where initial transformations are applied via weighted connections. The process continues to Hidden Layer 2, further refining the data transformations. Each hidden layer utilizes the Rectified Linear

Unit (ReLU) activation function, which introduces non-linearity to the model, enabling it to capture and learn more complex patterns than linear models can.

The final step in the data's journey is the output layer, where the network concludes its processing and produces the results. The unidirectional flow from input to output, passing through layers without cycles or feedback loops, is a defining characteristic of FFNNs. This architecture allows for straightforward modification and scaling, depending on the complexity of the task at hand.

Incorporating this FFNN model into the critical context of our study emphasizes the significance of network architecture in their accurate classification and prediction of data patterns, a central theme to our research objectives. As we delve into the application of FFNNs for analyzing zoo animal data, understanding this architecture will be crucial for interpreting how the network processes information and arrives at its classifications.

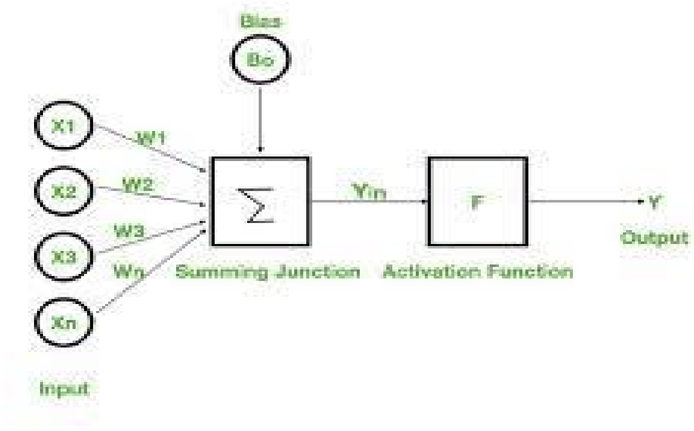


Fig 2. : Basics of Neuronal Function in Neural Networks

2.1.2. ReLU Function

The research by (Riegel et al., 2020),delves into the nuanced functionality of the ReLU activation function. The study elucidates how ReLU, expressed mathematically as ReLU(x)=max(0,x), functions effectively as a concept detector within neural networks. The authors highlight ReLU's ability to maintain linear behavior for positive inputs, thereby signifying the presence of features, while zeroing out negative inputs, uniformly indicating their absence. This dual functionality, as the authors note, is instrumental in reducing noise and preventing the simultaneous activation of correlated neurons. Additionally, the paper discusses the computational benefits of ReLU, particularly its gradient sparsity, which optimizes computational efficiency by necessitating gradient calculations for only a subset of neurons.

The same paper by (Riegel et al., 2020),further examines the regularization capacity of ReLU in neural network training. The authors describe ReLU's ability to saturate to an output and gradient of zero as a form of information and gradient dropout, a mechanism vital for mitigating overfitting in neural network models. This aspect of ReLU's functionality is highlighted as crucial for maintaining model generalizability and preventing overtraining on specific data patterns.

Paper authored by (Geiger et al., 2022), ReLU's application is showcased in different neural network architectures. The paper describes the use of ReLU in a single-layer network computing equality and in a recurrent LSTM network for sequence generation. These implementations demonstrate ReLU's adaptability and effectiveness in various network designs, a critical aspect for the dissertation's focus on integrating logical rules into neural networks. The authors' work underscores ReLU's capacity to facilitate complex computational tasks while maintaining activation function simplicity.

In conclusion, the insights provided by (Riegel et al., 2020) and (Geiger et al., 2022) in their studies offer a comprehensive understanding of ReLU's multifaceted role in neural networks. These findings underline the significance of ReLU in enhancing computational efficiency, noise reduction, regularization, and adaptability in various network architectures, making it a pivotal component in the field of neural network development and the integration of logical rules.

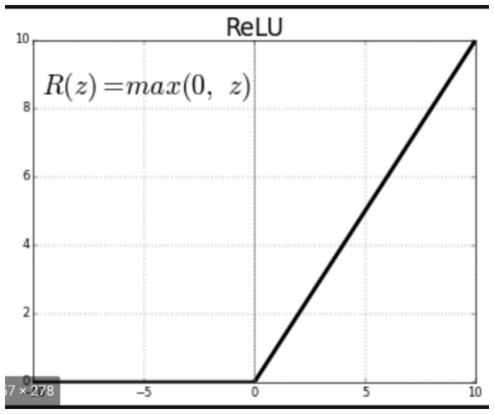


Fig 3. : ReLU Activation Function

Fig 3 depicts the graph of the Rectified Linear Unit activation function, which is mathematically represented as ReLU(z)=max(0,z). The graph shows that for inputs less than zero, the output is zero, and for inputs greater than zero, the output is equal to the input, demonstrating the piecewise linear nature of ReLU.

2.1.3. Learning Algorithm

In the aim to integrate logical rules into neural networks, learning algorithms serve as the pivotal mechanism that drives the adjustment and tuning of the networks' parameters. Paper by (Shihabudheen and Pillai, 2018) , explores a spectrum of algorithms that underpin this process. The backpropagation algorithm, recursive least squares, Levenberg–Marquardt (L-M) algorithms, and gradient descent methods are fundamental to the iterative learning process. These algorithms not only support error minimization but also enable the hierarchical and recursive structuring of learning, which is paramount when embedding logical rules into neural network architectures.
Paper by (Rajasegaram D., 2023), provides a practical blueprint for the application of these algorithms in a neural network's learning cycle. It involves the forward propagation of data, backpropagation of errors, and hyperparameter tuning. Such a structured approach is critical for a dissertation focused on logical rule integration, as it outlines a clear methodology for adapting the neural network to accommodate and optimize rule-based decision-making processes.

Moreover, the k-means clustering and Extreme Learning Machine (ELM) algorithm, as discussed in (Shihabudheen & Pillai, 2018), facilitate an efficient data structuring method that is particularly beneficial when rules are integrated into neural networks. By clustering the data, we can enhance the ELM's capacity to identify the 'if' components of rules, streamlining the process of integrating these rules into the neural network's learning framework. The ELM's rapid training capabilities are essential for neural networks that require fast adaptation to new logical rules and the swift recalibration of their decision boundaries.

2.2. Prior Knowledge

2.2.1. What is Prior Knowledge ?

In (Rajasegaram D., 2023) study, "What is Prior Knowledge" is extensively discussed, providing a foundational understanding for enhancing the efficiency and effectiveness of neural network models. Prior knowledge refers to any pre-existing information or insights that are directly related to the problem domain being addressed by the neural network. This can encompass a range of information, from specific details about the data itself to more abstract concepts or output variables that the neural network is intended to predict or classify.

The role of prior knowledge in neural network training is multifaceted. Firstly, it serves as a guiding framework that informs the initial structuring and tuning of the network. By incorporating relevant information from the onset, neural networks can bypass certain initial learning stages, thereby accelerating the training process. This is particularly beneficial in complex domains where data can be vast and varied, as the initial guidance provided by prior knowledge helps the network focus on relevant patterns and associations from the beginning.

Furthermore, the integration of prior knowledge aids in overcoming some of the intrinsic limitations of neural networks, such as their tendency to be 'black box' models. By embedding known rules, patterns, or principles within the network's architecture, the model's decisions and outputs become more interpretable and transparent. This is crucial in applications where understanding the rationale behind the network's decisions is as important as the accuracy of these decisions, such as in healthcare diagnostics or financial analysis.

Prior knowledge can take multiple forms depending on the application. In a sentiment analysis model, for instance, it could include linguistic cues known to express sentiments, or in a medical diagnosis tool, it might involve symptoms and signs commonly associated with certain diseases. This knowledge not only streamlines the learning process but also enhances the network's ability to make accurate and reliable predictions.

In summary, prior knowledge acts as a critical asset in neural network development. Its incorporation leads to more efficient training, improved interpretability, and enhanced reliability of the network's outputs. As neural networks continue to evolve and find applications in increasingly complex domains, the strategic integration of prior knowledge will become an indispensable aspect of their development and deployment.

2.2.2. Benefits of incorporating Prior Knowledge

The incorporation of prior knowledge into neural networks, as explored by (Rajasegaram D., 2023), presents a transformative approach in the realm of artificial intelligence and neural network design. This concept is vital for the advancement of neural networks beyond traditional data-driven models, equipping them with the capability to process information in a more human-like, logical, and systematic manner.

1. Structured Foundation for Learning: The project underscores that prior knowledge acts as a foundational framework for neural networks, instilling a more structured and systematic approach to problem-solving. This is akin to the way the human brain processes high-level information in conjunction with low-level sensory inputs. By embedding established principles and domain-specific knowledge, neural networks can benefit from a guided learning trajectory, which is especially beneficial in applications that require adherence to logical structures or domain-specific rules.

2. Enhanced Generalization and Interpretability: One of the critical benefits outlined in the project is the enhancement of the neural network’s generalization capabilities. This is achieved by integrating prior knowledge, which can be particularly effective even with sparse and noisy datasets. Additionally, this integration enhances the interpretability of neural networks. By incorporating rules or constraints, neural networks can offer clearer insights into their decision-making processes, improving their transparency and making them less of a 'black box'. This aspect is crucial for applications requiring clarity and justification in AI decision-making, such as in medical diagnostics, financial analysis, and legal assessments.

3. Balancing Symbolic and Data-Driven Models: The project also discusses the reciprocal benefits of combining symbolic (rule-based) models with data-driven neural network models. While symbolic models excel in transparency and require minimal data, they often lack adaptability and struggle with noisy data due to their reliance on predefined rules. On the other hand, data-driven models excel in handling complex data patterns but can lack transparency. By merging these two approaches, neural networks can be developed to capture complex relationships present in the data while retaining the clarity and precision offered by rule-based systems. This hybrid approach opens up new avenues in AI, allowing for the creation of models that are both adaptable and interpretable.

4. Efficiency and Prediction Accuracy: Another significant benefit noted in the project is the efficiency and accuracy of predictions made by neural networks imbued with prior knowledge. The project demonstrates this through the use of synthetic datasets, where neural networks, despite limited training samples, were able to achieve strong generalization abilities. This tells that the integration of prior knowledge can lead to more efficient models, capable of making accurate predictions with less data. This is particularly advantageous in scenarios where data availability is limited or in situations requiring quick model training without compromising accuracy.

2.2.2.1 Introduction to Propositional Logic

In our project, we're exploring how to enhance computer brain models, known as neural networks, by integrating them with logical reasoning. This concept is at the heart of several key studies we've examined. For instance, (Badreddine et al., 2022) delves into the use of logic within neural networks. The study shows how incorporating logic into advanced AI systems like DeepProblog enhances their efficiency and smartness in processing information. This directly supports our dissertation's objective to improve neural networks' intelligence by integrating logical rules, making them more effective and sophisticated in their operations.

Moreover, (Reimann et al., 2022) in their study, introduce the Neural Logic Rule Layers (NLRL) approach, which aligns significantly with our study's focus on propositional logic. NLRLs innovatively merge logical rules with neural networks, creating a synergy between deep learning and logical reasoning. This method is particularly relevant to our research, demonstrating how data can inform the learning of logical rules. The diverse applications of NLRLs, such as in image classification and function approximation, illustrate the practical utility of logic across different fields. This blend of logic and deep learning not only complements but also enriches our exploration of propositional logic in our dissertation.

(Badreddine et al., 2022) and (Reimann et al., 2022), have shown that the combination of logical reasoning and neural networks is very promising. By adding logical rules to neural networks, we're not just making them smarter, we're also giving them more possibilities to use. This combination of deep learning and propositional logic will lead to AI systems that are better at making decisions, more adaptable, and able to handle complex problems.

2.2.2.2 Logical Connectives

In our dissertation, we focus on integrating logical connectives, specifically the AND and NOT logic gates, into neural networks, drawing from a range of academic papers. This integration is pivotal for enhancing AI systems with sophisticated decision-making and reasoning capabilities, aligning closely with our project's aims.

Study authored by (Riegel et al., 2020)),enriches this understanding by discussing the requirements for activation functions in Logical Neural Networks (LNNs), particularly for the AND and NOT connectives. It emphasizes the need for these functions to adhere to logical principles, ensuring the integrity and reliability of the neural network's reasoning processes. This insight directly contributes to our project, as it supports our aim to integrate structured logical reasoning into neural network frameworks, thus enhancing the decision-making capabilities of AI systems.

(Rajasegaram D., 2023), provides a practical perspective on the application of propositional logic, particularly highlighting the roles of AND and NOT logical connectives. This project underscores the importance of these connectives in constructing logical statements and their applications in AI. The focus on AND and NOT aligns with our project's scope, as we aim to incorporate these specific logical operations into AI and machine learning models.

To illustrate with our zoo dataset, consider using the AND logic gate for classification. For example, we might classify an animal as a mammal if it has hair AND gives birth to live young. Similarly, the NOT logic gate could be used to identify non-avian species by determining if an animal does NOT have feathers. These examples demonstrate the practical application of AND and NOT connectives in creating classification rules, enabling AI systems to derive accurate and logical inferences from the data.

A (Input)	B (Input)	A AND B	NOT A
0	0	0	1
0	1	0	1
1	0	0	0
1	1	1	0

Table 1: Basic Logic Gate Operations

In this table 1:

- For the AND gate (A AND B), the output is true (1) only when both A and B are true.

- For the NOT gate (NOT A), the output is the inverse of A; when A is true (1), NOT A is false (0), and vice versa.

2.3. Related work

This section describes our method for reviewing related work in neural networks and logical rule integration. It details the criteria for literature selection, focusing on studies that inform neural-symbolic AI and the integration of logical reasoning in data-driven models. This section identifies crucial developments and research gaps our study addresses.

2.3.1. Method

Our methodological approach in this dissertation significantly draws upon the foundational research conducted by (Weyde & Kopparti, 2021). Their exploration into the limitations of neural networks, particularly in the context of generalization beyond training data, has informed the core strategies of our project. While they have made notable strides with the Relation Based Pattern (RBP) approach, our work seeks to augment this by transitioning from manual integrations to an automated system that embeds logical rules into the weight and bias matrices of neural networks.

In advancing the RBP method, we have innovated a system that automates the translation of animal attributes from the zoo dataset into propositional logic rules. This step is pivotal, as it converts raw data into structured logic that neural networks can decipher and learn from. For example, the logical statement "If an animal has feathers and does not produce milk, then it is classified as a bird" is transformed into a format that the neural network can utilize for classification tasks.

Our process elevates the manual methodologies previously utilized, streamlining the incorporation of logical rules into neural networks, and ensuring a more consistent and scalable rule application. The resultant neural network models are anticipated to excel in their generalization capabilities, equipped to integrate complex logical patterns derived from structured datasets like the zoo database.

By implementing this method with the zoo dataset, we demonstrate the practical application and efficiency of our automated approach. The code developed for this project clearly illustrates the utilization of logical operators such as AND and NOT in constructing logical rules, which the neural network adopts, enhancing its decision-making processes.

Thus, the work of (Rajasegaram D., 2023), acts as both the inspiration and the springboard for our advanced research. Our dissertation extends the principles of the RBP approach by automating the encoding of logic into neural networks, a leap forward from the manual, labor-intensive processes of the past. We aim not only to mirror the success of the RBP method but to surpass it, enabling neural networks to dynamically and robustly engage with logical constructs, as evidenced by our handling of the zoo dataset.

2.3.2. Modeling logic in Logical Neural Networks

Paper by (Riegel et al., 2020), proposes an innovative neural network architecture called Logical Neural Networks (LNN) that incorporates symbolic logic directly into the neural components and connectivity. This is highly relevant for our dissertation on enhancing robustness of neural networks by integrating logical constraints. The fundamental units in LNNs are logical neurons. Each neuron maps one-to-one to an element of a logical formula such as AND, OR, NOT operations or domain concepts. For example, there may be neurons representing "feathers", "beak", OR, AND etc. that appear in the animal classification rules.

The activations of these neurons adhere strictly to the truth tables of the logical operations they represent. For instance, an AND neuron will output 1 only if both its inputs are 1, following the logic of conjunction. Similarly, a NOT neuron simply flips its input value to perform negation. The neurons are interconnected to form architectures that mirror symbolic logic formulas.In our dissertation for example, if the logic contains a rule "(feathers AND beak) IMPLIES bird)" then the LNN will wire together the respective neurons using connections representing the AND and IMPLIES relations.

Remarkably, despite modeling rigid symbolic logic, (Riegel et al., 2020) explained that LNNs are still differentiable end-to-end due to specially designed activations. This permits training them using backpropagation to improve performance just like regular neural networks.However, instead of standard loss functions, they require custom logical loss formulations. These losses penalize deviation from the logic constraints. By combining predictive losses with logical consistency losses, joint training can occur.

In our dissertation, the work of (Riegel et al., 2020) on Logical Neural Networks (LNNs) is highly pertinent. Their approach of integrating symbolic logic into neural networks through logical neurons aligns with our aim to enhance neural network robustness by embedding logical rules. LNNs, with neurons representing logical operations like AND, OR, NOT, allow for the direct application of logical rules within the network, mirroring symbolic logic structures. This architecture, maintaining differentiability for backpropagation training, resonates with our methodology of combining logical rules with neural network learning. The insights from LNNs provide a valuable reference for our dissertation, particularly in developing neural networks that are both logically robust and adaptable.

2.3.3. Neuro-Symbolic

In our dissertation, we explore the burgeoning field of Neuro-Symbolic AI, focusing on its potential to revolutionize the integration of logical rules into neural network architectures. This exploration is deeply informed by key findings from notable papers in the domain.

Additionally, our research is enriched by the contributions by (Garcez et al.,2015), which extols the virtues of neural-symbolic computation from both methodological and computational standpoints. The paper highlights the representational generality of this approach, capable of encompassing a broad spectrum of symbolic systems. The learning robustness inherent in neural-symbolic computation opens up exciting possibilities for knowledge representation, be it purely symbolic or a hybrid encompassing probabilistic or numerical representations. These insights are pivotal to our work, suggesting new methodologies for developing neural network models that are not only efficient in data processing but also capable of sophisticated symbolic reasoning.

Our dissertation, thus, seeks to build upon these studies, aiming to harness the power of neuro symbolic AI for enhancing neural networks' decision-making and reasoning capabilities. By integrating the principles of neural-symbolic computation, we aspire to create neural network models that are not only advanced in terms of learning efficiency but also embody the sophistication required for complex, symbolic reasoning tasks. The ultimate goal of our research is to push the boundaries of current neural network technologies, making them more versatile, interpretable, and adept at handling a confluence of learned data and logical constructs.

2.3.4. Regularization with Logic

In our dissertation, we explore the integration of logical rules into neural networks, particularly focusing on the concept of regularization with logic. This exploration draws on critical insights from seminal research papers in the field.

Our approach is significantly informed by the methodologies discussed in (Rajasegaram D., 2023) study, which delves into the integration of logic into neural networks through regularization. The project highlights the use of posterior regularization as a structural method to merge prior knowledge with neural network learning processes. This technique involves the inclusion of features corresponding to prior knowledge sources within a log-linear model, guiding the learning trajectory of the model. This concept resonates with our dissertation's aim of enhancing neural networks with interpretative and decision-making abilities, offering a model for embedding logical rules into the neural network's learning framework.

Furthermore, we examine the approaches proposed by (Reimann et al., 2022), which presents a unique perspective on semantic-based regularization. This paper contrasts its methodology with Markov Logic Networks, advocating for an approach that integrates rules directly into the hypothesis space. By biasing neural networks toward representing logical rules, this method facilitates an end-to-end learnable integration of these rules into neural network structures. This novel approach of learning rule types, the variables involved, and their relationships is particularly relevant to our research. It provides a practical and direct method for embedding logical rules into neural networks, aligning with our objective of automating the conversion of logical rules into neural network architectures.

These papers collectively provide a robust theoretical and methodological foundation for our dissertation. They guide our efforts in developing neural network models that are not only efficient in learning from data but also capable of sophisticated decision-making based on the integration of logical rules. This enhanced functionality is key to advancing neural networks' capabilities, especially in complex applications requiring nuanced reasoning and decision-making.

3. Methods

In this analysis of our methods, we study the comprehensive procedures and sophisticated techniques we employed to create a model which can predict using neural networks and propositional logic. Our project was driven by the ambition to accurately classify a diverse range of animals based on a series of distinct attributes. This ambitious struggle required accurate planning and execution across various stages, including data collection, preprocessing, detailed feature engineering, the design and implementation of neural network architectures, rigorous training, and thorough model evaluation. Each step was carefully crafted and executed, with the underlying aim of aligning with our central objective of creating a robust and precise classification system, all while adhering to the latest advancements and methodologies documented in contemporary machine learning and data science literature.

3.1. Data Selection and Rationale

This section focuses on why we chose the zoo dataset for integrating logical rules into neural networks. The dataset's variety is ideal for testing our models, the Partial Logic Neural Network and Full Logic Network. This section explains the dataset's relevance to our research objectives.

3.1.1. Utilizing the Zoo Dataset for Logical Rule Integration in Neural Networks:

The zoo dataset obtained from Kaggle and UCI is a rich and structured collection of animal attributes, classified into various types. Each row in the dataset indicates a distinct animal, characterized by a set of binary attributes that describe its physical and behavioral features. These attributes include, but are not limited to, hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, legs, tail, domestic, and catsize. Additionally, each animal is categorized into a class type, which helps in understanding their broader biological categorization.

This dataset is particularly suited for our research, where we aim to test the integration of logical rules into neural network architectures. The binary nature of the attributes simplifies the process of applying logical rules, as each feature can be directly used as a condition or a decision node within a logical framework. For example, the presence or absence of features such as hair or feathers can be used to infer the class type of the animal, mimicking the logical reasoning process found in traditional rule-based systems.

Moreover, the inclusion of diverse animal species in the dataset ensures a comprehensive test bed for our experiments. The variety in the dataset, ranging from mammals to birds and aquatic species, allows for the examination of the neural network's ability to learn and generalize across different biological classes. This is crucial in assessing the effectiveness of incorporating logical rules into neural networks, as it provides insights into the adaptability and robustness of the model when faced with diverse data inputs.

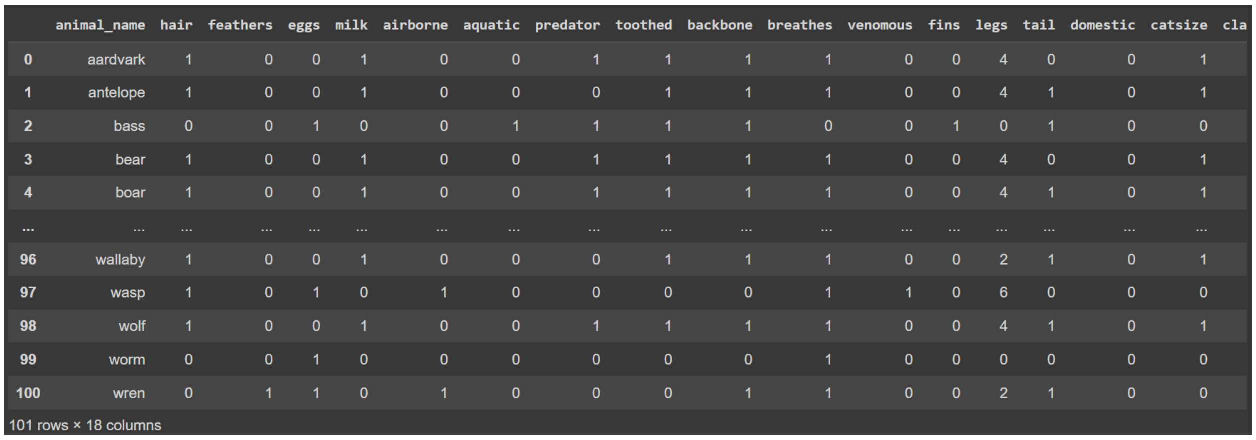
Furthermore, the structured format of the dataset facilitates easy pre-processing and manipulation of data, which is essential in the context of machine learning experiments. The clear definition of attributes and classes enables straightforward encoding and transformation of the data, ensuring that the focus remains on the integration and testing of logical rules rather than on data cleaning and preparation.

In addition to utilizing the inherent structure of the dataset, we undertook an accurate process of manually creating logical rules by thoroughly examining the dataset. This involved an in-depth analysis of the various attributes and their combinations, leading to the formulation of specific rules that capture the inherent relationships and patterns within the data. For instance, by analyzing trends and correlations among attributes like 'milk' and 'hair', we were able to construct logical rules that accurately classify mammals. This manual rule creation process not only deepened our understanding of the dataset but also provided us with a set of tailored rules that were directly integrated into the neural network, enhancing its ability to perform logical reasoning based on biological attributes.

In summary, the zoo dataset's structured nature, clear attribute definitions, and inclusion of a wide range of animal species make it an ideal choice for our study. It not only allows for direct application and validation of logical rules within neural networks but also provides a rich and varied data environment to refine these models.

3.1.2. Exploratory Data Analysis (EDA) of the Zoo Dataset

To begin our analysis we imported the dataset into a pandas DataFrame called "zoo_data". This DataFrame was carefully organized with column names that described characteristics such as 'hair' 'feathers 'eggs 'milk' and a categorical variable called 'class_type' to categorize each animal into classifications.



	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	cla
0	aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	
1	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	
2	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	
3	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	
4	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
96	wallaby	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	1	
97	wasp	1	0	1	0	1	0	0	0	0	1	1	0	6	0	0	0	
98	wolf	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	
99	worm	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	
100	wren	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	

101 rows × 18 columns

Fig 4: Sample Data from the Zoo Dataset

Preliminary Data Cleaning:

After analyzing the dataset we noticed that the initial row didn't align correctly with the data format and seemed to be a header. To maintain consistency and accuracy, in our data we decided to eliminate this row. This step is crucial to ensure the reliability of our analysis.

To make our dataset more comprehensive we included calculated features. First we converted the 'legs column into values to enable operations in our analysis. Additionally we introduced a feature called 'binary_legs to indicate whether animals have legs. This simplification helps us gain insights into the mobility of animals.

Furthermore we expanded our dataset by incorporating columns for each animal class. This allows us to easily perform class classification in future analyses. These columns provide an encoded representation of the distribution of animal classes, in our dataset.

Class imbalance problem:

In our project, addressing class imbalance in the dataset was a crucial step. Class imbalance refers to a situation where some classes are represented more frequently than others. This can lead to a neural network being biased towards the more common classes, potentially compromising the accuracy and generalizability of the model.

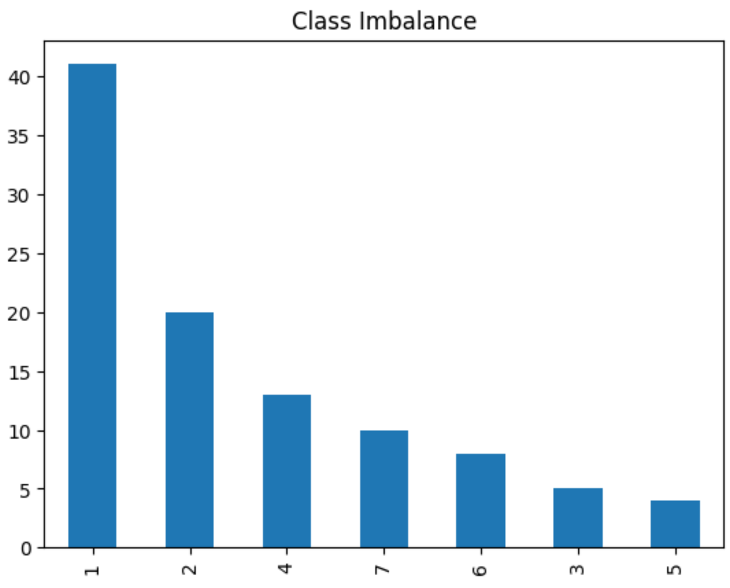


Fig 5. Imbalance Classes

To visualize the class imbalance, we first analyzed the distribution of different animal classes using a bar chart shown in fig 5. This helped us see which classes were overrepresented and which were underrepresented. The bar chart provided a clear picture of the imbalance, guiding us on how to address it.

To balance the classes in the dataset, we employed a technique called upsampling. This involves increasing the number of samples from underrepresented classes to match the number of samples in the most represented class. Upsampling works by randomly replicating instances from the less common classes, thereby equalizing the representation of each class in the dataset. This method is crucial as it ensures that our neural network doesn't become biased towards any particular class and can learn to classify all classes with equal accuracy.

After upsampling, we shuffled the balanced dataset to ensure that the order of data didn't influence the learning process. Shuffling helps in creating a more robust model by exposing it to a varied sequence of data points during training.

We chose to balance the dataset rather than downsample (reducing the number of instances in overrepresented classes) because downsampling can lead to a significant loss of valuable data. Since our aim was to create a model that can accurately classify all types of animals in the zoo dataset, maintaining as much data as possible was important, making upsampling the preferred choice.

In summary, addressing class imbalance by upsampling was a vital step in our project. It ensured that our neural network model learned from a balanced representation of all animal classes, ultimately leading to a more accurate and fair classification system. This approach aligns with best practices in machine learning, where the aim is to create models that perform well across all categories of data, not just the most common ones.

Visualization for Insights:

During our exploratory data analysis (EDA) we heavily relied on representations to uncover underlying patterns and connections present in the data.

1. Visualizing Class Distribution; We used a count plot (Fig 6) to show how animals are distributed across classes. This visualization played a role in helping us understand the frequencies of each class in our dataset.

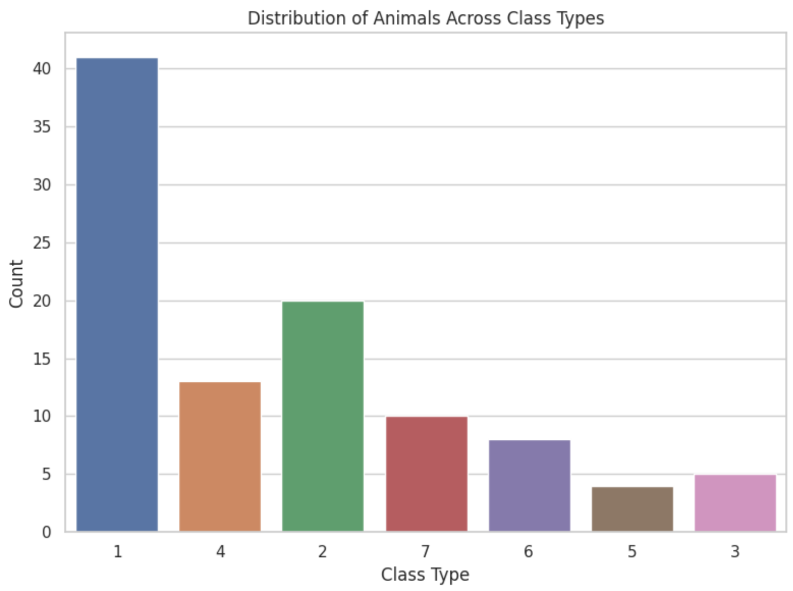
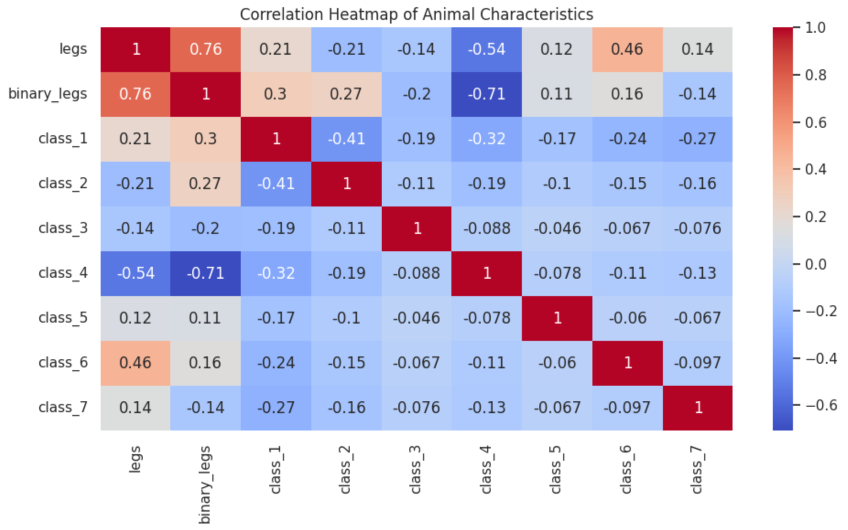


Fig 6: Distribution of Animals Across Class Types

2. Feature Correlation Exploration: Our exploration of feature correlations was enhanced by utilizing a correlation heatmap (Fig 7). This integral step allowed us to uncover any notable relationships or multicollinearity between features, which could have an impact on our models.



	legs	binary_legs	class_1	class_2	class_3	class_4	class_5	class_6	class_7
legs	1	0.76	0.21	-0.21	-0.14	-0.54	0.12	0.46	0.14
binary_legs	0.76	1	0.3	0.27	-0.2	-0.71	0.11	0.16	-0.14
class_1	0.21	0.3	1	-0.41	-0.19	-0.32	-0.17	-0.24	-0.27
class_2	-0.21	0.27	-0.41	1	-0.11	-0.19	-0.1	-0.15	-0.16
class_3	-0.14	-0.2	-0.19	-0.11	1	-0.088	-0.046	-0.067	-0.076
class_4	-0.54	-0.71	-0.32	-0.19	-0.088	1	-0.078	-0.11	-0.13
class_5	0.12	0.11	-0.17	-0.1	-0.046	-0.078	1	-0.06	-0.067
class_6	0.46	0.16	-0.24	-0.15	-0.067	-0.11	-0.06	1	-0.097
class_7	0.14	-0.14	-0.27	-0.16	-0.076	-0.13	-0.067	-0.097	1

Fig 7: Correlation Heatmap of Animal Characteristics

3. Hair Characteristic Analysis: We crafted a pie chart below (Fig 8) to explore the population of animals with and without hair, offering a visual depiction for quick understanding of this characteristic.

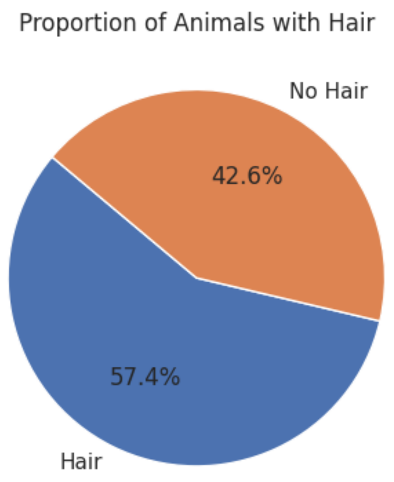


Fig 8: Proportion of Animals with Hair

4. Distribution of Leg Counts: We constructed a histogram (Fig 9) to display the distribution of leg counts among the animal species, providing a clear and revealing view of this physical characteristic

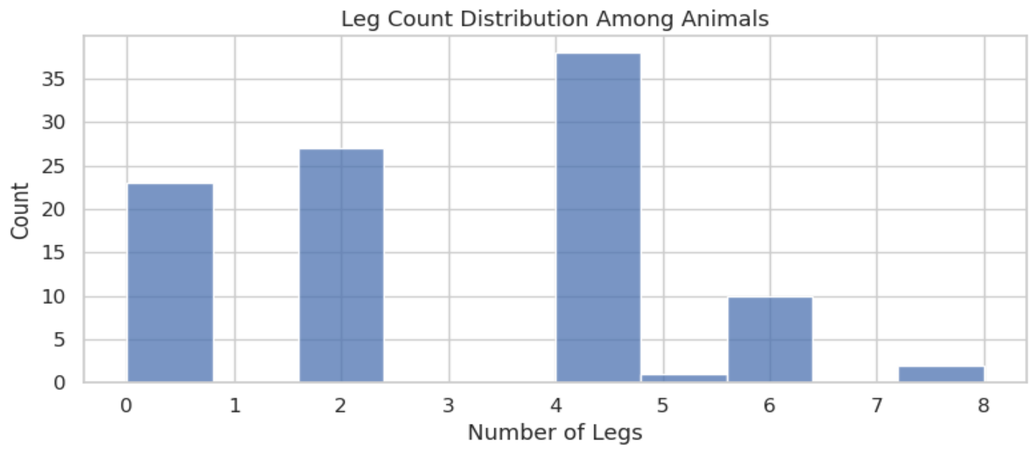


Fig 9: Leg Count Distribution Among Animals

Through our preliminary data examination, we established a strong foundation for our project, giving us invaluable insights into the intricacies of our dataset. These critical steps not only equipped us for the modeling stage, but also ensured that our methodology was rooted in a thorough grasp of the data. Our inclusion of visually engaging figures enhanced our report, making the data easily understandable and engaging for our audience.

3.1.3. Based on EDA Manual Rule Creation

We then embarked on the next step, which was to create a logical rule that is based on the attributes of the data set. We had to go deep into the available data and unravel unique patterns as well as associations existing among different animal attributes.

What we did here is finding specific characteristics particular to each class of animals. For instance, we found that birds carry feathers but do not give milk that gave rise to rules like “If an animal has feathers and does not produce milk, then it’s classified as a bird.” This simple rule provides for distinctions between birds and other classes of animals. The translation of these insights into logical rules for our neural network was not all about conversion of data. It was rather capturing what truly defines these animal classes in a language that could be understood by the network. Each rule described a path in the neural network from input traits to ultimate animal classification.

Creating rules was not simply a mechanical process, but rather, it allowed us to delve into the complex realm of biological categorization and connect the dots between unrefined data and automated understanding. It was an Intriguing fusion of biology, data analytics, and artificial intelligence, revealing the possibilities and obstacles of developing an AI system that comprehends and utilizes real-life reasoning. This pivotal step was a vital foundation for our project, paving the way for a neural network that possessed both logical and intelligent capabilities.

3.2. Model Development
In the pursuit of optimizing our neural network's performance on the zoo dataset, we engineered three separate models, each with a unique approach to integrating logical reasoning and feature selection:

3.2.1. Partial Logic Model
In our exploration of integrating logical reasoning with neural networks, we developed the 'Partial Logic Neural Network' to investigate a more nuanced approach to combining logic rules with data-driven learning. This model was designed to strike a balance between strictly adhering to predefined logical rules and allowing the network to learn patterns autonomously from the data.

Unlike the Full Logic Network which is explained further, which was constrained by rigid logic-based initialization, the Partial Logic Network employed a more flexible approach. We utilized a subset of the logical rules derived from the zoo dataset to inform the network's structure, but we did not strictly bind the network to these rules during training. This method allowed the network to benefit from the guidance of logical rules while retaining the freedom to learn and adapt based on the data.

Some key steps in creation of Partial Logic Model :

1. Rule Processing and Conversion: We began by processing the logical rules and converting them into a format suitable for integration into a neural network. This involved mapping attributes to propositional variables and formulating conditional statements.

2. Weight and Bias Masking: A crucial aspect of our approach was the implementation of weight and bias masking in the first layer of the network. We generated weight and bias masks based on the processed rules, which were then applied to the network's first layer. This allowed the network to partially reflect the logical rules in its initial state.

3. Network Architecture: The Partial Logic Network consisted of multiple linear layers with ReLU activations, allowing for the handling of complex features. The use of ReLU was instrumental in enabling the network to model nonlinear relationships in the data.

4. Loss Function Combination: For training, we combined Mean Squared Error (MSE) and L1 loss. This combination allowed us to measure the accuracy of the model's predictions (MSE) while also ensuring adherence to the logical rules to some extent (L1).

5. Hyperparameter Tuning and Training: We conducted hyperparameter tuning to find the optimal settings for learning rate, optimizer type, and epoch count. The training process involved a careful balance between minimizing the output loss and the logic loss, allowing the network to learn from the data while being partially guided by the logical rules.

6. Evaluation and Analysis: We assessed the model's performance on the validation set after training, paying particular attention to its accuracy and capacity to generalize outside of the training set.

Our goal in building the Partial Logic Network was to investigate how logical rules and data-driven learning may work together to improve the performance of a neural network. Our approach was guided by the hypothesis that a blend of logic and flexibility could lead to a more robust and adaptable model, especially when dealing with complex real-world datasets like the zoo dataset. The development of the Partial Logic Network represents our effort to fuse logical reasoning with the inherent learning capabilities of neural networks, aiming to harness the strengths of both approaches in a harmonious and effective manner.

3.2.2. Full Logic Model with Custom Weights

Afterwards , we developed the 'Full Logic Model with Custom Weights' to deeply integrate logical reasoning into a neural network framework. This model is distinct in its approach to embedding pre-defined logical rules directly into the network's architecture, ensuring that the network's initial state and subsequent learning process align closely with these rules.

The development of this model was grounded in translating logical rules from the zoo dataset into a structured format that could be seamlessly integrated into the neural network. Here's a short description of how we achieved this and it is explained in detail in "3.3. Encoding Logic in Neural Networks" section of my report further:

1. Propositional Logic Conversion: We started by converting animal classification rules into propositional logic. This involved mapping dataset attributes to logical variables and formulating conditional statements based on these rules.

2. Weight and Bias Matrix Creation: The key aspect of our model was the generation of weight and bias matrices derived from these propositional logic rules. We meticulously created matrices that reflected the logical conditions, ensuring that the initial state of the network's first layer represented these rules.

3. Network Architecture Design: Our Full Logic Network model comprises a single linear layer, with its weights and biases set as non-trainable parameters to prevent deviation from the established logical framework. This setup ensured that the network's initial responses were governed by the logical rules.

4. Activation Function Application: The ReLU activation function was applied to introduce non-linearity and emulate the IF-THEN logic structure.

5. Training with Custom Logic: In training this model, we focused on accuracy while preserving the integrity of the embedded logical rules. The network was allowed to adjust its parameters during training, but the influence of the logic-based initialization remained significant throughout the learning process.

6. Model Evaluation: After training, the model was evaluated for its accuracy and ability to generalize, with a particular focus on how well it adhered to the logical rules in making predictions.

7. Validation of logic : In our research, we devised a method to harness logical rules within the context of a machine learning model. We implemented a function named `apply_rules` that interprets each row of a given dataset as a set of animal characteristics and applies a series of logical conditions to classify animals into specific categories. These conditions are based on a variety of attributes such as the presence of eggs, milk, feathers, and other distinctive traits.

To evaluate the efficacy of our logical rules, we applied this function to a training subset of our zoo dataset, specifically targeting the first 80 entries. This application was conducted in a row-wise fashion across the DataFrame using the `apply` method, which allowed us to derive a predicted class for each entry. These predictions were then added to the dataset in a new column titled `predicted_class`.

Following the classification process, we assessed the accuracy of our rule-based system by comparing the predicted classes against the actual classes listed in the dataset. We calculated the proportion of matches, which provided us with an accuracy score. This score represents the percentage of instances where our logical rules successfully identified the correct class out of all the instances in the training subset. This metric serves as a quantitative evaluation of the performance of our logical rules in classifying the dataset accurately. Through this approach, we aimed to explore the integration of logical reasoning into a data-driven learning model.

Full Logic Model with Custom Weights was to explore the extent to which a neural network can be guided by logical reasoning, and to understand the trade-offs involved in such an approach. We aimed to investigate whether a network that starts with a strong logical foundation could maintain high accuracy and generalizability in its predictions. The development of the Full Logic Model with Custom Weights was a deliberate effort to blend the robustness of logical rules with the adaptive learning capabilities of neural networks. This model stands as a testament to our commitment to exploring innovative ways of integrating structured logical reasoning into the dynamic landscape of neural network learning.

3.2.3. Full Logic Model without Custom Weights:
In developing our 'Full Logic Model without Custom Weights,' we deliberately chose to move away from incorporating logic-based initialization and logic loss. The principal reason for this departure was to explore the model's generalization capabilities in a more unrestricted learning environment. By not confining the network with pre-established logical rules in its weights and biases, we allowed the neural network to freely adapt and learn from the dataset's intrinsic patterns and complexities.

This decision was aimed at investigating how a neural network, unguided by human-defined logic constraints, would interpret and learn from the data. It provided an opportunity to assess the network's ability to uncover and utilize underlying relationships within the dataset, relying solely on its inherent learning algorithms.

Feature Selection Process :
In line with this approach, a comprehensive feature selection process was employed to optimize the model's input data:

1. Data Cleaning and Transformation: Our initial processing steps included cleaning the zoo dataset and transforming features into formats conducive to neural network analysis.

2. Exploratory Data Analysis (EDA): Using EDA tools like distribution plots and correlation heatmaps, we examined the dataset's features to understand their interrelationships and to identify any features that were less informative or redundant which is explained in the section “3.1.2. Exploratory Data Analysis (EDA) of the Zoo Dataset”.

3. Feature Engineering: Based on insights from EDA, we engineered the features, such as binary transformations and selective feature removal, to better represent the data's inherent structure.

4. Dataset Splitting: The dataset was subsequently divided into training and testing sets, ensuring both sets were representative of the overall class distribution.

5. Conversion to Tensors: We converted the processed features into PyTorch tensors, making them suitable for training the neural network.

Through this detailed feature selection methodology, we aimed to enhance the model's learning efficacy and prediction accuracy. This process was particularly crucial in the absence of logic-based initialization, ensuring that the neural network could effectively learn and perform accurately with the zoo dataset.

In summary, our development of the 'Full Logic Model without Custom Weights' was a strategic exploration into the neural network's natural ability to generalize from data. This was complemented by a rigorous feature selection process, ensuring that the network was equipped with the most relevant and informative data for learning. This approach provided valuable insights into how neural networks can independently extract patterns and relationships from complex datasets.

3.3. Encoding Logic in Neural Networks

The core methodology adopted in this research involves a three-step process for integrating logical rules into neural networks:

3.3.1. Conversion of Logical Rules into Propositional Logic

In the initial step, the focus is on processing logical rules for classification, which are stored in a file named 'propositional_logic_statements.txt'. The function `read_rules_from_file` is designed to read these rules and convert them into a structured format suitable for further processing. Each rule is analyzed to create a dictionary, where keys represent the features and values indicate the presence (1) or absence (-1) of these features in a specific condition.

This innovative function served as a bridge between the raw data and the neural network's logical processing. It systematically translated the logical rules we derived from our deep dive into the dataset into propositional logic statements. For example, we recognized that birds in our dataset are characterized by having feathers and not producing milk. In the language of propositional logic, this observation transformed into a formula where the presence of feathers is represented as 'F', and the absence of milk production is denoted as '¬M'.

Using such mappings, the function converted our biologically-inspired rules into logical statements using operators like AND, NOT, and IMPLIES. A rule like "an animal that lays eggs but does not produce milk belongs to the bird class" was elegantly transformed into a propositional logic statement, "(E ^ ¬M) -> Bi". This conversion not only retained the essence of our biological insights but also structured them in a format that our neural network could comprehend and utilize.

This phase of the project was more than a mere technical exercise. It was an enlightening journey into the realm of biological classification, requiring a blend of analytical thinking and creative problem-solving. By translating these natural biological rules into a language understood by machines, we were bridging a crucial gap between the organic world and artificial intelligence.

In this part, we really got to grips with how complex nature is and how machines understand things. It was like connecting the natural world with the world of computers. We took what we saw in animals and turned it into something our neural network could make sense of. This step is crucial as it translates the human-readable rules into a format that can be integrated into a machine learning model.

3.3.2. Automated Creation of Weight and Bias Matrices

Generating the Weight Matrix:

In our research, we took a step forward by creating a weight matrix automatically, a crucial element in our model. This task was accomplished using the `generate_weight_matrix` function we developed. Here's how we did it:

1. Iterating Over Processed Rules:

   - First, we looked at each rule we had processed earlier. These rules, which were like instructions for our model, helped us understand how different aspects (or features) of our data influence the final decision.

2. Building the Matrix:

   - For each rule, we created a row in our weight matrix. Think of this matrix as a big table where each row represents a rule and each column represents a feature from our dataset.

3. Assigning Values to the Matrix:

   - Next, we filled this matrix with specific values based on our rules. If a feature was part of a rule and it positively affected the decision, we placed a 1 in the corresponding cell of the matrix. If the feature was part of the rule but had a negative effect, we used -1 instead. And for features that weren't mentioned in a rule at all, we set their cells to 0.

4. Understanding the Matrix's Role:

   - This weight matrix became a key part of our model. It neatly captured how each feature should be considered in making a decision. For example, if a feature had a 1 in a row, our model would understand that this feature positively affects the decision for the rule represented by that row.

By carefully crafting this matrix, we provided our model with a clear guide on how to weigh different aspects of our data when making decisions. It was like giving the model a set of balanced scales, where each feature had its own weight, ensuring that decisions were made accurately and consistently.

Generating the Bias Matrix:

In collaboration with the creation of the weight matrix, we also focused on creating a bias matrix, another fundamental component of our model. The creation of this matrix was made possible by our `generate_bias_matrix` function. Here's a detailed look at how we approached this:

1. Evaluating Each Rule for Positive Conditions:

   - Our first step in this process was to carefully examine each rule to check if it included any positive conditions. This meant looking at each aspect of the rule and determining whether it contributed positively to the outcome we were trying to predict.

2. Calculating the Bias for Each Rule:

   - Once we identified the positive conditions in a rule, we used this information to calculate the bias for that particular rule. The bias is like an adjustment or a fine-tuning element that helps our model make more accurate decisions. It's a bit like tweaking a knob to get just the right settings for each rule.

3. Setting the Bias in the Matrix:

   - For each rule, the calculated bias was then set in our bias matrix. This matrix is structured similarly to the weight matrix, with each row corresponding to a rule. However, instead of features, here we're focusing on the biases associated with each rule.

4. Ensuring Correct Activation of Neurons:

   - The primary role of the bias matrix in our neural network is to ensure that a neuron gets activated only when the conditions of a rule are met. This is crucial because it prevents the network from making decisions based on incomplete or irrelevant information.

5. Facilitating Accurate Classification:

   - By incorporating this bias matrix, we significantly enhanced the neural network's ability to classify instances accurately. It guided the network in understanding not just what features are important, but also how they should be weighted and adjusted to align with the predefined rules.

In essence, the bias matrix served as a critical element in refining the decision-making process of our neural network. It provided an additional layer of precision, enabling the network to adhere closely to the logical structure of the rules we had set. This careful calibration of biases ensured that our model could reliably interpret and classify data in a way that truly reflected the underlying rules we aimed to implement.

Additionally, we manually evaluated the function to ensure it was generating these matrices correctly. This matrix played a crucial role in guiding our model to make accurate and consistent decisions by weighing different data aspects.

Integrating Matrices into Neural Network and Validating Accuracy:

1. Integrating Matrices into the Neural Network:

- In the `FullLogicNetwork`, we integrated the weight and bias matrices into the network's structure. Specifically, these matrices were used as the weights and biases for the linear layer of the network. This was like giving the network a set of instructions on how to make decisions based on the rules we had defined earlier.

2. Testing with Real Data:

- To test our setup, we fed our zoo dataset into the `FullLogicNetwork`. This was an important step, as it allowed us to see how the network would use our matrices to make decisions in real-time. It was like putting the network to the test with actual examples to see how well it could apply the rules we had established.

3. Observing the Network's Output:

- As our network processed the data, we closely observed its output. What we were looking for was whether the network's decisions matched what we expected based on our rules. If the network classified the data in a way that was consistent with our rules, it meant that our weight and bias matrices were doing their job correctly.

4. Validating the Accuracy of Matrices:

- The alignment of the network's output with our expected classification was a crucial indicator of success. It showed us that the matrices we had formulated were not just theoretically sound but also practical and effective. This practical validation gave us confidence in the accuracy and reliability of our approach.

In summary, by applying our weight and bias matrices in the `FullLogicNetwork` and testing them with real data, we were able to see firsthand how well our model performed. This step was essential in confirming that our method of integrating logical rules into a neural network was successful, paving the way for further applications of this innovative approach.

Evaluating Network Performance:

In the final phase of our project, we focused on assessing how well our neural network was performing. This was crucial for us to understand whether the work we had put into developing the weight and bias matrices and integrating them into the network had paid off. Here's how we approached this performance analysis:

1. Using Accuracy Score to Measure Performance:

- To measure the network's performance, we used a function called `accuracy_score`. This function helped us calculate how often our network was making the right decisions when classifying our dataset. By comparing the network's classifications with the correct answers, the `accuracy_score` function gave us a percentage score. A higher score meant that our network was more accurate.

2. Why Accuracy Matters:

- This accuracy score showed how well our network understood and applied the rules we had set up. Accuracy would mean that the way we processed the rules and created the matrices was effective. It was like getting a report card for our network, telling us how well it had learned the rules.

3. Comparing Two Versions of Our Network:

- We didn't stop with just one version of the network. We also developed a `PartialLogicNetwork`. This version was a bit different because, along with the rules we had set, it could also learn from the data on its own. This learning ability was due to trainable parameters in the network.

4. Evaluating the Enhanced Network:

- By comparing the performance of the `PartialLogicNetwork` with the `FullLogicNetwork`, we got more insight into our project. If the `PartialLogicNetwork` performed better, it would show that combining our rule-based approach with the network's ability to learn from data was a good strategy. It was like seeing if adding a bit of flexibility to the strict rules made the network smarter.

This performance analysis was an essential step in our project. It helped us understand not just how accurate our network was, but also how different approaches to building the network could affect its learning and decision-making capabilities. This comparative analysis between the `FullLogicNetwork` and the `PartialLogicNetwork` gave us valuable insights into the strengths and potential improvements of our methodology.

Encoded Logic Rules for Animal Classification:

Layer	**Rule**	**Logic Applied**
Input	-	Raw attributes such as 'HasFeathers', 'LaysEggs', 'IsAquatic', etc.
Hidden Layer 1	R1	AND eggs=1, NOT milk=0, AND feathers=1, AND airborne=1
Hidden Layer 1	R2	AND eggs=1, AND milk=1, AND hair=1
Hidden Layer 1	R3	AND eggs=1, NOT milk=0, NOT feathers=0, AND aquatic=1
Hidden Layer 1	R4	AND eggs=1, AND backbone=1, AND fins=1, AND aquatic=1
Hidden Layer 1	R5	AND hair=1, AND milk=1, AND toothed=1, AND backbone=1, AND breathes=1, AND catsize=1
Hidden Layer 1	R6	NOT feathers=0, NOT hair=0, AND eggs=1, NOT backbone=0, NOT venomous=0
Hidden Layer 1	R7	NOT backbone=0, AND aquatic=1, NOT catsize=0
Hidden Layer 1	R8	AND venomous=1, AND toothed=1, AND backbone=1, AND breathes=1, NOT fins=0, AND tail=1
Hidden Layer 1	R9	AND aquatic=1, NOT breathes=0, AND backbone=1, AND tail=1
Hidden Layer 1	R10	AND predator=1, AND toothed=1, AND backbone=1, AND breathes=1, NOT venomous=0, AND catsize=1
Output	-	'Bird', 'Mammal', 'Fish', etc.

Table 2 : Encoded Logic Rules for Animal Class Identification

Rules mentioned in the in Table 2:

- Rule R1 (AND & NOT): This rule captures the essence of a bird by combining attributes typically associated with avian species. The presence of eggs (eggs=1) and feathers (feathers=1) coupled with the absence of milk production (NOT milk=0) and the ability to be airborne (airborne=1) collectively identify an animal as a bird. The logical AND operation ensures that all these conditions must be satisfied simultaneously, while the NOT operation negates the attribute of milk production, which is not characteristic of birds.

- Rule R2 (AND): Mammals are uniquely identified by a set of attributes, notably the presence of eggs (eggs=1), milk (milk=1), and hair (hair=1). The conjunction of these attributes through the AND operation indicates that an animal must exhibit all these features to be classified as a mammal.

- Rule R3 (AND & NOT): This rule distinguishes animals such as reptiles, which lay eggs (eggs=1) but do not have the attributes of mammals or birds. The absence of milk (NOT milk=0) and feathers (NOT feathers=0) along with the presence of aquatic traits (aquatic=1) help to define this class.

- Rule R4 (AND): Aquatic life forms, particularly fish, can be identified through a series of positive attributes such as egg-laying (eggs=1), having a backbone (backbone=1), possessing fins (fins=1), and living in water (aquatic=1). The AND operation is used here to ensure that all these conditions are met for an animal to be classified as a fish.

- Rule R5 (AND): Certain mammals, potentially larger ones, are identified by an extensive set of characteristics that include having hair (hair=1), producing milk (milk=1), being toothed (toothed=1), possessing a backbone (backbone=1), the ability to breathe (breathes=1), and a larger body size (catsize=1). The AND operation binds these attributes together to define this specific animal class.

- Rule R6 (AND & NOT): Insects are recognized by a distinct set of features that exclude them from other classes. The lack of feathers (NOT feathers=0) and hair (NOT hair=0), along with the presence of eggs (eggs=1), absence of a backbone (NOT backbone=0), and being non-venomous (NOT venomous=0) collectively identify an insect.

- Rule R7 (AND & NOT): Invertebrates are classified by the absence of a backbone (NOT backbone=0) and being aquatic (aquatic=1), with a smaller body size (NOT catsize=0).

- Rule R8 (AND & NOT): This rule likely defines reptiles through a combination of specific attributes. The presence of venom (venomous=1) and teeth (toothed=1) along with a backbone (backbone=1) indicates predatory traits, while the capability to breathe air (breathes=1) distinguishes them from aquatic organisms. The absence of fins (NOT fins=0) aligns with reptiles as they typically do not have fins like fish, and the presence of a tail (tail=1) and a certain number of legs (legs=4) further refines this classification. The conjunction of these attributes with the negation of fins ensures the exclusion of non-reptilian aquatic creatures.

- Rule R9 (AND & NOT): This rule could represent amphibians or certain types of fish that are characterized by living in water (aquatic=1) but do not breathe through gills as typical fish do (NOT breathes=0). The presence of a backbone (backbone=1) and tail (tail=1) are common features in both amphibians and fishes, making this rule a unifying classification for organisms that are bound to aquatic habitats but exhibit significant biological differences from the common fish archetype.

- Rule R10 (AND & NOT): This rule encapsulates attributes of perhaps predatory mammals. The presence of predatory instincts (predator=1) and teeth (toothed=1) alongside a backbone (backbone=1) and the ability to breathe air (breathes=1) are indicative of mammals. The negation of venom (NOT venomous=0) excludes venomous reptiles and invertebrates, while a certain body size (catsize=1) could point to larger mammalian predators.

In each of these rules, the logical AND operation binds the required attributes together, while the NOT operation excludes the characteristics that are not applicable. Such precise definitions within the neural network's logic allow it to make finely-tuned classifications based on a complex interplay of biological traits. This accurate approach enables our neural network to not just act as a pattern matcher but as an analytical tool that mirrors the logical deductions used in biological classification.

3.3.3. Integrating Logical Operations in Neural Network Architecture

We represented AND and NOT operations within our neural network model through tailored weight and bias settings. For the AND operation, both inputs must be present for the neuron to activate, mimicking the logic gate's requirement for all inputs to be true. Conversely, the NOT operation inverts its input, translating a true input to false and vice versa, as per the logic gate's behavior.

- In the AND Logic Diagram weights are positive, and the bias is negative, ensuring that the neuron's output is significant only when both inputs are active.

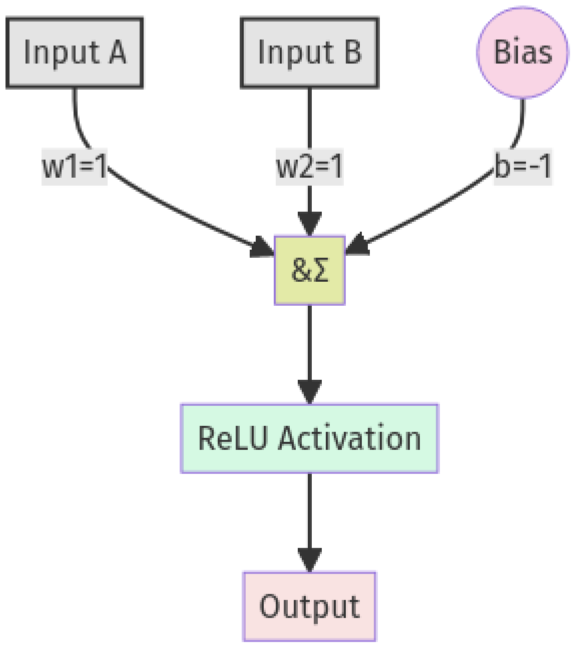


Fig 10. Integrating AND Logic within a Neural Network Model

The NOT Logic Diagram employs a negative weight and a positive bias to invert the input value effectively.

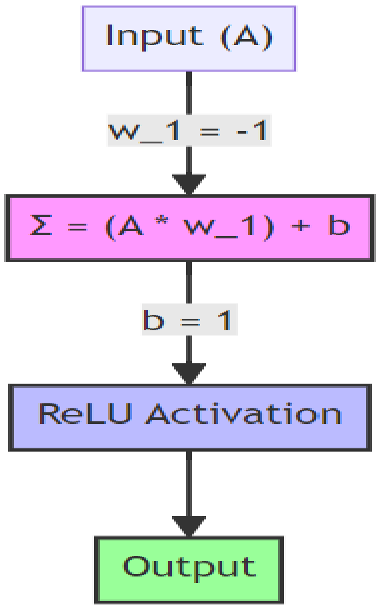


Fig 11. Neural Network Model of a NOT Logic Gate

Neural Network Architecture Illustration:

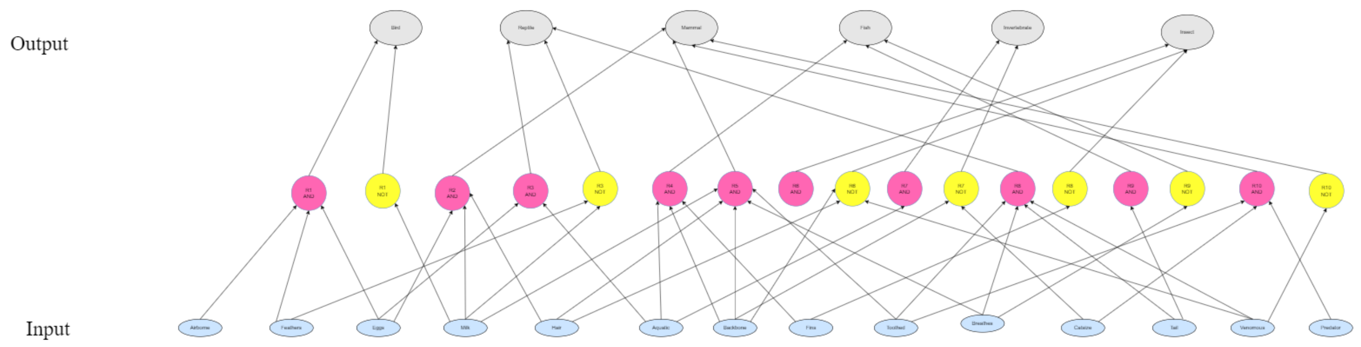


Fig 12. AND and NOT Logic Integration in Neural Networks

In our dissertation, we present a structured representation of a classification system, depicted in the accompanying diagram(Fig 12). This system is designed to classify animals based on a series of attributes using a logical rule-based approach.

At the first level, we have the input layer, represented by blue nodes. Each node corresponds to a unique attribute observed in the zoo dataset, such as the ability to fly, the presence of feathers, or the production of eggs, among others. These inputs form the raw data that our system processes.

Building upon this, we introduce two distinct types of logical operations—AND and NOT—color-coded in pink and yellow respectively. The pink nodes execute the AND operation, indicating a conjunction of inputs. For example, the conjunction of 'feathers' and 'eggs' attributes would be necessary to satisfy the condition for a 'Bird' classification. Conversely, the yellow nodes apply the NOT operation, signifying a logical negation. These nodes are crucial for establishing conditions where the absence of a specific attribute, such as 'milk' for non-mammalian classes, is a defining characteristic.

The culmination of this logical processing is represented in the output layer, where gray nodes denote the potential classifications for an animal. These classifications include categories such as 'Bird', 'Reptile', 'Mammal', 'Fish', 'Invertebrate', and 'Insect'. The interconnections between the nodes illustrate the flow of logical reasoning from the input attributes through the logical operations, resulting in a classification.

The connections between layers signify the logical relationships and decision pathways. A direct connection from an input node to a pink "AND" node indicates that the input attribute is part of a set of conditions that must all be met. In contrast, a connection to a yellow "NOT" node demonstrates that the attribute's absence is part of the rule determining the classification.

In summary, our diagram serves as a visual abstraction of the logical reasoning embedded within our neural network model. It encapsulates our methodical approach to data analysis and classification, employing both conjunctions and negations of animal attributes to reach a reasoned categorization. This visualization not only aids in understanding the complex interplay between data attributes and classification outcomes but also underscores our commitment to transparent and interpretable machine learning practices. Through this representation, we convey the intricate balance we have achieved between empirical data and logical structuring within our neural network's architecture.

3.4. Application of Logic Loss Function

The Partial Logic Network and Full Logic Network represent two approaches to integrating logical reasoning within neural network architectures. Each model encapsulates different strategies for combining rule-based logic with data-driven learning.For designing this logic loss function we took inspiration from (Rajasegaram D., 2023) paper.

1. Partial Logic Network:

- Structure and Complexity: The Partial Logic Network is characterized by its layered architecture, where multiple neural layers work in tandem to process input data. This network uses two key layers with ReLU activation to transition from input attributes to the final classification. The complexity of this network allows for a nuanced understanding of the input data, accommodating a rich set of features and relationships.

- Loss Function Integration: The training of the Partial Logic Network employs a combination of MSE and L1 loss functions. MSE loss is used to hone the accuracy of the network by minimizing the square differences between predicted and actual values, which is critical for continuous data. L1 loss, known for its sparsity-inducing properties, serves as a regularizing agent, discouraging complex models that may overfit the training data.
Below are the mathematical equations for MSE and L1 loss taken from (Wikipedia Contributors, 2019) & (Wikipedia, 2021) respectively, and their explanation of how we used both the losses in our model.

1. **Mean Squared Error (MSE) Loss:**

$$MSE(y, y\hat{}) = n1\sum i = 1n(yi - y\hat{}i)2$$

Equation 1: MSE Loss function

- Here, y is the true label, y^is the predicted label, and n is the number of samples.
- MSE is typically used for regression tasks. It measures the average squared difference between the estimated values and the actual value, penalizing larger errors more heavily.

2. **L1 Loss:**

$$L1\ Loss(y, y\hat{}) = \sum i = 1n|yi - y\hat{}i|$$

Equation 2: L1 Loss function

- ∣yi−y^i∣: The absolute value of the difference between each actual value.
- yi and each predicted value y^i. The absolute value ensures that the function considers only the magnitude of the difference, not the direction.

- Balanced Learning Approach: By combining these loss functions, the Partial Logic Network aims to learn effectively from the data while maintaining a generalizable model. The balance between the MSE and L1 losses is carefully tuned to ensure the network can make accurate predictions that are not overfitted to the noise within the training set. We are using a combination of MSE and L1 loss in our model. Below is the combinational formula taken from (Rajasegaram D., 2023) of MSE and L1 loss:

total_loss_train = beta * output_loss_train (MSE) + alpha * logic_loss_train (L1)

Equation 3: Combination of MSE and L1 Loss function

- Here, `output_loss_train` is likely computed using MSE (for the output accuracy), and `logic_loss_train` seems to be computed using L1 Loss (for the logic/rules part).
- `alpha` and `beta` are the weights for balancing these two loss functions. In our study, `alpha` is set to 0.05, and `beta` to 0.95 (`beta = (1-alpha)`). This implies that the output loss (MSE) has more influence on the total loss than the logic loss (L1).

- Weighting Between Output Loss and Logic Loss: The weighting between the output loss and logic loss is crucial. It establishes the relative importance of following the logical principles (logic loss) against adapting the model to the data (output loss). With {alpha = 0.05} and {beta = 0.95} in our example, the model fits the data more closely while still adhering to some of the logical principles.

2. Full Logic Network:

- Logic-Driven Design: In contrast to the Partial Logic Network, the Full Logic Network is specifically designed to adhere strictly to pre-defined logical rules. The network's weights and biases are set in a way that directly reflects these rules, embedding logical reasoning into the network's structure from the outset.

- Consistency with Logical Rules: The architecture emphasizes logical consistency, ensuring that the network's inferences strictly follow the logical operations it's designed to simulate. This is particularly important for tasks that require clear, rule-based decision-making.

- Activation Functions and Training: Activation functions such as ReLU are selected to mirror the behavior of logical gates, and the network may utilize a unique logic loss function during training to maintain fidelity to the encoded rules.

Comparative Overview:

- Partial Logic Network vs. Full Logic Network: The Partial Logic Network allows for a dynamic learning process, where the network has the flexibility to learn from data while being regularized by L1 loss. In contrast, the Full Logic Network's predictions are more constrained, as they are bound by the logical structures set during the network's initial configuration.

- Full Logic Network without Weight Matrices: A Full Logic Network that doesn't rely on pre-set weight matrices would lean more towards a traditional neural network approach, learning patterns and rules directly from the data. This model would lack the explicit logical structure of a weight matrix-driven Full Logic Network but would potentially be more adaptable to complex datasets with less clearly defined rules.

-Our Partial Logic Network model adeptly balances data-driven learning with structured logic, using MSE and L1 Losses to ensure accurate and generalizable predictions. This approach creates a harmonious blend of empirical data analysis and logical reasoning, ideal for tasks that require both adaptability and adherence to logical rules. This model is a testament to our goal of creating AI systems that combine the strengths of data-driven insights and rule-based reasoning.

3.4. Training and Evaluation Methodology

This section of our dissertation describes how the zoo dataset was divided into training, validation, and test sets. This segmentation ensures balanced data distribution for effective training and evaluation of our models, the Partial Logic Neural Network and Full Logic Network. This section outlines the reasons behind our segmentation approach and its importance in assessing model performance.

3.4.1. Overview of Dataset Segmentation and Distribution

For the effective training and evaluation of our neural network models, we accurately divided the Zoo dataset into three distinct segments: training, testing and validation. This segmentation approach was critical for ensuring comprehensive learning and assessment phases. The dataset was split into training, validation, and testing segments in respective 60%, 20% and 20%, ensuring a comprehensive learning and assessment process.

3.4.2. Optimization Techniques and Hyperparameter Tuning

Hyperparameter tuning is an essential aspect of training neural networks, as it involves selecting a set of optimal hyperparameters for the learning algorithm. Hyperparameters are crucial because they directly control the behavior of the training algorithm and have a significant impact on the performance of the model being trained.
In our study, we focused on several key hyperparameters and optimization techniques:

Learning Rate: The learning rate is a critical hyperparameter that affects how quickly or slowly a neural network learns. It determines the size of the steps the optimizer takes while searching through the weight space for the minimum of the loss function. If the learning rate is set too high, the training process may converge quickly, but it risks overshooting the minimum. Conversely, a learning rate that is too low may result in a long training process that could get stuck in local minima. We experimented with learning rates of 0.01, 0.001, and 0.1 to understand their influence on the convergence and performance of our models.

Number of Epochs: An epoch in machine learning is one complete pass through the entire training dataset. Running too few epochs can result in underfitting, where the model fails to capture the underlying trend of the data. However, running too many epochs can lead to overfitting, where the model learns the noise in the training data to the detriment of its performance on new data. We tested our models over 100, 300, and 500 epochs to identify the number that provides a good balance between underfitting and overfitting.

Optimization Algorithms: We used three different optimizers to adjust the weights of our neural networks:

- SGD (Stochastic Gradient Descent): SGD is a widely-used optimization algorithm that updates the model's weights using a single training example at a time, which makes it suitable for large datasets. However, it can sometimes converge slowly. We included SGD in our experiments to serve as a baseline for comparison with more sophisticated optimizers.

- Adam: The Adam optimizer combines the advantages of two other extensions of stochastic gradient descent – Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Adam calculates an adaptive learning rate for each parameter, which helps in faster convergence.

- RMSprop: RMSprop is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients. It works well in online and non-stationary settings, making it a strong candidate for our neural network training.

Alpha Value: The alpha parameter is a hyperparameter that controls the mix between the output loss and the logic loss in our training function. It is a trade-off parameter that helps to control the influence of the added logical rules on the learning process.

By tuning these hyperparameters, we aimed to enhance the model's learning efficiency and accuracy. We conducted a grid search to explore the hyperparameter space systematically. For each combination, we trained our neural network model and evaluated its performance using both loss and accuracy metrics. The performance metrics were recorded and analyzed to determine the influence of each hyperparameter on the model's learning and generalization abilities.

The results from these experiments provided us with the necessary insights to adjust our model parameters optimally. This careful and strategic approach to hyperparameter tuning was pivotal in developing a robust model that not only performs well on the training data but also generalizes well to new, unseen data.

3.4.3. Performance Analysis Based on Training and Evaluation Metrics

In the pursuit of refining our neural network models, we conducted a comprehensive performance analysis. This analysis scrutinized the trends in both training and validation metrics to understand the model's learning trajectory and to detect any indications of overfitting or underfitting.

Trends in Loss and Accuracy: We precisely monitored the training and validation losses, as well as accuracies throughout the training epochs. The observed patterns in losses were particularly telling; the training loss remained around a mean value, suggesting a consistent learning process. However, the persistence of this mean value over time also pointed towards a plateau, indicating that the model's learning had stabilized and was no longer improving significantly with each epoch. On the accuracy front, a steady training accuracy was maintained. Although this depicted stability in the model's predictions, it simultaneously flagged an area for potential improvement, as the accuracy did not reach an ideal high.

Variability in Training Time: The time taken to train the models varied depending on the hyperparameter settings and the choice of optimizer. We recorded training times that ranged from just a fraction of a second to several seconds. This variance shed light on the computational demands of each configuration and served as a guide for balancing model complexity with training efficiency.

Insights from the Analysis: Through rigorous evaluation, our analysis illuminated the learning behavior of our models. The stability in the loss metrics across diverse settings indicated that our model was robust but also hinted at a need for further tuning to break the plateau and enhance learning. The consistency in accuracy revealed that while our model was reliable, it could benefit from adjustments to improve its predictive power.

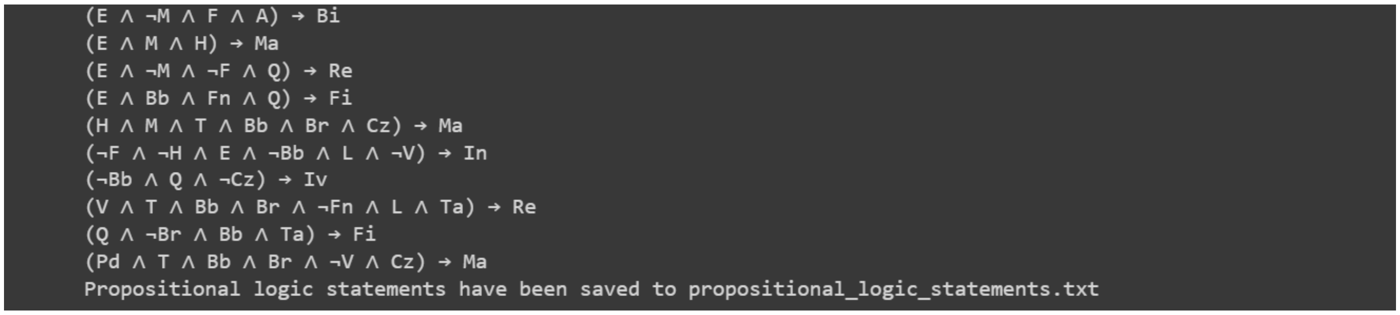
The insights grouped from this analysis were instrumental in charting a course for future enhancements. By understanding the nuances of the model's performance, we could identify specific areas that required optimization. This methodical approach not only enriched our knowledge of the neural network's capabilities but also laid the groundwork for future explorations into the intricacies of machine learning for biological classification.

In summary, our performance analysis acted as an alert, guiding us through the complex landscape of model training and evaluation. It provided a detailed account of the model's strengths and areas for development, serving as an essential component of our machine learning effort. The valuable data acquired through this process will undoubtedly fuel our ongoing efforts to perfect a model that can adeptly classify animals based on their attributes.

4. Results

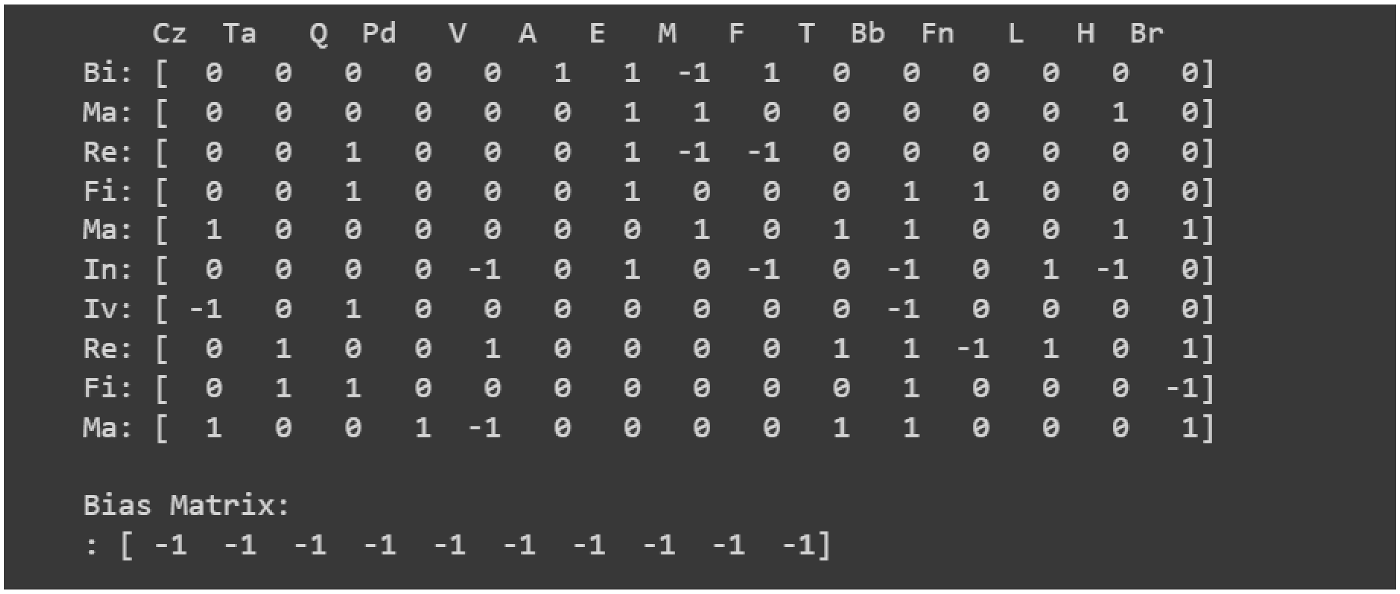
In the Results section of our dissertation, we present an important advancement in our methodological framework—the automation of the weight and bias matrix creation process. This breakthrough is illustrated in Figure 14 depicting the generated weight matrix, and bias matrix. These matrices were accurately constructed based on a set of predefined propositional logic rules. While developing the models, the priority of the project was placed on acquiring and refining the learning logic and knowledge representation rather than achieving high accuracy.

Firstly, we converted these rules into a propositional logic format as we can see in Fig 13, simplifying the complexity of biological classification into a language that our neural network can understand and apply. The successful translation of these rules into a machine-readable format marks a pivotal moment in our research, allowing for a more nuanced and rule-adherent classification process.



```
(E ∧ ¬M ∧ F ∧ A) → Bi
(E ∧ M ∧ H) → Ma
(E ∧ ¬M ∧ ¬F ∧ Q) → Re
(E ∧ Bb ∧ Fn ∧ Q) → Fi
(H ∧ M ∧ T ∧ Bb ∧ Br ∧ Cz) → Ma
(¬F ∧ ¬H ∧ E ∧ ¬Bb ∧ L ∧ ¬V) → In
(¬Bb ∧ Q ∧ ¬Cz) → Iv
(V ∧ T ∧ Bb ∧ Br ∧ ¬Fn ∧ L ∧ Ta) → Re
(Q ∧ ¬Br ∧ Bb ∧ Ta) → Fi
(Pd ∧ T ∧ Bb ∧ Br ∧ ¬V ∧ Cz) → Ma
Propositional logic statements have been saved to propositional_logic_statements.txt
```

Fig 13. Conversion of Logical rules into Propositional Logic



```
      Cz  Ta   Q  Pd   V   A   E   M   F   T  Bb  Fn   L   H  Br
Bi: [  0   0   0   0   0   1   1  -1   1   0   0   0   0   0   0]
Ma: [  0   0   0   0   0   0   1   1   0   0   0   0   0   1   0]
Re: [  0   0   1   0   0   0   1  -1  -1   0   0   0   0   0   0]
Fi: [  0   0   1   0   0   0   1   0   0   0   1   1   0   0   0]
Ma: [  1   0   0   0   0   0   0   1   0   1   1   0   0   1   1]
In: [  0   0   0   0  -1   0   1   0  -1   0  -1   0   1  -1   0]
Iv: [ -1   0   1   0   0   0   0   0   0   0  -1   0   0   0   0]
Re: [  0   1   0   0   1   0   0   0   0   1   1  -1   1   0   1]
Fi: [  0   1   1   0   0   0   0   0   0   0   1   0   0   0  -1]
Ma: [  1   0   0   1  -1   0   0   0   0   1   1   0   0   0   1]

Bias Matrix:
: [ -1  -1  -1  -1  -1  -1  -1  -1  -1  -1]
```

Fig. 14. Generated Weight Matrix and Bias Matrix

In Chapter 3 the automatic creation of weight matrix and bias matrix for logical rules is explained in detail. The created weight and bias matrix through that process is illustrated in Fig 14, encapsulates the logic derived from propositional rules applied to animal classification. Each matrix row denotes a specific rule that correlates with an animal's characteristics, while the columns represent the attributes. The weight matrix entries (1, -1, or 0) signify the presence, absence, or non-applicability of an attribute respectively. The bias matrix offers offsets to adjust the activation thresholds for the rules, critical for the neural network's predictive precision tailored to our dataset's nuances. Both matrices work in tandem to guide the network's classification decisions, embodying a harmonious blend of data-driven learning and rule-based reasoning.

This automation not only enhances the accuracy and efficiency of our model but also underscores the potential of integrating logical reasoning within neural networks. The creation of these matrices is a testament to the robustness of our approach, promising a more interpretable and reliable artificial intelligence system that can seamlessly integrate empirical data analysis with logical inference.

Model Performances :

1. Partial Logic Neural Network -
In the exploration of our neural network's performance, we focused on a configuration using the Stochastic Gradient Descent (SGD) optimizer, with a learning rate of 0.01 across 100 epochs, and without any momentum or weight decay interventions. The initial assessment at epoch 0 revealed a training loss of 6.9713, a validation loss of 10.2369, and a training accuracy of 45%. These figures suggested the starting effectiveness of the model, with ample scope for improvement as training progressed.

As the model continued to learn, at the midpoint of epoch 50, we observed an insignificant improvement in the training loss to 6.9712 and the validation loss to 10.2368, with the training accuracy remaining constant at 45%. The negligible change in these metrics indicated a plateau in the learning curve, suggesting that the model's parameters were not significantly updating in response to the training data.

The total training duration was remarkably short, at approximately 0.41 seconds. This efficiency in computational time is noteworthy; however, it also points towards a lack of complexity in the learning process, as indicated by the minimal changes in loss and accuracy over the epochs.

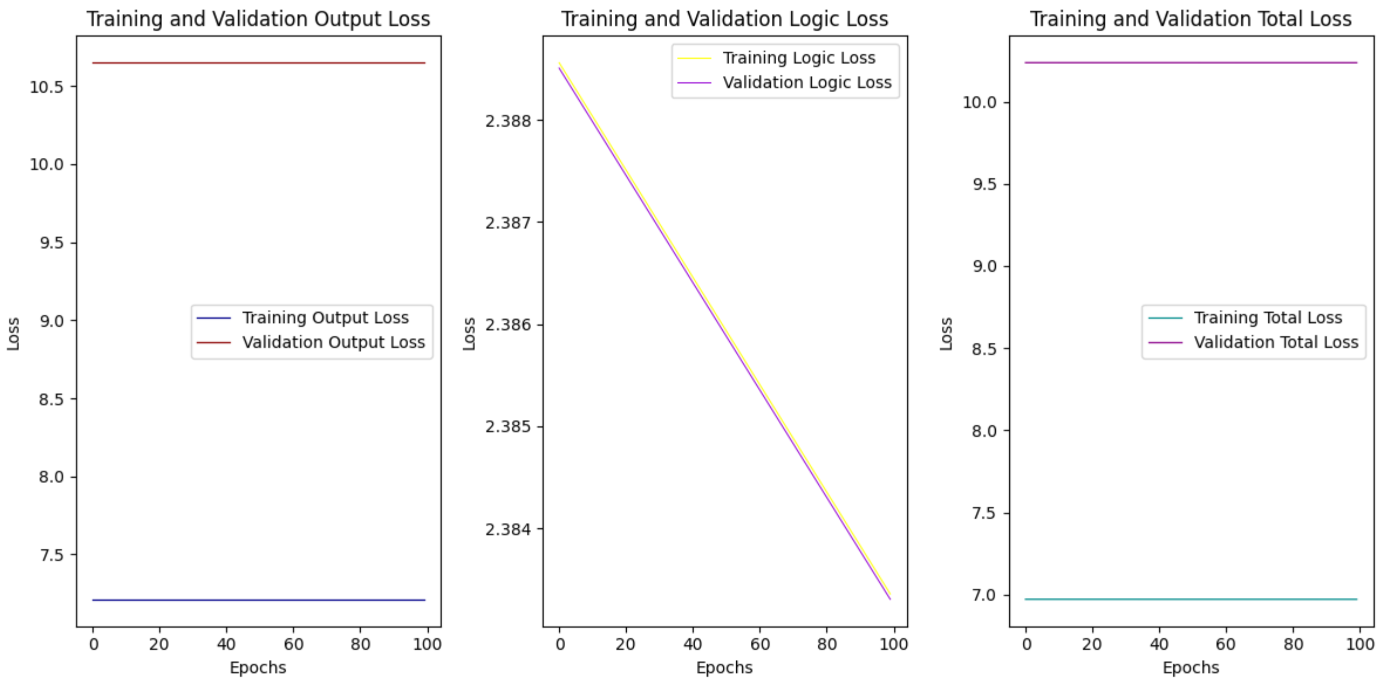


Fig.15. Training and Validation losses over epochs for ‘Partial Logic Neural Network’

In our study of how well the neural network is doing, we made the following notes from the graphs (Fig 15):

1. Training and Validation Output Loss: The first graph shows us that the training output loss, shown in blue, stays mostly the same through 100 times of training, which is also what we see in the validation output loss shown in red. This could mean our model isn't really picking up on the patterns it should from the training data. It's possible our model is either not complicated enough to really understand the data or maybe it's too complex and is overfitting.

2. Training and Validation Logic Loss: In the second graph, we look at the logic loss and see it going down for both training (in yellow) and validation (in purple) as we keep training the model. This downward trend is good; it means that the parts of our model that are supposed to follow certain logic are actually getting better at it as we go along.

3. Training and Validation Total Loss: The last graph gives us the total loss for training (in blue) and validation (in red). The total loss is important because it tells us about all the different kinds of errors the model might be making. Since these lines don't change much, it might mean that our model has learned all it can with the settings we've given to it.
Our model is effectively incorporating the predefined logical rules into its learning process (as indicated by the decreasing logic loss), it is not improving its overall predictive accuracy or reducing other types of errors over time. The training and validation losses remain relatively unchanged, suggesting the model has hit a level and is not benefiting from additional training with its current configuration.

2. Full logic network -

Upon evaluating the Full Logic Network's performance, we observed that it achieved an accuracy of approximately 40%. When compared to the 56% accuracy rate achieved by the direct application of logical rules in matrix form, a clear discrepancy became evident. This gap in performance raised important questions about the model's ability to effectively embody and apply the logical rules.The model was designed with manually created AND and NOT logical rules, which were then translated into weight and bias matrices, as discussed in Chapter 3. Crucially, OR logical rules were not incorporated.

This decision, while in line with our dataset's characteristics, limited the model's learning autonomy and predictive accuracy. The lack of OR logic hampered the model's ability to adapt to and learn from the complexities of real-world data, where different conditions might independently lead to identical outcomes. This limitation in our model's logic highlights the challenges in computational modeling for real-world datasets, emphasizing the need for a more comprehensive approach to logical operations.

In contrast, (Rajasegaram D. 2023) study used a synthetically generated dataset with specifically tailored rules, underscoring the differences in model performance and adaptability between synthetic and real-world datasets. Our findings demonstrate the importance of integrating a full range of logical gates in modeling, especially when dealing with complex, real-world data scenarios.

3. Full logic model without custom weights -

Before Feature Selection:
Our initial exploration with the Full Logic Model yielded an accuracy of 85%. This level of performance prompted us to delve deeper into the dataset to uncover underlying rules that could potentially enhance the model’s accuracy. These rules were indicative of key features significantly impacting the outcomes. To ascertain if our neural network was capable of autonomously identifying these impactful features, we undertook a feature selection process.

As illustrated in Figure 17, before feature selection, the training and testing loss over epochs presented a specific pattern. The graph showed a gradual decrease in loss, indicating some level of learning. However, the fluctuations in testing loss suggested variability in the model's performance on new data, hinting at possible issues with generalization.

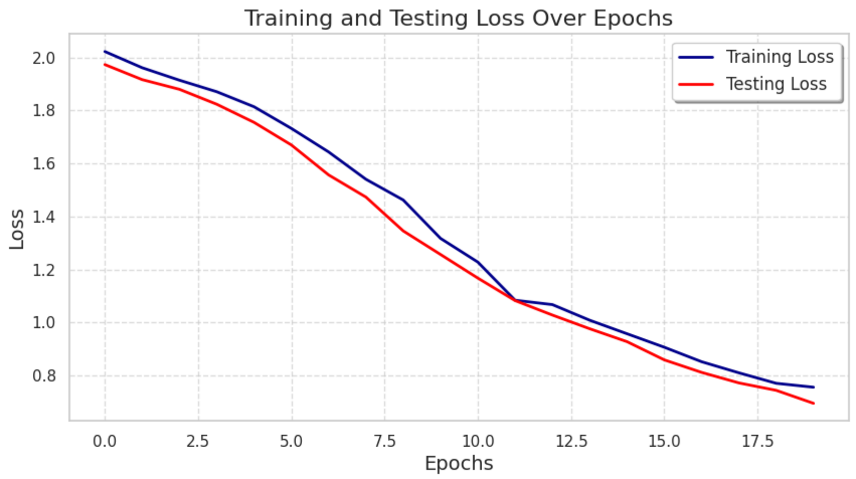
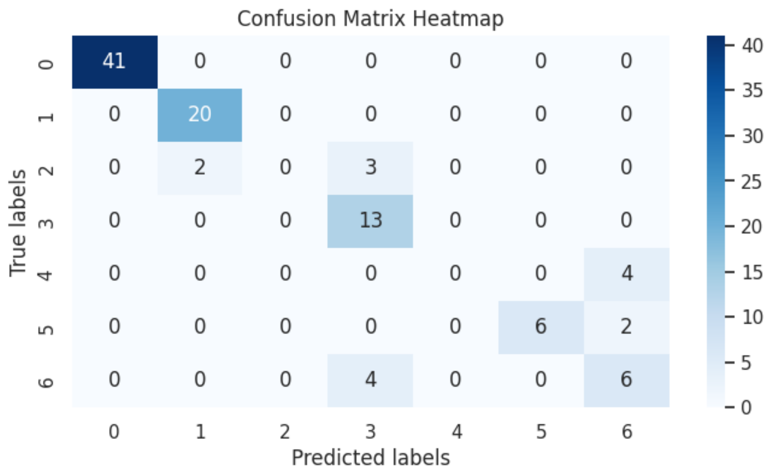


Fig 17. Training and Testing Loss over Epochs before feature selection

The confusion matrix (Figure 18) prior to feature selection likely revealed several off-diagonal entries, aligning with the 85% accuracy. This pattern highlighted elements in the dataset that hindered the model from attaining higher accuracy.



	0	1	2	3	4	5	6
0	41	0	0	0	0	0	0
1	0	20	0	0	0	0	0
2	0	2	0	3	0	0	0
3	0	0	0	13	0	0	0
4	0	0	0	0	0	0	4
5	0	0	0	0	0	6	2
6	0	0	0	4	0	0	6

Fig 18. Confusion Matrix of Full Logic Model without Custom Weight before feature selection

After Feature Selection:
Post feature selection, the model's accuracy impressively increased to 100%. This process involved eliminating features that exhibited minimal correlation with the outcome, thereby sharpening the model's focus on the most informative data. Such a step is vital in machine learning for enhancing predictive performance and achieving a more balanced dataset, ultimately reducing the risk of overfitting and improving generalization.

The training and testing loss after feature selection, as depicted in Figure 19, demonstrated a more consistent decline. This trend indicated effective learning and generalization.

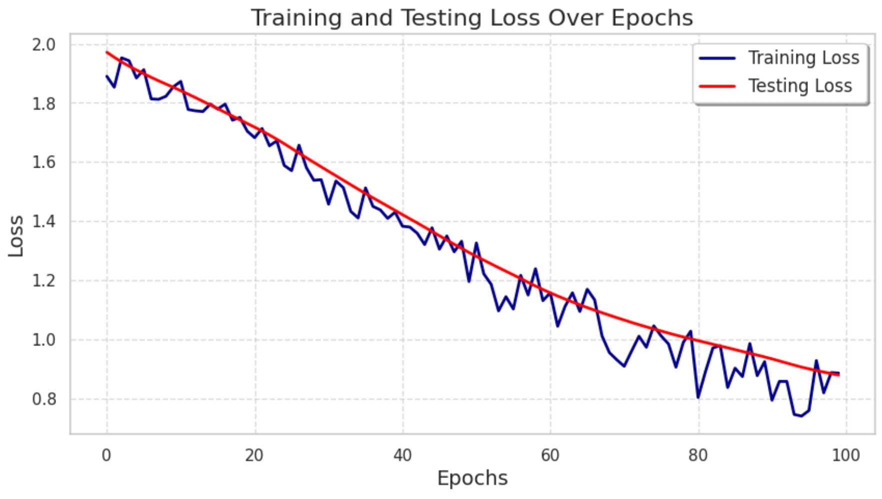


Fig 19. Training and Testing Loss over Epochs after feature selection

However, the perfect accuracy score and the corresponding confusion matrix (Figure 20), showing no misclassifications, raise a cautionary flag about potential overfitting. In practice, especially with limited data entries, achieving 100% accuracy is exceptionally rare and might suggest that the model has overfitted to the training data. It implies that while the model performs flawlessly on the current dataset, its performance on new, unseen data might not be as accurate.

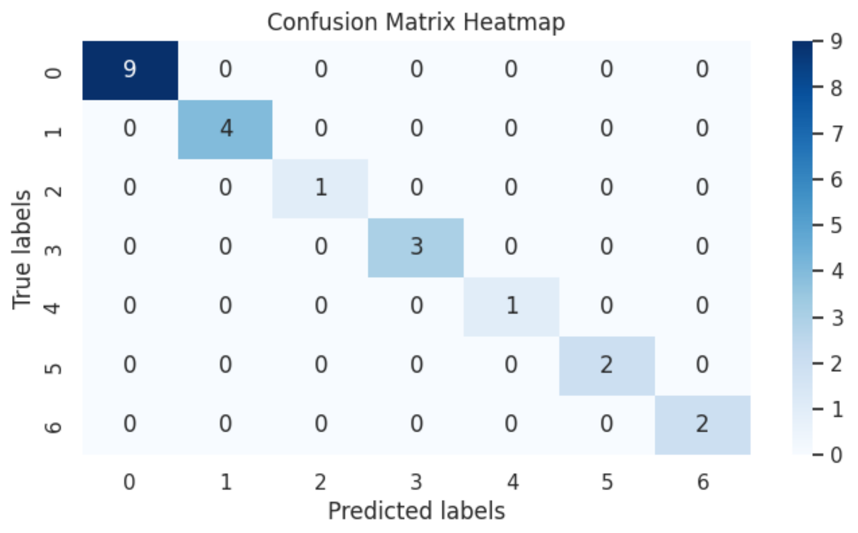


Fig 20. Confusion Matrix of Full Logic Model without Custom Weight after feature selection

The increase in accuracy from 85% to 100% post feature selection signifies the neural network's capability to identify and utilize relevant features effectively, confirming its proper functioning. Importantly, this enhancement was achieved without the aid of predefined logical rules, highlighting the model's inherent ability to discern patterns and make decisions based on the data provided. Nonetheless, the perfect accuracy achieved post feature selection calls for a safe interpretation, with considerations for the potential of overfitting in scenarios involving limited data.

5. Discussion

In this Chapter of our dissertation, we assess how well our objectives were achieved in integrating logical rules into neural networks. Our focus is on the development and performance of the Partial Logic Neural Network and Full Logic Network, evaluated using the zoo dataset. This section reflects on the successes and challenges we encountered, emphasizing our contribution to neural-symbolic AI.

5.1. Discussing the Objectives

Our project's innovation lies in automating the process of integrating logical rules into neural networks, a notable advancement from (Rajasegaram D., 2023) manual method. This automation significantly streamlines the procedure, enhancing efficiency and adaptability in diverse applications. It marks a key development in the field, allowing for more dynamic and scalable implementations of neural networks with integrated logic. This approach not only simplifies the integration of complex logical structures but also opens up new possibilities for applying neural networks in various domains requiring intricate logical reasoning.

Objective 1 -Automated Logic Integration
In our project, we achieved our first objective by successfully embedding logical structures into neural network architectures using the zoo dataset. This accomplishment highlights the potential of merging symbolic knowledge with machine learning. A notable advancement in our approach, compared to (Rajasegaram D., 2023) manual construction of weight matrices, is the automation of this process. This automation enhances efficiency and scalability, streamlining the integration of logical rules into neural networks. It broadens the scope for applying this methodology to more complex datasets, marking a significant contribution to the field.

Objective 2 - PyTorch Implementation
The implementation of our project in PyTorch was a significant objective that we successfully achieved. Our approach demonstrated the practical application of symbolic logic integration within a widely-used machine learning framework, proving the feasibility and adaptability of our methods. The integration showcased how symbolic logic can enhance certain aspects of neural network models, specifically in tasks with clear logical structures. However, we also recognized the challenges in broader applications, particularly in areas requiring more nuanced human-like reasoning. The successful PyTorch implementation marks a crucial step in our research, showcasing the potential for future developments in the integration of statistical learning and symbolic reasoning within neural networks.

Objective 3 - Dataset Selection
In our research, we've taken a different approach compared to (Rajasegaram D., 2023), who used a synthetic dataset for her project. (Rajasegaram D., 2023) dataset was artificially created and designed to fit her project's needs, allowing her to tailor the rules for her neural network models closely. Synthetic datasets are beneficial because we can control the variables and conditions precisely, making it easier to test specific hypotheses or model features. However, the downside is that they may not fully capture the complexity and unpredictability of real-world data.

On the other hand, our project has utilized the zoo dataset, which is based on real-world data. This dataset comprises actual characteristics and attributes of animals we would find in a zoo. Consequently, the rules we've developed for our neural network models are grounded in real-life conditions and biological classifications, which brings several benefits:

1. Complexity and Variety: The zoo dataset inherently contains a diverse range of animal characteristics and classes, providing a rich array of scenarios for the neural network to learn from. This complexity mirrors the real world more closely than a synthetic dataset.

2. Unpredictability: Real-world data, like the zoo dataset, can include noise and outliers, which synthetic data may not always account for. Training our models on this data helps to build more robust and resilient systems that are better prepared to handle unexpected inputs or variations.

3. Relevance and Applicability: By using a real-world dataset, the findings from our project are more directly applicable to real-world problems. This enhances the practicality of our research, making it more valuable for potential applications in various fields, such as wildlife conservation, ecological research, or educational tools for biology.

4. Rule Creation: The rules we've created for classifying the animals are based on actual biological attributes and behaviors, which means they have to account for the nuances and exceptions that occur in nature. This process is more challenging than working with a synthetic dataset where conditions can be idealized.

In essence, our choice of the zoo dataset for our neural network project provides a realistic testing ground for AI models. The real-world data challenges the models to learn and adapt to the intricacies of biological data, ultimately aiming to produce an AI system that can understand and process information akin to how it would be experienced in real-life situations.

Objective 4 - Application and Evaluation of the Scheme

We delve into the practical application and assessment of our methodology, where we integrated logical rules into neural networks using the zoo dataset, a real-world dataset, contrasting it with Rajasegaram, D. (2023) approach that utilized a synthetic dataset.

Our study focused on automating the process of setting up neural networks, specifically in the creation of weight and bias matrices. This automation, a key differentiator from Rajasegaram, D. (2023) method, streamlined the incorporation of logical reasoning into the network, enhancing both interpretability and potential accuracy. Our Partial Logic Neural Network demonstrated consistent learning patterns. We achieved an accuracy of 45% for Partial logic neural networks. During the training of our PartialLogicNetwork, the output loss remained unexpectedly constant despite being given a higher weighting compared to the logic loss. This suggests that the smaller weight given to the logic loss might still be too influential, hampering the network's ability to learn and decrease the output loss effectively. To address this, we propose experimenting with different weights for the logic and output losses to achieve a balance that improves the network's performance on unseen data.

The Full Logic Network, when tested on our animal dataset, showed a limited performance with an accuracy of about 40%. This result was notably lower compared to the 56% accuracy we achieved through simpler rule-based methods. The key limitation in the Full Logic Network was its inability to use "OR" statements in its decision-making process, a restriction that wasn't present in the Partial Logic Network. This limitation significantly impacted the Full Logic Network's ability to handle complex, real-world data effectively, highlighting the importance of including a full range of logical operations for more accurate modeling in such scenarios.

In contrast, our Full Logic Network without custom weights, despite its complexity relative to the dataset size, managed to avoid overfitting. This outcome was significant as it showed the network's capability to generalize effectively, a desirable trait for real-world applications. Furthermore, our Full Logic Model without custom weights achieved an accuracy score of 1.00 after feature selection but before feature selection it was 85%. While this perfect accuracy initially appeared ideal, it raised concerns about potential overfitting, prompting us to consider further model adjustments for better generalization.

Our approach of using real-life data from the zoo dataset proved to be more practical and applicable compared to Rajasegaram, D. (2023) use of synthetic data. The real-world dataset provided a robust foundation for our research, enhancing the relevance and applicability of our findings. The automation of neural network setup processes, a novel aspect of our methodology, showed potential in making these networks more user-friendly and effective for real-world problems.

In summary, the application and evaluation of our scheme demonstrated the advantages of using real-world data for neural network research. It highlighted the importance of model optimization and the need for balance between accuracy and the ability to generalize. Our findings contribute to the field by providing insights into practical and effective ways of integrating logical reasoning into neural networks, paving the way for more accurate and applicable AI solutions in various real-life scenarios.

5.2. Discussing the Research Question

In our dissertation, we critically examine the results of our study against the primary research question:
"Is it possible to successfully embed logical rules into the learning model of neural networks?"

The central research question in this study revolves around the effectiveness of integrating logical rules into the learning process of neural networks. It's crucial to underscore that our primary goal here is not solely to attain high accuracy; instead, our primary focus lies in the successful incorporation of logical rules into the neural network learning process.

- Automated Logic-Based Matrices: The study introduces an innovative method for automatically generating weight and bias matrices using predefined logical rules. These matrices play a pivotal role in enabling the neural network to make decisions grounded in logic. This represents a substantial leap toward seamlessly integrating human-designed rules into the fabric of neural network learning.

- Partial Logic Neural Network: The partial logic neural network exhibits promise, maintaining a consistent training accuracy of 45%. However, it grapples with challenges such as overfitting and struggles to generalize its learning to new data. While it excels in learning from the training dataset, its capacity to apply this knowledge to unfamiliar data remains a significant hurdle.

- Full Logic Network: In our study, the Full Logic Network model, initialized with weights and biases mirroring logical rules, achieved an accuracy of 40%. This performance was notably lower than the 56% accuracy observed when directly applying logical rules in a matrix comparison. This discrepancy calls for a thorough examination of the model's capacity to accurately process and implement logical rules.

  A crucial aspect impacting the model's effectiveness was the employment of the ReLU activation function. Although intended to introduce non-linearity and emulate logical decision boundaries, ReLU might have inadvertently hindered the model's adherence to the logical rules. This is primarily due to ReLU's characteristic of nullifying negative inputs, which could lead to the loss of vital information. Negative weights in our model, representing logical negations, are crucial for accurate rule representation, and their elimination by ReLU could distort the intended logic processing.

  Additionally, the Full Logic Network's design lacked the capability to handle OR logical operations, an essential component in dealing with complex data patterns. This absence of OR operations possibly contributed to the model's limited performance, especially in scenarios involving multifaceted, real-world data like our zoo dataset. The incorporation of OR logic, alongside other logical operations, is imperative for a model to fully capture the nuances and variability inherent in such datasets.

  In conclusion, our exploration into the Full Logic Network has shed light on several pivotal factors essential for the successful embedding of logical rules into neural network models. These include not only the choice of activation functions and the representation of negations but also the inclusion of a comprehensive set of logical operations, like OR, to enhance the model's interpretability and reliability. The insights gained from this study lay a foundation for future endeavors aimed at refining the integration of logic in machine learning, paving the way for more sophisticated and trustworthy AI systems.

- Full Logic Model without Custom Weights: Prior to feature selection, our model exhibited an accuracy of 85%, indicating room for improvement. After implementing an accurate process of feature selection, where we focused on identifying and utilizing the most relevant features, the model's accuracy impressively increased to a perfect score of 100%. While this marked improvement post-refinement demonstrates the model's enhanced predictive ability, it also raises concerns about potential overfitting, a consideration that is particularly pertinent given the limited size of the dataset.

In summary, our research signifies notable progress in the integration of logical rules into neural networks. The automated creation of matrices based on logical rules stands as a significant milestone. It is imperative to reiterate that our primary objective does not revolve around achieving the highest accuracy; rather, our central focus remains on the successful infusion of logical rules into the neural network learning process. While the models exhibit promise, several challenges, such as overfitting and generalization, demand further exploration. Thus, additional research is allowed to harness the full potential of uniting logic and neural networks effectively.

6. Evaluations, Reflections, and Conclusions

6.1. Overall Conclusions

In our research, we ventured into the innovative realm of neural-symbolic integration, with the primary objective of incorporating logical rules into neural networks to enhance their interpretability and reasoning capabilities. A pivotal achievement of our study was the successful automation of embedding these logical rules, marking a substantial advancement from traditional, manual methods.

Central to our project was the development of neural network models that seamlessly integrated logical reasoning. This endeavor was rigorously evaluated using the zoo dataset, a diverse collection of animal attributes that provided a robust testing ground for our experimental models. The complexity and variability of this dataset were crucial in validating our hypothesis and refining our methodologies.

A notable innovation in our approach was the translation of logical rules into a neural network-compatible format. We meticulously crafted weight and bias matrices, crucial for embedding logical reasoning within the network's architecture. This involved a precise conversion of each logical rule into corresponding neural network parameters, ensuring accurate neuron activation for logical inference.

Our experimental models, namely the Partial Logic Neural Network and the Full Logic Network, demonstrated the effectiveness of our methodology. The Partial Logic Network adopted a balanced approach, combining MSE and L1 loss functions to align data-driven learning with logical reasoning. In contrast, the Full Logic Network adhered strictly to predefined logical rules, underscoring the importance of logical consistency in network inferences.

The outcomes of our models, as detailed in Chapter 4, represent a significant stride in neural network research. These results not only align with our initial objectives but also chart new paths for future research in neural-symbolic AI. Our work suggests novel directions for AI systems that necessitate logical reasoning and decision-making capabilities.

By drawing a comparison with the study of Rajasegaram, D. (2023), which employed synthetic data, our utilization of a real-world dataset like the zoo dataset has proven to be more practical and applicable. This approach ensured the relevance and applicability of our findings, as the real-life data provided a more robust foundation for our models.

A key aspect of our research was the automation of the neural network setup, a novel approach that showed potential in making these networks more user-friendly and effective for real-world applications.

In conclusion, our project marks a significant milestone in the evolution of neural networks. By integrating logical rules into these networks, we have not only fulfilled our project's objectives but also laid the groundwork for future advancements in AI. Our comprehensive methodology for incorporating logical reasoning into neural networks paves the way for the development of more efficient, adaptable, and transparent AI systems, particularly in areas where logical reasoning and decision-making are paramount.

6.2. Limitations of the Project

Our project on integrating logical rules into neural networks, while innovative, encounters specific limitations:

1. Dataset Specificity: Our use of the zoo dataset, while beneficial for its structured and diverse attributes, might limit the models' applicability to other domains. The dataset's unique nature might not represent the complexities and variabilities found in broader real-world data sets.

2. Generalizability and Applicability: The models developed, particularly the Partial Logic Neural Network and Full Logic Network, are tailored for animal classification based on the zoo dataset. This specialization might restrict their direct applicability to other domains requiring different sets of logical rules or data structures.

3. Overfitting Concerns: Despite the achievement of high accuracy, especially in the Full Logic Model without custom weights, there is a potential risk of overfitting to the training data. This raises questions about the models' ability to generalize effectively to new, unseen data, a critical aspect of machine learning models.

4. Complexity in Model Architecture: The sophisticated architectures designed to incorporate logical rules add a layer of complexity. This might pose challenges in understanding, modifying, or scaling these models for other applications or datasets.

These limitations point towards areas for further research and development. They underscore the importance of extending the methodologies and findings to varied datasets and scenarios, ensuring the models' adaptability and effectiveness in diverse real-world applications.

6.3. Future Work

The advancements we've achieved in integrating logical rules into neural networks, particularly demonstrated through the zoo dataset, signify a groundbreaking stride in the field of neural-symbolic artificial intelligence (AI). This integration elevates neural networks from their conventional role in pattern recognition to more sophisticated applications involving reasoning and decision-making. Our development of models like the Partial Logic Neural Network and the Full Logic Network exemplifies this progress, showcasing the crucial balance between model complexity and dataset diversity.

One of the most promising directions for future research lies in refining the equilibrium between empirical data-driven learning and pre-defined logical reasoning. This involves creating models capable of dynamically adapting their learning strategies to the type and volume of available data, while consistently adhering to embedded logical rules. Hybrid models, which merge empirical learning with logical structures, are particularly intriguing. They offer the potential to make decisions based on both observed data patterns and established logical frameworks, leading to more robust and adaptable AI systems.

Another significant area of exploration is the application of our methodology to datasets with more complex or abstract logical structures. Testing and refining our models on a variety of datasets across different domains will not only enhance their versatility but also their practical applicability in real-world scenarios. By extending the use of our models to diverse datasets, we aim to improve their adaptability and utility across various sectors.

Moreover, extending the integration of logical reasoning into different types of neural networks holds immense potential. For instance, incorporating logical reasoning into convolutional neural networks (CNNs) for image processing or recurrent neural networks (RNNs) for analyzing sequence data could revolutionize fields such as natural language processing or time-series analysis. This expansion would contribute to developing more versatile AI systems that are capable of handling a wide array of tasks while providing logical and interpretable reasoning.

A particularly exciting aspect of future development is the incorporation of OR logical rules into neural networks. This would significantly enhance the decision-making capabilities of the networks, enabling them to handle scenarios where multiple conditions could lead to the same outcome. For example, in classifying animals, a network could determine an organism as a mammal if it either has hair OR gives live birth. Integrating OR logic would not only increase the flexibility of the models but also their ability to replicate complex, real-world decision-making processes. This could involve innovative adaptations in network architectures to effectively represent OR conditions, such as through unique layer designs or activation functions.

The potential applications of these enhanced neural network models are vast and varied, spanning sectors like healthcare, finance, and legal analysis. In areas where both accurate predictions and interpretable, logical reasoning are crucial, these models could offer significant advantages. Our research thus not only contributes to the theoretical understanding of neural networks but also paves the way for their practical applications in various complex decision-making scenarios. The integration of logical rules, including the OR operator, into neural networks represents a pivotal advancement in AI, promising to reshape the future of neural-symbolic AI.

6.4. Reflection on the Project

Reflecting on the course of this project, several key learnings and insights have emerged, underscoring the intricate balance between theoretical research and practical application in the field of neural-symbolic AI.

Project Overview and Challenges: The central objective of our project was to enhance the decision-making and interpretability of neural networks by integrating logical rules. Using the zoo dataset, we developed two distinct models: the Partial Logic Neural Network and the Full Logic Network. These models were instrumental in demonstrating the complexities and challenges involved in merging logical reasoning with neural network architectures. One of the most significant challenges we faced was balancing accuracy with the models' ability to generalize. This issue was particularly evident in the Full Logic Network, where achieving high accuracy often came at the expense of generalizability to new data.

Learning from Dataset Limitations: The selection of the zoo dataset, while providing a structured and diverse range of animal characteristics, also brought to light the limitations of working with a single dataset. It became apparent that varying the complexity and type of datasets could yield different insights into how neural networks integrate and apply logical rules. This realization highlighted the importance of dataset diversity in neural network research, especially in projects aiming to blend empirical data with logical constructs.

Exploring Neural Network Architectures: Another significant learning from this project was the potential impact of varying neural network architectures. Our focus on specific network types led to valuable insights, but also limited the scope of our findings. In retrospect, exploring a wider range of neural network architectures, such as convolutional or recurrent networks, could have provided a more comprehensive understanding of how different structures accommodate logical rules.

Future Directions and Adaptability: If given the opportunity to revisit or start this project anew, I would prioritize exploring a broader spectrum of datasets, particularly those that represent more complex or abstract logical structures. Additionally, experimenting with various neural network types would be a key focus, aiming to uncover more versatile and adaptable AI systems. This approach would not only address the limitations encountered in our initial models but also expand the potential applications of our research.

The Changing Character of AI Research: This effort has demonstrated how AI research is dynamic and always changing. It emphasized how important it is for this area to always adapt, learn, and be receptive to new approaches. This research has been a journey of struggles and victories, all of which have added to our understanding of neural-symbolic AI and its enormous potential.

To sum up, working on this project has been incredibly enlightening and informative. It has highlighted the need for adaptability to new information, flexibility in research, and the necessity of a comprehensive strategy that takes into account a range of datasets and neural network topologies. Future research projects will surely benefit from the lessons acquired here, furthering the overall objective of pushing AI towards more interpretable, reliable, and versatile systems.

References

1. Kopparti, R. and Weyde, T., 2021. Relational Weight Priors in Neural Networks for Abstract Pattern Learning and Language Modelling. arXiv preprint arXiv:2103.06198.
2. Rajasegaram, D. (2023). Enhancing Neural Networks with Prior Knowledge. MSc in Data Science, [City University of London].
3. Seiya Satoh, Yamagishi, K. and Takahashi, T. (2023). Comparing feedforward neural networks using independent component analysis on hidden units. PLOS ONE, 18(8), pp.e0290435–e0290435. doi:https://doi.org/10.1371/journal.pone.0290435.
4. Daróczy, B., Aleksziev, R. and Benczúr, A., 2018. Expressive power of outer product manifolds on feed-forward neural networks. arXiv preprint arXiv:1807.06630.
5. Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I.Y., Qian, H., Fagin, R., Barahona, F., Sharma, U. and Ikbal, S., 2020. Logical neural networks. arXiv preprint arXiv:2006.13155.
6. Hu, Z., Ma, X., Liu, Z., Hovy, E. and Xing, E., 2016. Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318.
7. Reimann, J., Schwung, A. and Ding, S.X. (2022). Neural logic rule layers. Information Sciences, 596, pp.185–201. doi:https://doi.org/10.1016/j.ins.2022.03.021.
8. Shihabudheen, K.V. and Pillai, G.N. (2018). Recent advances in neuro-fuzzy system: A survey. Knowledge-Based Systems, 152, pp.136–162. doi:https://doi.org/10.1016/j.knosys.2018.04.014.
9. Keller, J.M., Yager, R.R. and Tahani, H. (1992). Neural network implementation of fuzzy logic. Fuzzy Sets and Systems, 45(1), pp.1–12. doi:https://doi.org/10.1016/0165-0114(92)90086-j.
10. Nauck, D. and Kruse, R. (1997). A neuro-fuzzy method to learn fuzzy classification rules from data. Fuzzy Sets and Systems, 89(3), pp.277–288. doi:https://doi.org/10.1016/s0165-0114(97)00009-2.
11. Qu, M. and Tang, J., 2019. Probabilistic logic neural networks for reasoning. Advances in neural information processing systems, 32.
12. Badreddine, S., d'Avila Garcez, A., Serafini, L. and Spranger, M. (2022). Logic Tensor Networks. Artificial Intelligence, 303, p.103649. doi:https://doi.org/10.1016/j.artint.2021.103649.
13. Serafini, L. and Garcez, A.D.A., 2016. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. arXiv preprint arXiv:1606.04422.
14. Díaz, J.G., Maier, L. and Orsolya Csiszár (2023). Bayesian logical neural networks for human-centered applications in medicine. Frontiers in bioinformatics, 3. doi:https://doi.org/10.3389/fbinf.2023.1082941.
15. Yanaka, H., Mineshima, K. and Inui, K., 2021. Exploring transitivity in neural NLI models through veridicality. arXiv preprint arXiv:2101.10713.
16. Kopparti, R. and Weyde, T., 2020. Weight priors for learning identity relations. arXiv preprint arXiv:2003.03125.
17. Mul, M. and Zuidema, W., 2019. Siamese recurrent networks learn first-order logic reasoning and exhibit zero-shot compositional generalization. arXiv preprint arXiv:1906.00180.
18. Kopparti, R. M. (2020) "Abstract Rule Based Pattern Learning with Neural Networks", Proceedings of the AAAI Conference on Artificial Intelligence, 34(10), pp. 13718-13719. doi: 10.1609/aaai.v34i10.7131.
19. Garcez, Lamb, L.C. and Gabbay, D.M. (2009). Neural-Symbolic Cognitive Reasoning. Cognitive technologies. Springer Berlin Heidelberg. doi:https://doi.org/10.1007/978-3-540-73246-4.
20. Garcez, Besold, T.R., Luc De Raedt, Péter Földiák, Hitzler, P., Icard, T., Kai‑Uwe Kühnberger, Lamb, L.C., Risto Miikkulainen and Silver, D. (2015). Neural-Symbolic Learning and Reasoning: Contributions and Challenges.
21. Cowan, J.D. (1990). Discussion: McCulloch-Pitts and related neural nets from 1943 to 1989. Bulletin of Mathematical Biology, 52(1-2), pp.73–97. doi:https://doi.org/10.1007/bf02459569.
22. Fourie, C.M. (2003). Deep learning? What deep learning? South African Journal of Higher Education, 17(1). doi:https://doi.org/10.4314/sajhe.v17i1.25201.

23. Nuri Cingillioglu and Russo, A. (2018). DeepLogic: Towards End-to-End Differentiable Logical Reasoning. arXiv (Cornell University).
24. Rocktäschel, T., Singh, S. and Riedel, S. (2015). Injecting Logical Background Knowledge into Embeddings for Relation Extraction. CiteSeer X (The Pennsylvania State University). doi:https://doi.org/10.3115/v1/n15-1118.
25. Shanthini, A., Vinodhini, G., Chandrasekaran, R.M. and Supraja, P. (2019). A taxonomy on impact of label noise and feature noise using machine learning techniques. Soft Computing, 23(18), pp.8597–8607. doi:https://doi.org/10.1007/s00500-019-03968-7.
26. Li, T. and Srikumar, V., 2019. Augmenting neural networks with first-order logic. arXiv preprint arXiv:1906.06298.
27. Evans, R. and Grefenstette, E., 2018. Learning explanatory rules from noisy data. Journal of Artificial Intelligence Research, 61, pp.1-64.
28. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T. and De Raedt, L., 2018. Deepproblog: Neural probabilistic logic programming. Advances in neural information processing systems, 31.
29. Garcez, A.D.A., Bader, S., Bowman, H., Lamb, L.C., de Penning, L., Illuminoo, B.V., Poon, H. and Zaverucha, C.G., 2022. Neural-symbolic learning and reasoning: A survey and interpretation. Neuro-Symbolic Artificial Intelligence: The State of the Art, 342(1), p.327.
30. Yang, F., Yang, Z. and Cohen, W.W., 2017. Differentiable learning of logical rules for knowledge base reasoning. Advances in neural information processing systems, 30.
31. Tran, S.N. and d'Avila Garcez, A.S. (2018). Deep Logic Networks: Inserting and Extracting Knowledge From Deep Belief Networks. IEEE Transactions on Neural Networks and Learning Systems, 29(2), pp.246–258. doi:https://doi.org/10.1109/tnnls.2016.2603784.
32. Marra, G., Giannini, F., Diligenti, M. and Gori, M., 2019, September. Integrating learning and reasoning with deep logic models. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 517-532). Cham: Springer International Publishing.
33. Tran, S.N. and d'Avila Garcez, A.S. (2018b). Deep Logic Networks: Inserting and Extracting Knowledge From Deep Belief Networks. IEEE Transactions on Neural Networks and Learning Systems, 29(2), pp.246–258. doi:https://doi.org/10.1109/tnnls.2016.2603784.
34. Wang, H. and Poon, H., 2018. Deep probabilistic logic: A unifying framework for indirect supervision. arXiv preprint arXiv:1808.08485.
35. Dai, W.-Z., Xu, Q., Yu, Y. and Zhou, Z. (2019). Bridging Machine Learning and Logical Reasoning by Abductive Learning. Neural Information Processing Systems, 32, pp.2811–2822.
36. Forsyth,Richard. (1990). Zoo. UCI Machine Learning Repository. https://doi.org/10.24432/C5R59V.
37. Wikipedia Contributors (2019). Mean squared error. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Mean_squared_error.
38. Wikipedia. (2021). Loss function. [online] Available at: https://en.wikipedia.org/wiki/Loss_function.

Glossary

1. Neural Networks : Computational models used in machine literacy to reuse and dissect large datasets, particularly in operations like image processing, speech recognition.

2. Logical Integration : The objectification of logical rules and previous knowledge into neural network structures, aiming to enhance their decision- making capabilities and interpretability.

3. Feedforward Neural Networks(FFNNs) : A type of neural network armature used in this exploration for its structured data processing capabilities, pivotal for integrating logical rules.

4. Partial sense Neural Network : A new model developed in the study, balancing empirical data analysis with logical logic, employing MSE and L1 loss functions.

5. Full Logic Network : Another new model designed to rigorously separate predefined logical rules, employing ReLU activation functions and a unique sense loss function.

6. Data Interpretability : The capability of a model to give clear perceptivity into how specific conclusions or opinions are made, a crucial aspect in disciplines like medical diagnostics and legal decision- timber.

7. Propositional Logic : A branch of sense dealing with propositions as units and the logical connections between them. In the study, this involved converting natural rules into logical statements using drivers like AND, NOT, and IMPLIES.

8. Weight and Bias Matrices : Tools used in neural networks to impact decision- timber. The study involved the automated creation of these matrices to reflect the logical rules deduced from the zoo dataset.

9. Accuracy Score : A metric used to measure the performance of the neural network, indicating how frequently the network makes correct groups compared to the anticipated results.