

Harnessing the Advantages of Binary Networks for Neural-Symbolic Computing

Nataliia Kunanets¹, Yuriy Shcherbyna² and Volodymyr Karpiv^{2,3}

¹Lviv Polytechnic National University, Ukraine

²Ivan Franko National University of Lviv, Ukraine

³SoftServe, Ukraine

Abstract

In the dynamic field of AI, this paper explores the fusion of Neural-Symbolic Computing with binary neural networks, aiming to unify the precise logic of Symbolic AI with the adaptability of Connectionist AI. Focusing on integrating logical reasoning, this approach seeks to overcome the constraints of conventional methodologies. Our study emphasizes the significance of binary networks in achieving computational efficiency and structured logic integration. Utilizing the MNIST dataset, we demonstrate the practicality of our framework, while acknowledging the need to extend our methods to more complex systems and a broader array of datasets. This research lays the groundwork for future AI models that harmoniously combine learning and reasoning, paving the way for enhanced capabilities in various AI applications.

Keywords

Neural-Symbolic Computing, Symbolic AI, Connectionist AI, Deep Learning, Binary Neural Networks, Logical Reasoning

1. Introduction


In the evolving landscape of artificial intelligence (AI), the pursuit of effective computational models has led to diverse philosophies and methodologies. Historically, the AI community has oscillated between two dominant paradigms: Symbolic AI and Connectionist AI. This paper argues for the importance of Neural-Symbolic Computing, a field that synergizes the strengths of both approaches. We aim to establish a framework based on binary neural networks as a foundation for integrating logic and logical operators, addressing a critical gap in current AI methodologies.

In the dawn of AI research, Symbolic AI reigned supreme. This paradigm, rooted in formal logic and symbolic reasoning, was driven by the belief that intelligence could be emulated by explicitly programming rules and symbols. Pioneers like Newell and Simon, with their General Problem Solver [1], exemplified this belief. Symbolic AI excelled in domains with well-defined rules and clear objectives, such as chess. However, it struggled with real-world scenarios that required adaptive learning and handling of ambiguous data.

COLINS-2024: 8th International Conference on Computational Linguistics and Intelligent Systems, April 12–13, 2024, Lviv, Ukraine

✉ nek.lviv@gmail.com (N. Kunanets); yshcherbyna@yahoo.com (Y. Shcherbyna); volodymyr.karpiv@gmail.com (V. Karpiv)

ORCID 0000-0003-3007-2462 (N. Kunanets); 0000-0002-4942-2787 (Y. Shcherbyna); 0009-0003-8439-8043 (V. Karpiv)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The limitations of Symbolic AI led to the ascendance of Connectionist AI, marked by the development of artificial neural networks. Inspired by biological neural networks, this approach focuses on learning from data, allowing systems to adapt to new situations and recognize patterns. Seminal works like Rumelhart, Hinton, and Williams' backpropagation algorithm [2] catalyzed the deep learning revolution. However, this shift also led to skepticism towards knowledge-based systems, such as knowledge graphs, which were seen as rigid and unable to cope with the complexity and variability of real-world data.

Neural-Symbolic Computing emerges as a promising paradigm that integrates the symbolic reasoning of Symbolic AI with the adaptive learning capabilities of Connectionist AI. This hybrid approach aims to leverage the interpretability and structured knowledge representation of symbolic systems alongside the pattern recognition and learning efficiency of neural networks. Notable works in this domain include the integration of logic programming with neural networks, as demonstrated by Garcez et al. [3], and the development of differentiable logic models.

The primary advantage of Neural-Symbolic Computing lies in its potential to handle complex, real-world problems that require both structured knowledge and adaptive learning. It offers interpretability, a critical aspect in fields like healthcare and finance, where understanding decision-making processes is crucial. However, challenges remain, particularly in integrating these paradigms efficiently and ensuring that the hybrid models retain the strengths of both parent domains.

This paper contributes to the Neural-Symbolic Computing field by proposing a binary neural network framework. This framework aims to serve as a robust basis for integrating logic and logical operators, addressing a gap in current methodologies. By focusing on binary neural networks, we aim to enhance computational efficiency and provide a more structured approach to logic integration in neural networks. The development of this framework is a step towards more sophisticated AI models that can seamlessly incorporate both learning and reasoning, a crucial advancement for complex problem-solving in various domains.

2. Related Works

Symbolic AI, a foundational pillar in artificial intelligence, has significantly evolved through contributions emphasizing logic, symbols, and rule-based processing. The General Problem Solver (GPS) by Newell and Simon was a pioneering development [1], showcasing AI's ability to replicate human problem-solving using symbolic representations. This was further advanced by John McCarthy's work on formal logic and knowledge representation Check [4], Check [5], leading to a deeper understanding of how machines manipulate abstract concepts. Terry Winograd's SHRDLU program [6] extended Symbolic AI into natural language processing, demonstrating machines' capability to interpret and respond to human language in structured environments. The practical application of Symbolic AI was further exemplified in Feigenbaum and Barr's DENDRAL project [7], applying it to chemistry, and Nilsson's STRIPS system for robotics [8], showcasing the versatility of Symbolic AI across various domains.

The emergence of Connectionist AI, with its focus on artificial neural networks and data-driven learning, marked a significant shift from Symbolic AI's rule-based approach. The development of the backpropagation algorithm by Rumelhart, Hinton, and Williams [2] laid the

foundation for modern deep learning, emphasizing adaptive learning's value. Yann LeCun's convolutional neural networks (CNNs) [9] revolutionized pattern recognition in image classification, demonstrating neural networks' practical capabilities in visual data processing. The introduction of Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber [10] expanded these applications to sequential data processing, like language understanding, showcasing Connectionist AI's versatility. Bengio, LeCun, and Hinton's overview of deep learning [11] and Krizhevsky, Sutskever, and Hinton's AlexNet [12] further underscored the adaptability and efficacy of neural network-based AI approaches.

Neural-Symbolic Computing, an integrative field blending neural networks' learning capabilities with Symbolic AI's structured reasoning, emerged as AI research progressed. The integration of logic programming with neural networks by d'Avila Garcez [3], laid the groundwork for combining adaptive learning with logical reasoning. The new approach was introduced by França et al. [13], and Harnad's exploration of the Symbol Grounding Problem [14]. These studies collectively illustrate Neural-Symbolic Computing's potential in addressing complex problems requiring both structured knowledge and adaptive learning.

Recent reviews in the field of neuro-symbolic computing have illuminated its evolving landscape and critical aspects. Wang and Yang [15] offer a systematic overview of neuro-symbolic computing advancements, emphasizing its role in merging symbolic reasoning with neural network learning for future AI development. Garcez and Lamb [16] discuss the integration of deep learning with logical reasoning in neuro-symbolic computing, stressing the need for AI to be safe and interpretable. Each paper collectively underscores the significance of neuro-symbolic computing in achieving trustworthy and advanced AI systems.

In the neuro-symbolic computing domain, key studies have focused on the synergy of symbolic reasoning and neural learning to enhance AI development. Smolensky [17] and [18], in two distinct papers, emphasizes neurocompositional computing, advocating for the integration of Compositionality and Continuity to facilitate advanced AI systems with human-like cognition. Hitzler [19] offers a broad survey of the neuro-symbolic field, noting the blend of machine learning and symbolic AI as a significant trend. Van Krieken [20] delves into differentiable fuzzy logic's role in neural network training, incorporating symbolic knowledge for improved learning outcomes. Hoernle [21] presents MultiplexNet, a method that integrates logical formulas to refine neural network training and decision-making. Silver [22] explores the application of neuro-symbolic approaches in robotics, particularly in task and motion planning. Giunchiglia [23] investigates the use of logical constraints to enhance deep learning models, focusing on performance and safety. Collectively, these works highlight the importance of merging symbolic and neural methods to create AI systems that are more effective, interpretable, and closely aligned with human cognitive processes.

Exploring the forefront of neuro-symbolic computing, recent studies have innovated in architecture, logic, and reasoning frameworks to enhance AI's capabilities. Karpas [24] presents the MRKL system, a neuro-symbolic architecture combining large language models with discrete reasoning, addressing the limitations of conventional language models. Stehr [25] proposes a Probabilistic Approximate Logic for neuro-symbolic learning, facilitating the integration of domain knowledge and neural computation. Pryor [26] introduces NeuPSL, an energy-based neuro-symbolic framework that significantly improves performance in low-data settings by integrating neural and symbolic learning. Aditya [27] discusses PyReason, a software

for open-world temporal logic, enhancing reasoning over graphical structures and providing explainable inference. Hersche [28] proposes a neuro-vector-symbolic architecture (NVSA) that addresses the binding problem and rule-search inefficiencies, demonstrating high accuracy in cognitive tasks. Lastly, Li [29] introduces a softened symbol grounding approach for neuro-symbolic systems, improving the interaction between neural training and symbolic reasoning. These contributions collectively advance the neuro-symbolic field, pushing AI towards greater efficiency, interpretability, and integrated reasoning capabilities.

In the realm of visual data analysis, neuro-symbolic computing is revolutionizing the way AI interprets and interacts with imagery. Yu, Yang [30] develop a bi-level probabilistic graphical reasoning framework, BPGR, enhancing Visual Relationship Detection (VRD) by integrating symbolic knowledge with deep learning, improving performance and interpretability. Gupta and Kembhavi [31] introduce VISPROG, a neuro-symbolic system for compositional visual tasks using natural language instructions, bypassing task-specific training by generating modular programs for interpretable solutions. Surís [32] presents ViperGPT, a framework combining vision-and-language models into executable subroutines for visual query answering, improving interpretability and task generalization without further training. Li [33] proposes LOGICSEG, a visual semantic parser that merges neural learning and logic reasoning, structuring semantic concepts hierarchically for improved segmentation and cognition-mimetic reasoning. These innovative approaches demonstrate neuro-symbolic computing's potential to advance AI's capabilities in visual data processing, offering more efficient, interpretable, and adaptable solutions.

In the sphere of reinforcement learning applications, neuro-symbolic computing is enhancing AI's problem-solving capabilities. Jin [34] introduces a deep reinforcement learning framework with symbolic options, addressing challenges of data efficiency, interpretability, and transferability. Their framework, validated in-game and real-world scenarios, shows improved performance by integrating symbolic knowledge to guide policy enhancement through planning and learning from interactive trajectories. Tian [35] proposes a weakly supervised neural symbolic learning model, WS-NeSyL, for cognitive tasks, leveraging logical reasoning. This model enhances learning efficiency and accuracy by using a back search algorithm to generate pseudo labels for supervision and incorporating probabilistic logic regularization. These approaches demonstrate how embedding symbolic reasoning into reinforcement learning can significantly improve AI's ability to learn and adapt across different domains and tasks.

In the quest for efficient AI systems, the field of Binary Networks has emerged as a promising avenue for integrating logic into AI. Courbariaux et al.'s BinaryConnect [36] introduced the concept of training neural networks with binary weights, significantly reducing computational complexity and memory requirements. Rastegari et al.'s study on Binary-Weight-Networks [37] applied binary weights to large-scale image processing tasks, demonstrating these networks' practicality in complex applications. Hubara et al.'s comprehensive study on Binarized Neural Networks [38] extended the binary concept to both weights and activations, enhancing efficiency in network computation and storage. Lin et al.'s research on reducing multiplication operations in neural networks [39] highlighted the importance of computational efficiency in AI deployment, especially in resource-constrained environments. Zhou et al.'s development of DoReFa-Net [40], proposing a method for training neural networks with low bit-width weights and activations, offered insights into balancing efficiency and accuracy in neural architectures.

These advancements in Binary Networks represent a significant step towards creating more efficient AI systems, making them highly suitable for integrating logic into AI, particularly in sectors where computational resources are limited.

In the realm of binary neural networks, significant strides have been made in various applications as evidenced by several notable papers. Zhuang et al. [41], introduces an innovative approach for detecting similarity in binary code, emphasizing the importance of semantic awareness in neural networks. Martinez et al. [42], explores techniques to enhance the training of binary neural networks, leveraging real-to-binary convolutions for improved efficiency and performance. Bai et al. [43], represents a breakthrough in natural language processing, pushing the boundaries of BERT model quantization to achieve efficient, yet powerful, binary representations. Lastly, Lin et al. [44], presents a specialized binary neural network design tailored for efficient keyword spotting, showcasing the adaptability and potential of binary networks in audio processing tasks. Together, these works by Zhuang, Martinez, Bai, and Lin highlight the versatility and advancing capabilities of binary neural networks in diverse domains of AI research.

This literature review encapsulates the evolution from Symbolic AI's structured problem-solving to Connectionist AI's data-driven learning models, the unifying efforts in Neural-Symbolic Computing, and the efficiency-driven innovations in Binary Networks. Each field, with its unique contributions, demonstrates the multifaceted nature of AI research and its continuous progression towards more sophisticated, efficient, and integrated AI systems.

3. Methods

3.1. Challenges of Integrating Logic in Learning-Based Algorithms

The integration of logical reasoning into learning-based algorithms faces the fundamental challenge of reconciling two inherently different paradigms: the symbolic, rule-based approach and the connectionist, data-driven approach. Symbolic models excel in structured problem-solving and explicit reasoning, whereas connectionist models thrive on pattern recognition and implicit learning. Merging these models requires a robust framework that can seamlessly accommodate the discrete, structured nature of logical rules within the fluid, statistical nature of neural networks.

Another significant challenge is preserving the interpretability of logic-based systems when integrated with learning-based algorithms. Neural networks, especially deep learning models, are often seen as "black boxes" due to their complex and opaque decision-making processes. Integrating logic into these models demands a methodology that enhances their transparency, ensuring that the decision-making process remains understandable and justifiable, which is vital for applications in critical domains like healthcare and law.

Efficiency and scalability pose a third challenge. Traditional logic-based systems are computationally intensive and do not scale well with the increasing size and complexity of data, unlike neural networks. Integrating logic into learning-based algorithms requires an approach that can handle large-scale data without compromising on computational efficiency and speed, ensuring that the integrated system is both practical and effective for real-world applications.

3.2. Methods to Integrate Logic in Learning-Based Algorithms

A key method for integrating logic into learning-based algorithms involves the creation of hybrid models that blend the strengths of both symbolic and connectionist approaches. In these models, neural networks are typically utilized for their ability in pattern recognition and data-driven inference, while symbolic systems are employed for rule-based reasoning and decision-making. The central aim is to design architectures where these two distinct paradigms can work in harmony, thus leveraging the adaptability and learning prowess of neural networks alongside the structured and clear logical reasoning of symbolic systems.

Addressing the challenge of interpretability in neural networks requires the development of transparent mechanisms that can effectively map and elucidate their decision-making processes. This entails devising methods that allow for the visualization and explanation of the neural network's inferences in a manner that is coherent with logical reasoning. Employing techniques such as attention mechanisms can shed light on the specific aspects of data that neural networks focus on during decision-making. Moreover, the use of explainable AI (XAI) methods can help in making the decisions of neural networks more transparent, ensuring they are in alignment with the principles of logical reasoning.

To enhance computational efficiency in the integration of logical reasoning with learning-based algorithms, a dual approach of algorithm optimization and hardware acceleration can be pursued. Developing efficient training algorithms that are less demanding in terms of computational power and memory is essential. Concurrently, utilizing specialized hardware designed for neural network processing, like Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs), can significantly boost the efficiency and scalability of these integrated systems. This optimization of both software algorithms and hardware resources is crucial for a seamless and effective integration of logical reasoning into learning-based algorithms.

3.3. Potential of Binary Networks in Integrating Logic

Binary Networks present a simplified computational model compared to traditional neural networks, which significantly benefits the integration of logical reasoning. Their ability to represent weights and activations in binary form greatly reduces computational complexity. This reduction in complexity is particularly harmonious with the structured nature of logical operations, potentially easing and streamlining the process of integrating logic into these networks. The simplicity of binary representation in Binary Networks aligns well with the discrete nature of logical reasoning, suggesting a more natural and effective pathway for merging these two paradigms.

Moreover, the binary architecture of these networks drastically cuts down memory requirements and computational overhead, a crucial advantage when melding them with logic-based systems. Logic systems, with their symbolic nature, tend to be memory-intensive. The efficiency of Binary Networks, therefore, makes them ideally suited for scenarios where computational resources are limited, yet there is a need for robust logical reasoning capabilities. This efficiency not only reduces the strain on resources but also enhances the feasibility of deploying complex AI systems in various real-world applications.

Inherent in their design, Binary Networks operate with discrete values, closely mirroring the

binary nature of logical operations. This intrinsic compatibility suggests that Binary Networks could serve as an efficient medium for embedding logical reasoning within a neural framework. Such alignment facilitates a more natural integration of logical reasoning in learning-based algorithms, potentially leading to AI systems that are both computationally efficient and logically coherent.

The architectural efficiency of Binary Networks extends to the processing of logical rules and operations. When these rules are represented in a binary format, they can be processed more rapidly and seamlessly within the network's binary architecture. This synergy enhances the system's overall efficiency and effectiveness, making Binary Networks a promising candidate for developing AI systems that seamlessly blend learning with logical reasoning.

A particularly noteworthy advantage of Binary Networks is their potential to eliminate the need for traditional backpropagation, which is a computationally intensive aspect of training conventional neural networks. The simplified learning process inherent in Binary Networks allows for the exploration of alternative learning mechanisms that could be more in tune with the processes of logical reasoning. This capability of integrating logic without relying on backpropagation represents a significant leap forward in creating more efficient and streamlined AI systems.

Lastly, Binary Networks take a step towards the realm of neuromorphic computing, where the goal is to develop computing architectures that mimic the neural structures of the human brain. This advancement holds considerable promise for the integration of logical reasoning, as neuromorphic designs could potentially provide a more intuitive framework for combining logical reasoning with learning-based approaches. By aligning AI systems more closely with human-like reasoning processes, Binary Networks could play a pivotal role in the evolution of artificial intelligence.

3.4. Dataset Preparation and Preprocessing

The MNIST dataset is a large database of handwritten digits, widely used for training and testing in the field of machine learning. This dataset contains 70,000 images, split into a training set of 60,000 examples and a test set of 10,000 examples. Each image in the MNIST dataset is a 28x28 pixel grayscale representation of a digit (from 0 to 9). The simplicity and size of the MNIST dataset make it ideal for experiments in machine learning and neural network architectures.

In our implementation, the MNIST dataset is loaded using the HDF5 file format, a versatile data model that can efficiently handle large, complex data. The dataset is then converted into a floating-point format, which is more suitable for processing with neural networks. Specifically, the pixel values are normalized to aid in the convergence of the training process. The target variable, which is the actual digit each image represents, is converted into a one-hot encoded format. One-hot encoding transforms the categorical data into a binary matrix representation, which is essential for classification tasks in neural networks.

Once loaded, the dataset is divided into training and test sets. The training set consists of 60,000 images, while the test set comprises 10,000 images. This separation is crucial for evaluating the performance of the neural network model; the training set is used to train the model, and the test set is used to evaluate its performance on unseen data.

The training data undergoes shuffling to ensure that the training process does not get biased

by the order of the data. Shuffling the data helps in reducing variance and making sure that models remain general and overfit less. The data is then divided into batches. Batching is a crucial process in neural network training, particularly for large datasets like MNIST. It involves dividing the dataset into smaller, manageable batches, which are then used to train the model iteratively. This approach is not only computationally efficient but also helps in optimizing the neural network more effectively.

3.5. Binary Network Model and Initialization

Unlike traditional neural networks that operate with high-precision weights, Binary Networks simplify these elements to binary values (-1 or 1). This architecture leads to a significant reduction in memory requirements and computational overhead, making them particularly suitable for applications where efficiency is a priority. The inherent simplicity of Binary Networks also aligns well with logical operations, potentially facilitating the integration of logical reasoning within a neural framework.

In our implementation, the Binary Network is initialized with binary values for weights and biases. Weights and biases are randomly assigned either -1 or 1. This binary initialization is crucial to maintain the network's binary nature and aligns with the overall computational efficiency goal. The architecture of the network can vary depending on the specified number of layers. For instance, a network with two layers would consist of one hidden layer and one output layer. However, the model's architecture is flexible and can be adapted with more layers, depending on the complexity of the task at hand.

Each layer in the Binary Network is designed to perform specific transformations on the input data. The first layer (input layer) receives the raw input data (in the case of the MNIST dataset, this would be the pixel values of the images). Subsequent hidden layers, if present, are responsible for extracting and processing features from this input. The final layer (output layer) produces the classification result, which, for the MNIST dataset, corresponds to the identified digit.

3.6. Activation Functions

In our Binary Network, the Rectified Linear Unit (ReLU) function is the primary activation function, selected for its effectiveness in introducing non-linearity while maintaining computational simplicity. ReLU, is especially suitable for binary network architectures, facilitating complex pattern learning efficiently. However, our framework is designed with inherent flexibility, allowing for easy application of alternative activation functions depending on specific task requirements or desired network characteristics.

The network architecture supports several other activation functions, each of which is already implemented and can be seamlessly integrated into the model. These include the sigmoid function, binary step functions (binary01, binary11), and the scaled exponential linear unit (SeLU).

The sigmoid function, known for its smooth gradient, is particularly useful in scenarios where a probabilistic output is required:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

On the other hand, binary step functions, including binary01 and binary11, align closely with the binary nature of the network, making them ideal for tasks that benefit from a clear, decisive output:

$$f_{01}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (2)$$

$$f_{11}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (3)$$

The SeLU function introduces self-normalizing properties, which can be advantageous for maintaining stable gradients in deeper network architectures. In the SeLU function, λ and α are predefined constants. Typically, $\lambda \approx 1.0507$ and $\alpha \approx 1.67326$ to ensure that the mean and variance of the inputs are preserved between layers during training:

$$\text{SeLU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases} \quad (4)$$

This flexibility in the choice of activation functions allows our Binary Network to be adaptable to a wide range of applications. Whether the task requires smooth probability distributions, clear binary outputs, or stable training dynamics in deep networks, the framework can easily accommodate these needs by simply switching the activation function. This feature enhances the network's versatility, making it suitable for various machine learning tasks and experimental setups.

3.7. Loss Functions

In our Binary Network framework, while cross-entropy is the default loss function, we have integrated two loss functions to provide different options. Cross-entropy, known for its effectiveness in classification tasks, measures the difference between the predicted probabilities and the actual distribution of labels. It is particularly valuable in guiding the optimization of neural networks, especially for multi-class tasks like digit recognition in the MNIST dataset. With M as the number of classes, the cross-entropy loss is defined:

$$L = - \sum_{i=1}^M y_{o,i} \log(p_{o,i}) \quad (5)$$

However, recognizing the need for versatility in handling different types of problems, our framework also includes the root mean square error (RMSE) as an alternative loss function. RMSE is readily available and can be easily applied for tasks where the focus is on the magnitude of errors. This loss function is especially suitable for regression tasks or scenarios where assessing the accuracy of predictions in a quantitative manner is more relevant than evaluating probabilistic differences. The RMSE loss function is defined:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(\frac{d_i - f_i}{\sigma_i} \right)^2} \quad (6)$$

The integration of both cross-entropy and RMSE in our framework offers users the flexibility to choose the most appropriate loss function based on their specific task. Whether it's a classification problem requiring a probabilistic assessment or a regression task needing a quantitative error evaluation, the framework allows for seamless switching between these loss functions. This adaptability makes the Binary Network framework a versatile tool, capable of addressing a wide range of machine learning challenges effectively.

3.8. Training Parameters and Sampling Methods

In our Binary Network framework, we diverge from the traditional backpropagation training approach, familiar in neural network training. Instead, we employ a variety of statistical sampling methods. These methods, well-established in statistical analysis, provide an alternative and potentially more efficient pathway for training neural networks, particularly apt for the unique characteristics of a binary architecture.

Sampling methods bring a different perspective to network training, allowing for exploration and optimization in a manner distinct from gradient-based approaches. This shift is particularly advantageous in the context of Binary Networks, where the discrete nature of the parameters aligns well with the sampling-based exploration of the solution space. By leveraging these statistical techniques, we aim to harness their robustness and efficiency for effective training in the specialized environment of binary neural networks.

The network undergoes training over 50 epochs, allowing for gradual and thorough learning from the MNIST dataset. The MNIST dataset comprises 60,000 training images of handwritten digits, each converted into a 784-dimensional input vector to fit the network's input layer. The network architecture can have several layers, implying various hidden layers with 64 units each and an output softmax layer for classification.

The training process involves batch processing with a size of 600 samples per batch. The batch size and the data-to-invert ratio (200) are calibrated to balance between computational efficiency and training effectiveness. The choice of the sampling method for weight updates plays a crucial role in the network's training dynamics. The current setup uses global Gibbs Sampling, which is effective for global optimization in binary networks. However, other sampling methods like Local Random Sampling, Global Random sampling, and global Metropolis–Hastings are implemented and can be potential alternatives, each offering different benefits in terms of exploration and exploitation in the weight space.

In summary, the current Binary Network implementation is designed with flexibility in mind, allowing for various activation and loss functions to suit different requirements. The choice of ReLU and cross-entropy aligns well with the network's architecture and the nature of the MNIST dataset. The training process, characterized by specific hyperparameters and a chosen sampling method, is geared towards efficient and effective learning.

3.9. Logical AND Operator Mechanism

In our methodology, we integrate the output features of three distinct binary neural networks to enhance prediction accuracy and reliability through a specialized logical operation. This process is not akin to conventional model ensembling techniques; instead, it involves a unique training and inference setup tailored for binary networks. The core of our approach lies in the application of a logical AND operator, designed to make final predictions based on the agreement between the outputs of the three networks.

The logical AND operator functions under the principle that if at least two out of the three networks concur on a prediction, this consensus dictates the final output of the operator. This method leverages the collective intelligence of the networks, ensuring that the prediction reflects a majority agreement, thereby increasing the confidence in the decision made. In scenarios where each network outputs a different prediction, indicating a complete disagreement, the methodology defaults to the prediction made by the first network. This decision rule is predicated on the premise that each network, while trained under the same overarching framework, may possess subtle variations in specialization due to differences in initialization or training nuances. By prioritizing the first network’s output, we acknowledge its potential slight edge in capturing the essential features necessary for the task at hand.

Let’s denote predictions as P_1 , P_2 , and P_3 respectively. Each prediction P_i for $i = 1, 2, 3$ can be either 0 or 1, representing the binary output class of each network. The output of the AND operator, denoted as P_{AND} , can be defined as follows:

$$P_{\text{AND}} = \begin{cases} 1 & \text{if } \sum_{i=1}^3 P_i \geq 2, \\ P_1 & \text{if } \sum_{i=1}^3 P_i < 2, \end{cases} \quad (7)$$

It is crucial to underline that this strategy diverges fundamentally from traditional model ensembling. While ensembles typically combine models post-training to leverage their individual strengths during inference, our approach intertwines the combination logic within the training phase itself. This ensures that the networks are not only trained to perform their respective tasks effectively but also to do so in a manner that is synergistic, considering the logical AND operator’s requirements during the decision-making process.

4. Experimental Results

4.1. Network Training and Sampling Method

Monte-Carlo sampling and its variants, such as Gibbs sampling and Metropolis-Hastings, have emerged as efficient methods for navigating high-dimensional spaces, which can be effectively applied to deep neural networks. Traditionally, the training of binary networks has involved either utilizing backpropagation while maintaining a full-precision version of the network or adopting Bayesian learning methodologies. However, Monte-Carlo methods present an alternative approach that complements the unique characteristics of binary networks.

In our training approach, we implemented distinct strategies for the Gibbs Sampling Net (GSNet) and Metropolis-Hastings Net (MHNet), each involving the flipping of a subset of weights. In the GSNet architecture, which stands for Gibbs Sampling Net, the process involves selectively

flipping a set of weights. The acceptance of this new configuration of weights is contingent on a specific criterion: it must lead to a reduction in the loss over a given batch of data. An interesting aspect of GSNet's training methodology is the progressive increase in batch size relative to the number of epochs. This gradual scaling allows the network to initially focus on learning from smaller data segments, gradually adapting to larger and more varied data as training progresses.

MHNet, short for Metropolis-Hastings Net, also engages in flipping a random subset of weights. However, MHNet distinguishes itself from GSNet in its acceptance criterion for new weight configurations. Unlike GSNet, MHNet may accept a new weight configuration that increases the loss, albeit with a small probability. This approach allows MHNet to explore a broader range of solutions in the weight space, potentially avoiding local minima and discovering more optimal configurations. In MHNet, the batch size remains constant over time, and the number of units flipped per batch is determined randomly, adding an element of variability and exploration to the training process.

Our networks were benchmarked against the classification task on the MNIST dataset. While BinaryConnect has maintained state-of-the-art (SOTA) results for some time, more recent binary network architectures have focused on improving precision on larger and more complex datasets like ImageNet. However, it's observed that these networks often do not perform as well on MNIST when compared to BinaryConnect.

Initially, we conducted 100 experimental runs with 2000 epochs each, varying the number of hidden layers (1, 2, 3, 5, 10), the number of units (10, 20, 64, 128, 256), and testing sigmoid against ReLU nonlinearities, as well as cross-entropy against RMSE loss functions. The most promising configurations from these initial experiments were then subjected to extended training, spanning tens of thousands of epochs, to optimize for top precision.

The training process is analyzed with respect to network depth, number of units, epochs, activation functions, and choice of loss function. It's noted that training binary networks with either Gibbs Sampling or Metropolis-Hastings requires significantly more epochs to converge. This observation is evident in our results as shown in Figure 1 and Figure 2, where various fully connected architectures trained over 2000 epochs are compared. Interestingly, adding more hidden layers resulted in lower precision compared to architectures with fewer or no hidden layers.

In our study, we observed that increasing the number of hidden units in the network initially leads to improved precision up to a certain point, after which the precision begins to decline, as illustrated in Figure 2. This trend suggests a nuanced relationship between the network's complexity and its performance. Our hypothesis is that while architectures with a greater number of neurons have a higher capacity for learning and abstraction, they also demand a more extended period of training to fully leverage this increased capacity. This extended training requirement might be necessary to optimize the more complex parameter space effectively, thus benefiting from the network's higher capability.

In our experiments, the sigmoid activation function outperformed ReLU in simpler network designs, especially in networks with more hidden layers and units where sigmoid showed less saturation. Regarding loss functions, while RMSE led to better initial precision, cross-entropy loss yielded slightly improved results after extended training.

Although binary inputs and activation functions slightly reduced performance, their potential

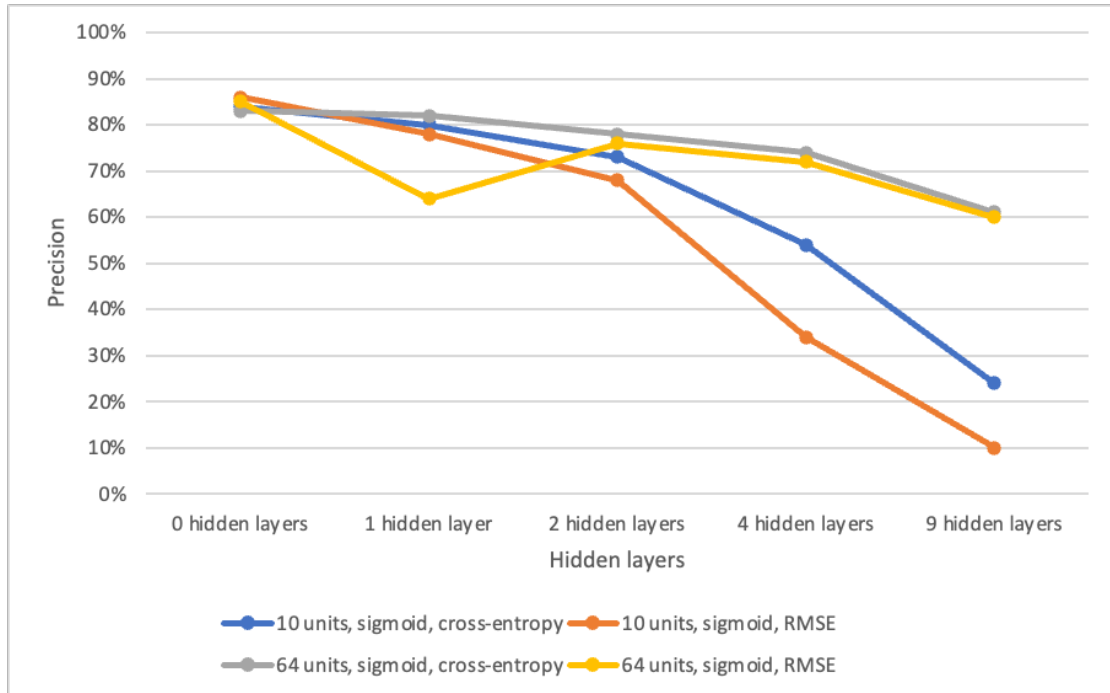


Figure 1: Varying Number of Hidden Layer Units Impact on Classification Accuracy.

for computational efficiency is notable. By replacing multiplication with bitwise operations, they could greatly accelerate network inference, making them an intriguing area for further research with various configurations and extended training periods.

4.2. Benchmarking Results

In our study, we conducted a comprehensive comparison of the precision across different binary network configurations, see Table 1. It's important to note that adding more layers and units generally improves the precision of neural networks. However, training networks with significantly more hidden layers and units poses challenges, often leading to convergence issues due to numerical limitations. Implementing techniques like batch normalization could potentially facilitate the training of deeper networks.

Given the challenges and variances in training, we evaluate precision for networks either without hidden layers or for networks with a single hidden layer comprising 64 units. These binary networks were benchmarked against a full-precision network with decimal weights, trained for 50 epochs using Stochastic Gradient Descent (SGD) with momentum (see Table 2 for details).

BinaryConnect, a notable approach in binary networks, involves using a decimal network for weight updates followed by binarization. This method, thanks to robust backpropagation, achieves high precision in just 250 epochs, showing only a 1% precision drop compared to the decimal network. The binarization process in BinaryConnect involves setting negative weights

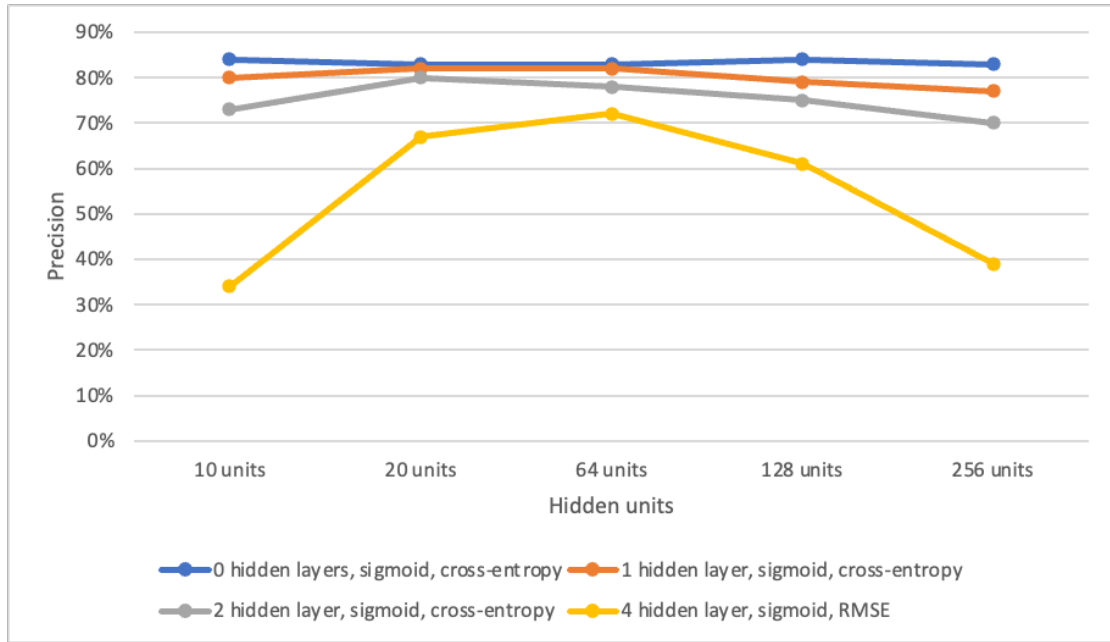


Figure 2: Varying Number of Hidden Layer Units Impact on Classification Accuracy.

to -1 and positive weights to +1.

An extension to basic weight binarization includes an additional scaling parameter per layer, ensuring the norm of weights per layer remains consistent post-binarization. This technique showed improved precision in networks with one hidden layer, while having a slight precision decrease in networks without hidden layers.

BinaryConnect slightly outperforms GSNet, as shown in Table 1, and this gap could potentially be reduced by implementing batch normalization, an important factor for binary networks. However, it is worth noting that GSNet required a significantly longer training period, 20,000 epochs, as detailed in Table 2, indicating a need for accelerated training methods.

On the other hand, MHNet demonstrated very competitive precision with substantially fewer training epochs compared to GSNet. Direct weight binarization yielded promising results, 79.67% and 73.76% as per Table 1, and the addition of a scale factor per layer further improved precision for networks with one hidden layer 76.83% as per Table 1.

4.3. Integration of Logical Reasoning

The training process for individual binary neural networks, as evidenced by the systematic reduction in training loss across epochs, indicates a successful optimization trajectory. Experiment 1 through Experiment 5, each representing a standalone binary network, shows a consistent decline in loss values, signifying that the networks are effectively learning from the data over time, see Figure 3. This pattern is characteristic of a well-tuned training regimen, where the network parameters are being refined in response to the given training stimuli, leading to improved performance on the training dataset.

Table 1
MNIST Benchmarking Results

Method	No Hidden Layers	Single Hidden Layer
Full Precision	91.6%	97.6%
BinaryConnect	90.6%	96.7%
GSNet	88.3%	94.2%
MHNet	83.7%	90.1%
Weight Binarization	79.7%	73.8%
Weight Binarization with Scaling	79.2%	76.8%

Table 2
Method Details for the MNIST Classification

Method	Activation	Last Activation	Loss	Optimization	Epochs	Other
Full Precision	sigmoid	softmax	cross-entropy	SGD with momentum	50	-
BinaryConnect	ReLU	identity	squared hinge loss	ADAM	250	Batch Norm
GSNet	sigmoid	softmax	RMSE	GS with adaptive rate	20000	-
MHNet	sigmoid	softmax	MSE	MS with temperature	5000	-
Weight Binarization	sigmoid	softmax	cross-entropy	SGD with momentum	50	-
Weight Binarization with Scaling	sigmoid	softmax	cross-entropy	SGD with momentum	50	-

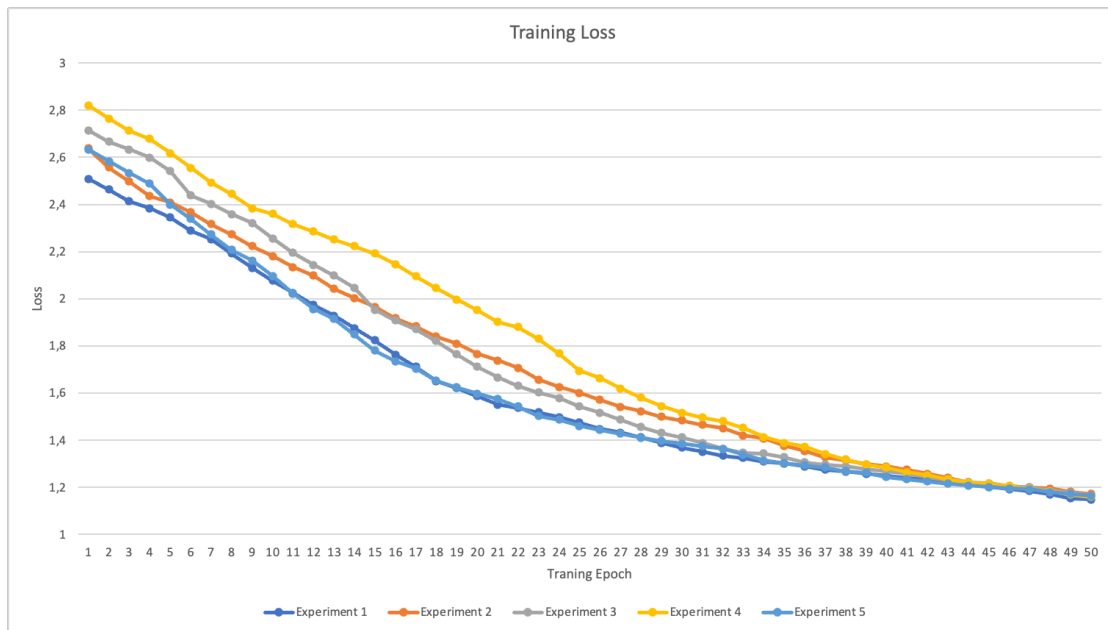


Figure 3: Training Loss for five experiments of a single binary network without AND operator.

However, an interesting divergence is observed when these binary networks are combined using the proposed AND operator. The resultant system, which integrates the output features of the three networks and adjudicates the final prediction based on a majority rule, exhibits a less favorable optimization pattern. The training loss for this ensemble does not decrease as

expected, suggesting that the joint training under the AND constraint is less effective. This could imply that the coupling of outputs in this manner introduces complexity that hinders the learning process, possibly due to conflicting gradients or a loss surface that is difficult to navigate. The intricacies of this combined operation warrant further investigation to understand the underlying causes and to explore potential modifications that could lead to more stable and efficient training dynamics.

5. Discussions

In our investigation, we have identified a current limitation within our framework: the training of the AND operator. Despite the individual binary networks showing promising results, the AND operator, which aims to combine the predictions from multiple networks, is not being trained effectively at this stage. A notable limitation in our current research is the challenge of applying our binary network framework to more complex neural architectures. Despite achieving high precision with simpler structures, particularly on the MNIST dataset, scaling up to networks with an increased number of layers and units poses significant challenges. These complex architectures demand more advanced training strategies and heightened computational requirements. Future work will be dedicated to developing methodologies that can efficiently integrate our binary network approach into these more intricate architectures. This will involve exploring new training techniques, possibly incorporating alternative activation functions, and considering innovative layer types tailored for binary networks.

Another area for improvement is the extension of our binary network framework to tasks beyond the MNIST dataset. While MNIST serves as a fundamental benchmark in machine learning, it lacks the complexity found in other datasets used for more advanced tasks like natural language processing or detailed image recognition. Our future research aims to apply the binary network framework to a diverse array of datasets and tasks. This expansion is crucial not only to test the versatility of our approach but also to refine the network's ability to process various types of data. Such an expansion could lead to new insights and improvements in how binary networks are structured and trained, potentially opening up new applications in AI.

Finally, a significant area for future exploration is the efficient integration of logical reasoning into the binary network framework. While binary networks are inherently well-suited for logical operations, embedding complex logical reasoning within these networks efficiently remains challenging. Future phases of our research will focus on discovering methods to incorporate sophisticated logical reasoning into binary networks more effectively. This could include developing novel training algorithms, experimenting with hybrid neural-symbolic models, or creating specialized layers for logic processing. The ultimate goal is to enhance binary networks not only in terms of computational efficiency but also to equip them with advanced reasoning and decision-making capabilities.

This research represents an early-stage exploration into integrating logical reasoning with binary networks. Currently, its applicability is limited, reflecting the developing nature of this innovative approach. However, with continued development and refinement, this framework has significant potential to scale across a broader range of datasets and tasks in the future. Success with this approach could pave the way for binary networks that not only perform standard

computational functions but also possess the capability for advanced reasoning. The prospect of binary networks effectively conducting logical operations opens up exciting possibilities for their application in more complex, real-world scenarios, ultimately enhancing the scope and functionality of AI systems.

6. Conclusions

In this paper, we explored the novel integration of Neural-Symbolic Computing with binary neural networks, an innovative approach that merges structured reasoning with the dynamic learning capabilities of Connectionist AI. This pioneering synthesis is designed to forge a cutting-edge framework that seamlessly blends logical operators within AI systems, thereby significantly boosting computational efficiency and adaptability. Our exploration marks a contribution to the field, highlighting the potential of binary networks to revolutionize Neural-Symbolic Computing by offering a more efficient, logical, and adaptable AI architecture.

Through our research, we demonstrated that binary neural networks could effectively embody this integration, as evidenced by our experiments with the MNIST dataset. These networks offer a promising avenue for AI applications, especially in scenarios demanding both logical processing and learning adaptability. However, our findings also underscore the challenges in scaling these networks for more complex architectures and broader datasets.

Looking ahead, our research opens several pathways for further exploration. The potential expansion of binary network applications to more sophisticated tasks, and their adaptation to handle a wider variety of data types, stand out as promising future endeavors. Additionally, the efficient integration of logical reasoning into these networks remains a pivotal area for ongoing development.

In conclusion, our work contributes to the broader field of AI by proposing a novel approach that leverages the strengths of both traditional symbolic systems and modern neural networks. The development of this binary neural network framework marks a step towards creating more advanced AI models capable of seamless learning and reasoning. It is a step towards the realization of AI systems that are not only efficient and powerful but also interpretable and adaptable, capable of tackling complex problems across various domains.

References

- [1] A. Newell, J. C. Shaw, H. A. Simon, Report on a general problem-solving program, Proceedings of the International Conference on Information Processing (1959).
- [2] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, *Nature* (1986). doi:10.1038/323533a0.
- [3] A. S. Garcez, L. C. Lamb, D. M. Gabbay, Neural-symbolic cognitive reasoning, 2008.
- [4] J. McCarthy, Programs with common sense, 1959.
- [5] J. McCarthy, Situations, actions, and causal laws, Comtex Scientific, 1963.
- [6] T. Winograd, Procedures as a representation for data in a computer program for understanding natural language (1971).

- [7] B. G. Buchanan, E. A. Feigenbaum, Dendral and meta-dendral: Their applications dimension, *Readings in artificial intelligence* (1981). doi:10.1016/B978-0-934613-03-3.50026-X.
- [8] R. E. Fikes, N. J. Nilsson, Strips: A new approach to the application of theorem proving to problem solving, *Artificial intelligence* (1971). doi:10.1016/0004-3702(71)90010-5.
- [9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, J. L. D., Back-propagation applied to handwritten zip code recognition, *Neural computation* (1989). doi:10.1162/neco.1989.1.4.541.
- [10] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural computation* (1997). doi:10.1162/neco.1997.9.8.1735.
- [11] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature* (2015). doi:10.1038/nature14539.
- [12] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems* (2012). doi:10.1145/3065386.
- [13] M. V. França, G. Zaverucha, A. S. d'Avila Garcez, Fast relational learning using bottom clause propositionalization with artificial neural networks, *Machine learning* (2014). doi:10.1007/s10994-013-5392-1.
- [14] S. Harnad, The symbol grounding problem, *Physica D: Nonlinear Phenomena* (1990). doi:10.1016/0167-2789(90)90087-6.
- [15] W. Wang, Y. Yang, Towards data-and knowledge-driven artificial intelligence: A survey on neuro-symbolic computing, *arXiv preprint* (2022). doi:10.48550/arXiv.2210.15889.
- [16] A. d'Avila Garcez, L. C. Lamb, Neurosymbolic ai: The 3 rd wave, *Artificial Intelligence Review* (2023). doi:10.48550/arXiv.2012.05876.
- [17] P. Smolensky, R. T. McCoy, R. Fernandez, M. Goldrick, J. Gao, Neurocompositional computing in human and machine intelligence: A tutorial, *Microsoft Technical Report* (2022). doi:10.48550/arXiv.2205.01128.
- [18] P. Smolensky, R. McCoy, R. Fernandez, M. Goldrick, J. Gao, Neurocompositional computing: From the central paradox of cognition to a new generation of ai systems, *AI Magazine* (2022). doi:10.1002/aaai.12065.
- [19] P. Hitzler, A. Eberhart, M. Ebrahimi, M. K. Sarker, , L. Zhou, Neuro-symbolic approaches in artificial intelligence, *National Science Review* (2022). doi:10.1093/nsr/nwac035.
- [20] E. van Krieken, E. Acar, , F. van Harmelen, Analyzing differentiable fuzzy logic operators, *Artificial Intelligence* (2022). doi:10.1016/j.artint.2021.103602.
- [21] N. Hoernle, R. M. Karampatsis, V. Belle, K. Gal, Multiplexnet: Towards fully satisfied logical constraints in neural networks, *Artificial Intelligence* (2022). doi:10.48550/arXiv.2111.01564.
- [22] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Perez, L. P. Kaelbling, Learning neuro-symbolic skills for bilevel planning, *Robot Learning* (2022). doi:10.48550/arXiv.2206.10680.
- [23] E. Giunchiglia, M. Stoian, T. Lukasiewicz, Deep learning with logical constraints, *IJ-CAI/AAAI Press* (2022). doi:10.24963/ijcai.2022/767.
- [24] E. Karpas, O. Abend, Y. Belinkov, B. Lenz, O. Lieber, N. Ratner, Y. Shoham, H. Bata, Y. Levine, K. Leyton-Brown, Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning, *arXiv preprint*

- (2022). doi:10.48550/arXiv.2205.00445.
- [25] M.-O. Stehr, M. Kim, C. Talcott, A probabilistic approximate logic for neuro-symbolic learning and reasoning, *Log Algebr Methods Program* (2022). doi:10.1016/j.jlamp.2021.100719.
 - [26] C. Pryor, C. Dickens, E. Augustine, A. Albalak, W. Wang, L. N. Getoor, Neural probabilistic soft logic, *IJCAI-23* (2022). doi:10.48550/arXiv.2205.14268.
 - [27] D. Aditya, K. Mukherji, S. Balasubramanian, A. Chaudhary, P. P. Shakarian, Software for open world temporal logic, *arXiv preprint* (2023). doi:10.48550/arXiv.2302.13482.
 - [28] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian, A. Rahimi, A neuro-vector-symbolic architecture for solving raven’s progressive matrices, *Nature Machine Intelligence* (2023). doi:10.48550/arXiv.2203.04571.
 - [29] Z. Li, Y. Yao, T. Chen, J. Xu, C. Cao, X. Ma, L. Jianetal, Softened symbol grounding for neuro-symbolic systems, *The Eleventh International Conference on Learning Representations* (2023). doi:10.48550/arXiv.2403.00323.
 - [30] D. Yu, B. Yang, Q. Wei, A. Li, S. Pan, A probabilistic graphical model based on neural-symbolic reasoning for visual relationship detection, *CVPR* (2022). doi:10.1109/CVPR52688.2022.01035.
 - [31] T. Gupta, A. Kembhavi, Visual programming: Compositional visual reasoning without training, *IEEE Conf. Comput. Vis. Pattern Recognit.* (2023). doi:10.48550/arXiv.2211.11559.
 - [32] D. S. ’is, S. Menon, C. Vondrick, Vipergpt: Visual inference via python execution for reasoning, *ICCV* (2023). doi:10.48550/arXiv.2303.08128.
 - [33] L. Li, W. Wang, Y. Yang, Logicseg: Parsing visual semantics with neural logic learning and reasoning, *ICCV* (2023). doi:10.1109/ICCV51070.2023.00381.
 - [34] M. Jin, Z. Ma, K. Jin, H. H. Zhuo, C. Chen, C. Yu, Creativity of ai: Automatic symbolic option discovery for facilitating deep reinforcement learning, *AAAI Conference on Artificial Intelligence* (2022). doi:10.1609/aaai.v36i6.20663.
 - [35] J. Tian, Y. Li, W. Chen, L. Xiao, H. He, Y. Jin, Weakly supervised neural symbolic learning for cognitive tasks, *AAAI* (2022). doi:10.1609/aaai.v36i5.20533.
 - [36] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, *Advances in neural information processing systems* (2015). doi:10.48550/arXiv.1511.00363.
 - [37] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, *European conference on computer vision* (2016). doi:10.1007/978-3-319-46493-0_32.
 - [38] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks, *Advances in neural information processing systems* (2016).
 - [39] Z. Lin, M. Courbariaux, R. Memisevic, Y. Bengio, Neural networks with few multiplications, *arXiv preprint* (2015). doi:10.48550/arXiv.1510.03009.
 - [40] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, *arXiv preprint* (2016). doi:10.48550/arXiv.1606.06160.
 - [41] Z. Yu, R. Cao, Q. Tang, S. Nie, J. Huang, S. Wu, Order matters: Semantic-aware neural networks for binary code similarity detection, *Proceedings of the AAAI conference on*

- artificial intelligence (2020). doi:10.1609/aaai.v34i01.5466.
- [42] B. Martinez, J. Yang, A. Bulat, G. Tzimiropoulos, Training binary neural networks with real-to-binary convolutions, arXiv preprint (2020). doi:10.48550/arXiv.2003.11535.
- [43] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, I. King, Binarybert: Pushing the limit of bert quantization, arXiv preprint (2021). doi:10.48550/arXiv.2012.15701.
- [44] H. Qin, X. Ma, Y. Ding, X. Li, Y. Zhang, Y. Tian, X. Liu, Bifsmn: Binary neural network for keyword spotting, arXiv preprint (2022). doi:10.48550/arXiv.2202.06483.