ISSN: 2454-9940

# INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 19, Issue 1, 2025

# Analyzing Reach ability in Binary Neural Networks with Continuous Inputs Using Star Methods

N Sreeram[1], Dr. B. Vasavi[2], C Preetham[3],*A Bheem Raj[4]

[1]AssistantProfessor, KL University, Guntur, Andhra Pradesh –522502

[3]UG Scholar, St. Martin's Engineering College, Secunderabad, Telangana – 500100

[2]Associate Professor, MVSR Engineering College, Hyderabad, Telangana-501510

[4]Assistant Professor, St. Martin's Engineering College, Secunderabad, Telangana 500100

*Corresponding Author

Email: abheemrajit@smec.ac.in

**ABSTRACT**

This study investigates the reachability of binary neural networks (BNNs) when subjected to continuous inputs, utilizing star methods for analysis. As BNNs gain prominence in various applications, understanding their behavior in the face of continuous variations is crucial for ensuring reliability and safety. The star method framework allows for the encapsulation of input uncertainties, providing a systematic approach to assess the reachability of neural network outputs. Through a combination of theoretical analysis and practical experimentation, this research elucidates the potential impacts of continuous inputs on BNN performance and robustness. The findings offer valuable insights for developers and researchers aiming to enhance the deployment of BNNs in real-world scenarios.

Keywords: Binary Neural Networks, Reachability Analysis, Continuous Inputs, Star Methods, Neural Network Robustness, Input Uncertainty, Theoretical Analysis, Performance Assessment.

## I. INTRODUCTION

DNNs have become a popular technique for complex problems in various areas such as computer vision [3], natural language processing [4], and information retrieval [5]. They are being used in various fields like robotics [6], healthcare[7], agriculture [8], construction [9], etc. In order to handle such a vast set of tasks, different deep neural networks possess different architectures [10]. Architecture of a DNN is defined by the number of parameters the network implements (neurons and synapses), layers, activation functions, etc. Depending on the task's complexity, larger architecture and datasets may be required to unlock better performance, which usually requires more training time. In addition, the architecture of a real-world DNN may grow exponentially in the number of parameters [11]. This becomes an issue when a DNN needs to be deployed on an edge device or as part of an embedded system, especially if in real-time conditions. To run applications in embedded systems, several demands must be met:(a) low power and memory consumption, (b) high accuracy, and (c) real-time performance. While modern training techniques allow (b) and (c) successfully,(a) is usually complicated by the usage of floating-point arithmetic format and a generally large scale of the models. One of the ways of overcoming the highlighted issues has been the usage of simplified DNN architectures [12]. One type of such architecture is called BNNs [13‑20]. BNNs utilize binarization, a 1-bit quantization where the values of the weights and layers can only have a limited number of values(for example, -1 or +1). These architectures enable a sizable reduction in memory consumption while preserving accuracy. In addition, they allow for a replacement of heavy operations with lightweight bitwise ones, which makes them hardware friendly.

Unlike other DNNs, BNNs use only Linear and Sign activation functions to reduce the number of floating-point arithmetic computations, which increases performance. This makes BNNs easy to deploy on devices that have limited computational and memory resources. For example, binary neural networks can beused on mobile devices to perform image recognition for various applications [21].

They can also be used to perform advanced speech recognition restricting word error rate [22]. Similarly, they can be implemented as part of software for robot adapted microcontrollers. For example, BNNs can be deployed in FPGAS, which

requires architectures and hardware adjustments [23]. Besides, they can serve as accelerators for parallelization of processes in embedded systems [24]. Similar to DNNs, BNNs are vulnerable to adversarial attacks [25, 26] in which slightly changing the inputs can completely fool a well-trained and highly accurate network. Adversarial attacks on DNNs exploit their vulnerabilities to input that underwent slight perturbations. These perturbations reveal significant security and reliability challenges in DNN-based applications [27] by creating adversarial examples. Adversarial examples cause DNNs to misclassify them without affecting human perception [28]. Multiple techniques have been proposed to generate adversarial examples, including white-box ones (the architecture and weights are available in advance) and black-box ones (the attacker has no knowledge of the model's internals). To defend against these attacks, the researchers have introduced techniques that include adversarial training to improve their robustness [29], and neural network verification approaches that aim to mathematically prove that for a given input space, the network's predictions remain stable [30].

While being efficient and easy to deploy, BNNs are generally more challenging to train and verify because of a performance-accuracy trade-off [31]. Only a few neural network verification methods have been proposed to deal with BNNs, and most of them require input quantization, which omits an infinite number of possible input states. For instance, the BDD-based methods [32] perform quantitative robustness analysis of BNNs based on constructing equivalent binary decision diagrams from the networks with quantized input data. The EEVBNN tool[2] can perform neural network verification for BNNs with quantized input space by converting the networks into SAT problems. It is important to emphasize that input quantization is an extra man-made step to ease neural network verification.

By reducing an input space to a discrete set, quantization makes the verification process more efficient with respect to computational time and resources. However, input quantization causes a decrease in the accuracy of neural networks after training, especially in those used for control purposes [33, 34], because originally, the networks were trained to work on continuous input space. Verifying BNNs on continuous input space allows accounting for all possible input states, increasing the certainty of the performed verification. The SAT/SMT-based neural network verification methods proposed in [1, 35] can verify BNNs on continuous input space, but they are not scalable.

## II. LITERATURE SURVEY

| Topic | Focus Area | Key Contributions | References |
|---|---|---|---|
| **Binary Neural Networks (BNNs)** | Definition and Properties | BNNs use binary weights and activations, reducing model size and computation costs while maintaining reasonable accuracy. | Hubara et al. (2016), "Binarized Neural Networks" |
| **Continuous Inputs in BNNs** | Handling Continuous Data | Examines methods for processing continuous inputs in BNNs, addressing accuracy drops when input data is not binary. | Courbariaux et al. (2015), "BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagation" |
| **Reachability Analysis** | Definition and Importance | Reachability analysis helps verify the behavior of neural networks by exploring all possible outputs given a range of inputs. | Gehr et al. (2018), "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation" |
| **Star Methods** | Definition of Star Methods | Star methods represent input regions in high-dimensional spaces, enabling efficient reachability analysis by capturing possible output ranges. | Tran et al. (2019), "Star-Based Reachability Analysis of Neural Networks" |
| **Reachability in Continuous Inputs for BNNs** | Challenges & Solutions | Analyzes the difficulty of reachability in BNNs with continuous inputs, focusing on adaptation methods using star approaches. | Xiang et al. (2018), "Reachability Analysis and Robustness Verification of Binarized Neural Networks" |

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 1, 2025

## III. PREVIOUS RELATED WORK DONE

### 1. Formal Verification and Reachability Analysis in Neural Networks

**Reluplex**: Katz et al. introduced the *Reluplex* algorithm in "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks" (2017), which extended the Simplex method to support ReLU activations. This work paved the way for verifying safety and robustness properties of neural networks by formalizing constraints, though it was originally geared towards networks with continuous weights.

**Layer-based Approximations**: Works such as "The AI2 Network Verification Tool" by Gehr et al. (2018) introduced layer-wise abstraction methods for reachability, applying interval bound propagation (IBP) and zonotope techniques to approximate reachable sets efficiently.

### 2. Polyhedral and Star Methods in Neural Networks

**Polyhedral Methods**: In "Reachability Analysis for Neural Networks with ReLU Activations" by Xiang et al. (2017), polyhedral and star set representations were used to represent input uncertainty. This work influenced further adaptations for quantized and binary networks by exploring how linear constraints could propagate through ReLU layers, establishing a basis for continuous input reachability analysis.

**Star Set Methodology**: Tran et al. developed the *Star Set* method for reachability analysis in "Star-Based Reachability Analysis of Deep Neural Networks" (2019), which improved over earlier polyhedral approaches by handling larger input spaces and deeper networks. Although not initially for BNNs, this method has been foundational in reachability analysis for all types of networks.

### 3. Verification and Robustness Analysis in Quantized and Binarized Networks

**Binary and Quantized Network Verification**: Narodytska et al.'s paper "Verifying Properties of Binarized Deep Neural Networks" (2018) tackled verification challenges specifically for binarized neural networks. This research used SAT-based solvers to verify properties and robustness of binarized networks, addressing reachability and safety in binary settings.

**Adversarial Robustness in BNNs**: Another key work by Raghunathan et al., "Certifying Robustness to Adversarial Examples with Interval Bound Propagation" (2018), explored methods for certifying robustness in networks using quantized activations. Although designed for quantized rather than fully binarized weights and activations, it helped establish bounds for adversarial robustness that could be extended to BNNs.

### 4. Optimized Reachability Analysis for Efficiency

**Layer-Wise Symbolic Propagation**: Huang et al., in "Safety Verification of Deep Neural Networks" (2017), introduced symbolic propagation techniques that efficiently computed reachability by representing inputs as symbolic variables. This approach, while computationally efficient, was best suited for shallow networks with continuous activations, leaving an opportunity to refine for binarized structures.

**Differentiable Approximations for Reachability**: Techniques involving *differentiable reachability*, such as in Gowal et al.'s work on "On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models" (2019), introduced efficient, scalable methods using differentiable approximations. This was particularly impactful in reducing computational cost and supporting scalability in real-time applications.

### 5. Reachability and Robustness in Binary Neural Networks

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 1, 2025

**Binary Neural Network Frameworks**: Papers like "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks" by Rastegari et al. (2016) and "Binarized Neural Networks" by Courbariaux et al. (2016) laid groundwork for BNN development, emphasizing memory efficiency and faster inference at the cost of precision. Although not focused on reachability, these works highlighted specific challenges and constraints within BNNs that affect how reachable sets are computed.

**Geometric Approximations for Binary Layers**: Liu et al.'s "Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers" (2019) proposed ways to geometrically approximate reachable sets even in networks with quantized or binary layers. Their approach also helped define boundaries for binary reachability analysis.

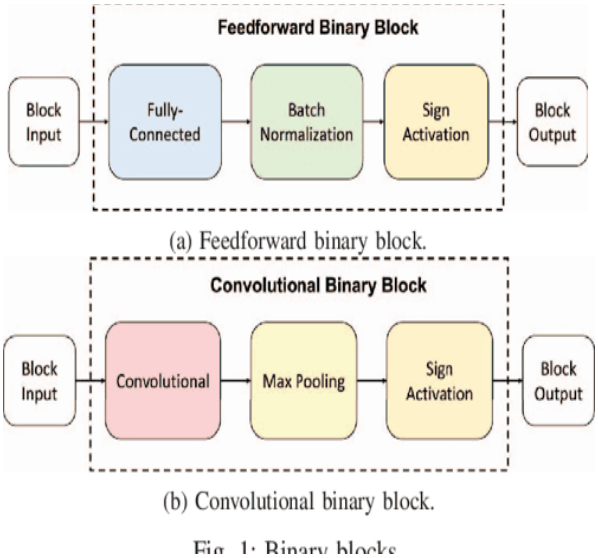
## IV. PURPOSE OF THE WORK

1) Identify potential vulnerabilities or strengths in BNNs to improve their reliability and robustness in real-world scenarios.
2) Utilize star methods as a systematic approach for reachability analysis, providing a framework for understanding input uncertainties.

## V. THE PROPOSED WORK

When developing the proposed algorithms, our goal is to guarantee that they are more efficient and precise than existing solutions. This allows us to show that the research proposed in this thesis is truly meaningful. In addition, gains in efficiency and accuracy allow the proposed techniques to be more suitable for larger networks, batches of data, and disturbance. This would show the advantage of the proposed approaches over existing state-of-the-art benchmarks. In sum, the main contributions of this thesis are: The extension of the Star reachability algorithms for verifying BNNs on continuous input space. The implementation of exact reachability analysis and over approximate reachability analysis algorithms in NNV that are publicly available for further evaluation and comparison. · A thorough evaluation of the proposed approach in comparison with Marabou and EEVBNN on a set of three datasets (MNIST, FMNIST, CIFAR10), and ten binary neural networks.

This thesis is organized as follows: Chapter 2 covers the related published research, Chapter 3 describes the foundation of Star-based reachability analysis and BNN neural network verification, Chapter 4 showcases the evaluation of the7proposed approach compared to the existing techniques, Chapter 5 presents a discussion of the work along with the conclusions.




(a) Feedforward binary block.

(b) Convolutional binary block.

Fig. 1: Binary blocks

Figure 1: Binary Blocks

*The Reachability Algorithm:*

1: **procedure** reach(N, Θ ,method)

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 1, 2025

2: R ←Θ
3: **for** <i = 1 to k>**do**
4: Li ←N.Layers(i)
5: R ←Li.reach(R,method)
6:Return R

Table I: The architectures of MLP networks

| Network | Architecture | Accuracy | Type |
|---|---|---|---|
| MLP0 | $(50 \times 4) : (10 \times 2)$ | 90% | BFFNN |
| XNOR0 | 3CB:1FB | 75% | BCNN |
| MLP1 | $(200 \times 2) : (100 \times 2) : (50 \times 2) : (10 \times 2)$ | 96% | BFFNN |
| MLP2 | $(200 \times 3) : (100 \times 2) : (50 \times 2) : (10 \times 2)$ | 96% | BFFNN |
| MLP3 | $(200 \times 3) : (100 \times 3) : (50 \times 2) : (10 \times 2)$ | 96% | BFFNN |
| MLP4 | $(200 \times 4) : (100 \times 3) : (50 \times 2) : (10 \times 2)$ | 96% | BFFNN |

b

Table II:Verification results of MLP0 network.

| $\delta$ | Marabou | | | | | Exact-Star | | | | Approx-Star | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time(s) | | #Sol | | | Time(s) | | #Sol | | Time(s) | | #Sol | |
| | UN | S | UN | S | UK | UN | S | UN | S | UN | S | UN | S |
| 0.1 | 7.42 | 68 | 47 | 1 | 0 | 0.8 | 0.8 | 48 | 0 | 0.5 | 0.5 | 48 | 0 |
| 0.15 | 13.9 | 16 | 40 | 2 | 0 | 0.9 | 0.9 | 41 | 1 | 0.5 | 0.5 | 41 | 1 |
| 0.2 | 13.06 | 115 | 43 | 1 | 0 | 0.9 | 0.9 | 44 | 0 | 0.5 | 0.5 | 44 | 0 |
| 0.3 | 69.69 | 128 | 40 | 3 | 0 | 0.9 | 0.9 | 42 | 1 | 0.5 | 0.5 | 42 | 1 |
| 0.5 | 457.29 | 314 | 33 | 9 | 0 | 0.9 | 0.9 | 41 | 1 | 0.5 | 0.5 | 41 | 1 |
| 1 | 1809.09 | 2889 | 25 | 13 | 8 | 0.8 | 0.8 | 42 | 4 | 0.5 | 0.5 | 42 | 4 |
| 3 | TO | 2432 | 0 | 25 | 14 | 0.8 | 0.8 | 36 | 3 | 0.5 | 0.5 | 36 | 3 |
| 5 | TO | 702 | 0 | 43 | 0 | 0.8 | 0.8 | 25 | 18 | 0.5 | 0.5 | 25 | 18 |
| 10 | TO | 441 | 0 | 40 | 0 | 0.8 | 0.8 | 18 | 22 | 0.5 | 0.5 | 18 | 22 |
| 15 | TO | 528 | 0 | 49 | 0 | 0.8 | 0.8 | 20 | 29 | 0.5 | 0.5 | 20 | 29 |

cation results of MLP0 network. *Notation*: $\delta$ represents disturbance bound values. Time(s) rep

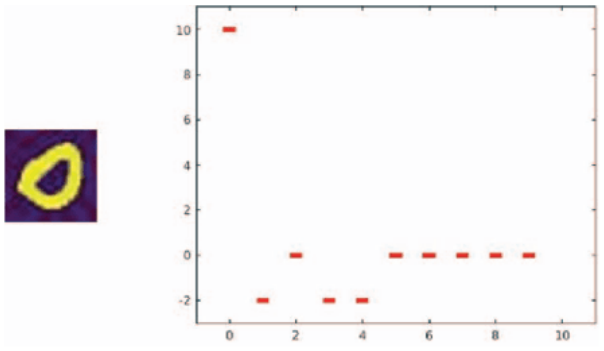



Figure 2: MLP0 verification mismatch for $\delta = 0.1$.

## VI. IMPLEMENTATION

In the exact analysis, a max-pooling or a Sign layer may produce multiple output sets from an input set. Therefore, we exploit the power of parallel computing to process multiple inputs simultaneously at a specific layer to speed up the verification. In addition, we usually use estimated ranges to determine the Sign of individual inputs in the reachability of a Sign layer to minimize unnecessary optimization time in the analysis. For example, if we know the estimated lower bound of xi is ˜li ≥ 0, then we do not need to find its exact lower bound li for the analysis as it is always non-negative xi ≥ li ≥˜li ≥ 0.Finally, we note that if a BNN is a BFFNN, a more efficient implementation using Depth-First Search (DFS) with exact reachability [37] can be used to verify the

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 1, 2025

network. Compared to the Breadth-First Search (BFS) implementation in this thesis to handle both BFFNNs and BCNNs, BFS is faster and more memory-efficient in searching a Counterexample when verifying the network. The algorithm will stop immediately once a counterexample is found. Using the reachable set computed in the previous section, verifying the safety of BNNs defined in the following is straight forward an unsafe specification U defined by a set of linear constraints on the network's outputsU ≜ {y | Cy ≤ d}, the network is called to be safe corresponding to the input set ×, if and only if R ∩ U = ∅, where R is the network's reachable set, i.e., R = N(×). Otherwise, the neural network is unsafe. Similar to verification of ReLU networks [36], wecan construct a complete set of counterexamples that makes a BNN unsafe if theexact reachability method is used. This is described in the following lemma.Let R = [Θ1, Θ2, . . . , ΘN] be the exact reachable set of a BNN N witha Star input set Θ = ⟨c,V, P⟩, i.e., R = N(Θ), and U ≜ {y | Cy ≤ d} be the unsafe specification of the network. If the network is unsafe, i.e., R ∩ U /= ∅, then a complete setof counterexample inputs C is computed as follows: · ∀k = 1, 2, . . . , N, Θk ∩ U = Θ′ k = ⟨c′ k,V′ k , P′ k ⟩/= ∅ (Proposition 3.1.3)Ck = ⟨Θ.c, Θ.V, P′ k ⟩, C ← Ck.In the exact reachability of a BNN, the input set and output set are defined based on the same set of predicate variables unchanged in the computation. When splitting occurs, new constraints on the predicate variables are Therefore, a Star set in the network's reachable set contains all constraints appearing in the input set, i.e., Θk.Pk⊆Θ.P. When a Star set Θk in the network's reachable set intersects with the unsafe region U,the intersection is an unsafe output set of the network, which is also a Star set Θ′ k = ⟨c′ k,V′ k , P′ k ⟩ (Proposition 3.1.3). Importantly, we have P′ k ⊆ Pk ⊆ P. Therefore,any input vectors corresponding to any predicate vectors α = [α1, . . . , αm]T ∈ P′ kcause the network to be unsafe. In other words, the Star Ck = ⟨Θ.c, Θ.V, P′ k ⟩ isa set of counterexamples of the network. We can construct a complete set of counter examples by checking the intersection of all Star sets in the reachable setwith the unsafe region U.

## VII. EXPERIMENTAL RESULTS

Note that on the given examples, the Star-based exact verification approach runs out of time and memory. For this reason, we only present the comparison with the over approximate analysis algorithm. The verification results of the EEVBNN method's proposed benchmarks are presented. Compared to EEVBNN, Star underperforms both with regard to the timing and the number of solved examples. **Timing Performance.** The experiments show that EEVBNN can be

| Network | δ | Times | Marabou | | | Exact-Star | | | | Approx-Star | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #Sol | | | Time(s) | | #Sol | | Time(s) | | #Sol | | |
| | | | UN | S | UK | UN | S | UN | S | UN | S | UN | S | UK |
| MLP1 | 0.1 | TO | 0 | 0 | 50 | 2.6 | - | 50 | 0 | 1.6 | - | 50 | 0 | 0 |
| | 0.15 | TO | 0 | 0 | 50 | 2.6 | 2.78 | 49 | 1 | 1.59 | 1.56 | 49 | 1 | 0 |
| | 0.2 | TO | 0 | 0 | 50 | 2.68 | - | 50 | 0 | 1.58 | - | 50 | 0 | 0 |
| | 0.3 | TO | 0 | 0 | 50 | 2.52 | 2.58 | 46 | 4 | 1.6 | 1.6 | 46 | 3 | 1 |
| | 0.5 | TO | 0 | 0 | 50 | 2.51 | 2.46 | 49 | 1 | 1.6 | 1.6 | 49 | 0 | 1 |
| | 1 | TO | 0 | 0 | 50 | 2.55 | 2.57 | 49 | 1 | 1.56 | 1.57 | 49 | 1 | 0 |
| | 3 | TO | 0 | 0 | 50 | 2.8 | 3.03 | 48 | 2 | 1.66 | 1.66 | 48 | 2 | 0 |
| | 5 | TO | 0 | 0 | 50 | 3.25 | 3.14 | 46 | 4 | 1.68 | 1.7 | 46 | 4 | 0 |
| | 10 | TO | 0 | 0 | 50 | 3.07 | 3.53 | 35 | 15 | 1.68 | 1.66 | 35 | 15 | 0 |
| | 15 | TO | 0 | 0 | 50 | 2.77 | 2.8 | 31 | 19 | 1.67 | 1.67 | 31 | 19 | 0 |
| MLP2 | 0.1 | TO | 0 | 0 | 50 | 3.46 | - | 50 | 0 | 1.98 | - | 50 | 0 | 0 |
| | 0.15 | TO | 0 | 0 | 50 | 3.39 | - | 50 | 0 | 1.93 | - | 50 | 0 | 0 |
| | 0.2 | TO | 0 | 0 | 50 | 3.51 | - | 50 | 0 | 2.01 | - | 50 | 0 | 0 |
| | 0.3 | TO | 0 | 0 | 50 | 3.46 | 3.3 | 49 | 1 | 2.09 | 2.2 | 49 | 0 | 1 |
| | 0.5 | TO | 0 | 0 | 50 | 3.44 | 4.89 | 49 | 1 | 2.1 | 2.4 | 49 | 0 | 1 |
| | 1 | TO | 0 | 0 | 50 | 4.3 | 5.38 | 49 | 1 | 2.3 | 2.4 | 49 | 0 | 1 |
| | 3 | TO | 0 | 0 | 50 | 4.19 | 3.48 | 48 | 2 | 2.15 | 2.05 | 48 | 2 | 0 |
| | 5 | TO | 0 | 0 | 50 | 3.4 | 3.54 | 46 | 4 | 2.13 | 2.13 | 46 | 4 | 0 |
| | 10 | TO | 0 | 0 | 50 | 2.94 | 2.93 | 39 | 11 | 2.16 | 2.15 | 39 | 11 | 0 |
| | 15 | TO | 0 | 0 | 50 | 3.03 | 3.02 | 32 | 18 | 2.2 | 2.2 | 32 | 17 | 0 |
| MLP3 | 0.1 | TO | 0 | 0 | 50 | 3.51 | - | 50 | 0 | 3.9 | - | 50 | 0 | 0 |
| | 0.15 | TO | 0 | 0 | 50 | 3.42 | - | 50 | 0 | 3.8 | - | 50 | 0 | 0 |
| | 0.2 | TO | 0 | 0 | 50 | 3.45 | 4.48 | 48 | 2 | 3.5 | 3.3 | 48 | 2 | 0 |
| | 0.3 | TO | 0 | 0 | 50 | 3.44 | 3.27 | 49 | 1 | 3.5 | 3.2 | 49 | 1 | 0 |
| | 0.5 | TO | 0 | 0 | 50 | 3.6 | 3.51 | 47 | 3 | 3.6 | 3.3 | 47 | 1 | 2 |
| | 1 | TO | 0 | 0 | 50 | 4.36 | 4.71 | 48 | 2 | 3.8 | 3.5 | 48 | 1 | 1 |
| | 3 | TO | 0 | 0 | 50 | 4.09 | 6.03 | 48 | 2 | 3.8 | 3.5 | 48 | 2 | 0 |
| | 5 | TO | 0 | 0 | 50 | 3.5 | 3.04 | 47 | 3 | 3.6 | 3.6 | 47 | 3 | 0 |
| | 10 | TO | 0 | 0 | 50 | 3.03 | 3.02 | 38 | 12 | 3.7 | 3.9 | 38 | 12 | 0 |
| | 15 | TO | 0 | 0 | 50 | 3.03 | 3.14 | 26 | 24 | 3.6 | 3.8 | 26 | 24 | 0 |
| MLP4 | 0.1 | TO | 0 | 0 | 50 | 4.21 | - | 50 | 0 | 3.3 | - | 50 | 0 | 0 |
| | 0.15 | TO | 0 | 0 | 50 | 4.23 | 4.08 | 49 | 1 | 3.2 | 3.3 | 49 | 0 | 1 |
| | 0.2 | TO | 0 | 0 | 50 | 4.17 | 4.1 | 49 | 1 | 3.2 | 3.3 | 49 | 0 | 1 |
| | 0.3 | TO | 0 | 0 | 50 | 4.13 | 4.52 | 47 | 3 | 3.1 | 3.2 | 47 | 1 | 2 |
| | 0.5 | TO | 0 | 0 | 50 | 5.09 | - | 50 | 0 | 3.2 | - | 50 | 0 | 0 |
| | 1 | TO | 0 | 0 | 50 | 4.9 | 5.06 | 48 | 2 | 3.6 | 3.6 | 48 | 0 | 2 |
| | 3 | TO | 0 | 0 | 50 | 3.89 | 3.64 | 49 | 1 | 3.6 | 3.6 | 49 | 1 | 0 |
| | 5 | TO | 0 | 0 | 50 | 3.63 | 3.59 | 46 | 4 | 3.6 | 3.7 | 46 | 3 | 1 |
| | 10 | TO | 0 | 0 | 50 | 3.43 | 3.26 | 42 | 8 | 3.5 | 3.7 | 42 | 7 | 1 |
| | 15 | TO | 0 | 0 | 50 | 3.15 | 3.14 | 35 | 15 | 1.16 | 1.17 | 35 | 15 | 0 |

Table III: Verification results for MLP1-4.TABLE III: Verification results for MLP1-4. Notations are the same with that of Table II

Table IV:Verification results for the networks presented by EEVBNN

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 19, Issue 1, 2025

| Network | δ | EEVBNN Time(s) UN | EEVBNN Time(s) S | EEVBNN #Sol UN | EEVBNN #Sol S | Approx-Star Time(s) UN | Approx-Star Time(s) S | Approx-Star Time(s) UK | Approx-Star #Sol UN | Approx-Star #Sol S | Approx-Star #Sol UK |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **mnist-small** | 0.1 | 0.021 | 0.022 | 436 | 64 | 0.09 | 0.079 | 0.078 | 42 | 19 | 439 |
| **mnist-large** | 0.1 | 0.164 | 0.177 | 454 | 46 | 1.191 | 0.881 | 0.97 | 42 | 10 | 448 |
| **mnist-small** | 0.3 | 0.027 | 0.029 | 312 | 188 | 0.103 | 0.088 | 0.09 | 42 | 31 | 427 |
| **mnist-large** | 0.3 | 0.169 | 0.171 | 379 | 121 | 1.915 | 1.634 | 1.55 | 42 | 21 | 437 |
| **cifar10-small** | 2/255 | 0.046 | 0.047 | 128 | 372 | 0.273 | 0.267 | 0.27 | 57 | 228 | 215 |
| **cifar10-large** | 2/255 | 0.304 | 0.341 | 147 | 353 | 3.032 | 3.04 | 3.049 | 57 | 226 | 217 |
| **cifar10-small** | 8/255 | 0.064 | 0.068 | 94 | 406 | 1.131 | 1.152 | 1.174 | 57 | 270 | 173 |
| **cifar10-large** | 8/255 | 0.342 | 0.372 | 123 | 377 | 9.589 | 9.721 | 9.665 | 57 | 272 | 171 |

TABLE VI: Verification results for the networks presented by EEVBNN [9]

from 4× to30× faster than Star, depending on a model’s size and the used disturbance value. For example, EEVBNN verifies all 500 examples for **cifar10-small** 6× faster than Star. This happens because Star reachability is aimed at handling continuous input while EEVBNN works with the quantized one. Star’s ability to operate incontinuous space introduces a trade-off as its computational operations are more complex. In addition, EEVBNN tests its approach on ’ solver-friendly’ networks that contain high-sparsity weights.

**Conservative ness.** According to the experiments, EEVBNN solves all the examples, while Star is only able to solve ≈ 14% for the MNIST-trained models and ≈ 88%for CIFAR10-trained models. This indicates that EEVBNN is less conservative compared to the Star-based reachability method.Although our approach is efficient and41scalable for BFFNNs, it is not scalable for BCNNs with max-pooling layers. Asanalyzed in [38], when dealing with large disturbance bounds, more predicate variables and their associated generators are introduced in the reachability of a max-pooling layer. This causes an explosion in memory and computation time.It is worth emphasizing that BCNNs using average pooling can achieve the same(or even better) accuracy and are amenable to our verification approach [63]. Wehave tried analyzing BCNNs with average pooling using our approach. However, we could not compare with Marabou on these networks as Marabou does not currently support average pooling. In addition, the given representation of Star cannot be efficiently used with quantized input space. For this reason, the method implemented in EEVBNNout performs Star in terms of timing and conservativeness. However, we emphasize that Star reachability algorithms have been designed to work with continuous input space. While it requires the operations to be more computationally expensive, it allows Star to generalize better as it deals with continuous (infinite but bounded) input space instead of quantized input space with finite states like

EEVBNN. In addition, quantization introduces various sources of errors (rounding,computational noise, etc.). All of this may not have an effect when verifying ” basic” benchmarks like MNIST or CIFAR10 but could have a huge impact in real-world tasks. Note that EEVBNN also uses ” solver-friendly BNNs” . These BNNs’ weights sparsity is artificially increased during the training process. Such an approach mayalso increase error accumulation. Thus, we believe that Star reachability is a good complementary approach when input quantization is not an option.

This section summarizes the results and how they attempt to answer the research questions posed in this thesis:

1.Can we develop a Star-based reachability analysis technique that would allow for verifying binary neural networks?

It shows that it is possible to verify BNNs using Star by putting together reachability algorithms that can compute a reachable set of the Sign activation layer. This initiated the development of the exact and over approximate reachability algorithms for the Sign layer. The obtained Star-based technique for BNN verification was tested on several benchmarks and compared to the existing Marabou framework. We included the developed approach into NNV, a neural network verification tool for DNNs and learning-enabled Cyber-Physical Systems.

2. Can we guarantee the soundness and completeness of Star-based BNN verification**?**

To answer this question, we address the original definitions of the ERA and ORA algorithms. Exact reachability guarantees soundness and completeness, while the over approximate reachability algorithm is sound but will not be complete.To compensate for the incompleteness of the over approximate algorithm, we construct counterexamples based on the original input and test them against the network. Even though it does not guarantee completeness due to the randomness of the process, the experiments show that such an approach allows us to identify quite a few input examples that can be successfully used for adversarial attacks.

3. Can the proposed technique be efficient enough to outperform existing approaches that verify neural networks on continuous input space?

we evaluate our approach on the respective benchmarks when comparing it to existing solutions. To compare our approach to Marabou, which works with networks on continuous input space, we evaluate exact and over approximate reachability analysis on 5 BFFNNs (MLP0−4) and 1 BCNN(XNOR0) trained on MNIST and FMNIST datasets, respectively. We evaluateMLP0−4 on 500 examples for 10 different disturbance values, while XNOR0is evaluated on 300 examples for 6 disturbance values. The experiments show that our approach is significantly faster than Marabou on their proposed benchmarks. For instance, the exact and over approximate verification can be3600× and 5700× faster than Marabou on a small network with 220 neurons. Additionally, our approach is also less conservative and more efficient than Marabou when dealing with severe $L\infty$ norm attacks, i.e., attacks with large disturbance bound $\delta$. For example, Marabou reaches a timeout of 5,000seconds when verifying the small network with $\delta > 1$ while our approach can prove the robustness of the network within 1 second. However, the proposed approach underperforms when compared to EEVBNN on the given benchmarks. It is understandable as EEVBNN works with quantized input space, which is not the primary target of this thesis.

## VIII. CONCLUSION

In this thesis, we have extended the star reachability algorithms for verifying BNNs with continuous input space. The proposed exact and over approximate reachability algorithms were compared to two existing frameworks − Marabou and EEVBNN − and evaluated using 9 different BNNs trained on three datasets:MLP0−4 and XNOR0 for Marabou trained on MNIST and FMNIST respectively,and mnist-small, mnist-large, cifar10-small, cifar10-large for EEVBNN trained on MNIST and CIFAR10. For MLP0−4 we used the following disurbance values{0.1, 0.15, 0.2, 0.3, 0.5, 1, 3, 5, 10, 15}. For XNOR0 - {0.05, 0.1, 0.15, 0.2, 0.25, 0.3}. For mnist-small, mnist-large - {0.1, 0.3}, for cifar10-small, cifar10-large - {2/255, 8/255}.The experiments show that the proposed method is more efficient and scalable than the SMT-based approach implemented in Marabou. On smaller BFFNNs, the exact and over approximate verification algorithms can be 3600× and 5700× faster than Marabou. For XNOR0, we demonstrate that the Star-based approach is 44.6×and 3877.25× faster than Marabou. On larger BFFNNs and higher disturbance values, Marabou keeps running into the timeout without providing any solutions. Our approach also proves to be less conservative than Marabou.

## REFERENCES

[1] G. Amir, H. Wu, C. Barrett, and G. Katz, "An SMT-based approach forverifying binarized neural networks," 2021. (document), 1, 2.1.3, 2.2, 4, 4.1, 4.1

[2] K. Jia and M. Rinard, "Efficient exact verification of binarized neuralnetworks," 2020. [Online]. Available: https://arxiv.org/abs/2005.03597(document), 1, 2.2, 4.2, 4.5, 4.6

[3] J. Watson, M. Firman, A. Monszpart, and G. J. Brostow, "Footprints and freespace from a single color image," 2020. 1

[4] N. Babanejad, A. Agrawal, A. An, and M. Papagelis, "A comprehensiveanalysis of preprocessing for word representation learning in affective tasks," in Proceedings of the *58*th Annual Meeting of the Association for ComputationalLinguistics. Online: Association for Computational Linguistics, July 2020, pp.5799‑5810. [Online]. Available: https://aclanthology.org/2020.acl-main.514 1

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborativefiltering," 2017. 1[6] N. PP and P. N. I. Anantharaman, "A survey of research in deep learning forrobotics for undergraduate research interns," 2023. 1

[7] O. Faust, Y. Hagiwara, J. H. Tan, S. L. Oh, and U. R. Acharya, "Deep learningfor healthcare applications based on physiological signals: A review," Computer Methods and Programs in Biomedicine, vol. 161, 04 2018. 1

[8] I. Attri, L. K. Awasthi, T. P. Sharma, and P. Rathee, "A review of deeplearning techniques used in agriculture," Ecological Informatics, vol. 77, p.102217, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954123002467 1

[9] T. Akinosho, L. Oyedele, M. Bilal, and O. Akinad´e, "Deep learning in theconstruction industry: A review of present status and future innovations," Journal of Building Engineering, vol. 32, 09 2020. 1

[10] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "Asurvey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11‑26, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216315533 1

[11] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, "Model complexity of deep learning:A survey," 2021. 1

[12] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of acceleratorarchitectures for deep neural networks," Engineering, vol. 6, no. 3, pp. 264‐274,2020. [Online]. Available:https://www.sciencedirect.com/science/article/pii/S2095809919306356 1

[13] D. Shriver, S. Elbaum, and M. B. Dwyer, "Reducing dnn properties to enablefalsification with adversarial attacks," pp. 275‐287, 2021. 1

[14] A. Bulat, B. Martinez, and G. Tzimiropoulos, "High-capacity expert binarynetworks," 2021. 1

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenetclassification using binary convolutional neural networks," 2016. 1

[16] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarizedneural networks: Training deep neural networks with weights and activationsconstrained to +1 or -1," 2016. 1