Deep Neural Network Verification

Srikar Chittari
ssc7dvy
Computer Science
University of Virginia

Scott Sikorski
nqj5ak
Computer Science
University of Virginia

ABSTRACT

Deep Neural Networks (DNNs) have revolutionized the field of Machine Learning and consequently countless fields such as computer vision, natural language processing, and autonomous vehicles. While these networks achieve unparalleled performance in complex tasks, the black-box nature of DNNs introduce concerns regarding interpretability of decision-making, prediction biases, and security against adversarial attacks. Researchers and practitioners have looked to employ DNN verification tools to mitigate these concerns by evaluating the robustness of networks to expose vulnerabilities in models. However, verifiers trail in development behind cutting-edge models due to the rapidly evolving field of DNN research. Looking to close this gap, there have been growing efforts to improve verification tools by determining their applications and shortcomings. This paper aims to take a step toward better understanding the strengths and weaknesses of DNN verifiers when applied to a variety of network architectures. In this paper, several DNNs are selected as benchmarks to determine the effects of network architectures and robustness properties on network verification. State-of-the-art DNN verification tools, α-β-CROWN and NeuralSAT, are utilized to verify benchmarks and are compared in their abilities to verify networks effectively and efficiently. Parameters of networks such as the number of hidden layers, activation functions, layer types, and degree of perturbation are varied to study their relationships with network verification and verification time.

KEYWORDS

Deep Neural Networks, DNN Verification, Adversarial robustness, Alpha-Beta-Crown, NeuralSAT

ACM Reference Format:
Srikar Chittari and Scott Sikorski. 2024. Deep Neural Network Verification. In . ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep Neural Networks (DNNs) are ubiquitous in today's world due to their ability to perform complex tasks and have been increasingly integrated in safety critical systems such as autonomous vehicles, medical diagnosis tools, aviation systems, and power grid management systems. In such systems, even the smallest accidents and failures can have severe consequences [28]. To prevent this, classically programmed components (non-machine learning systems) are tested rigorously with a variety of software analysis tools that have been developed over several decades. Machine learning (ML), on the other hand, is a relatively new paradigm and is a rapidly evolving field of research where major improvements are made each year [29]. This trend is especially seen in DNN research where cutting edge DNNs are developed and deployed at a faster rate than the development of formal DNN verification techniques. The lag between development and testing exacerbates existing concerns with DNNs such as the transparency of decision making, bias mitigation, and security against malicious attacks [11].

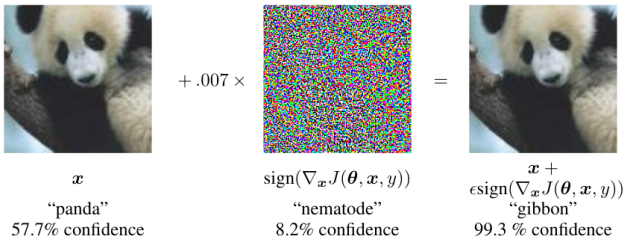

Figure 1: An adversarial example applied to GoogLeNet on ImageNet published by Goodfellow et al. [10]

DNN verification has become an increasingly important field of research as it aims to address the need for reliable, trustworthy, and secure ML models. One of the primary focuses of DNN verification is demonstrating the network's robustness to adversarial attacks. An adversarial attack is when an input is intentionally crafted in such a way that maliciously deceives the network. Discovering adversarial examples while testing a DNN exposes the network's blind spots in the data distribution, and the generation of adversarial examples has even been integrated into model training to produce more robust networks [1].

There are many examples of adversarial attacks. A well-known example in the literature is shown in Figure 1 which was published by Goodfellow et al. [10]. In this example, an image of a panda is perturbed by adding random noise. The perturbation causes the input image to cross the decision boundary of the model and get classified as a gibbon despite still being an image of a panda. Figure 2 shows an example of an adversarial attack in a safety critical system which was published by Metzen et al. [12]. In this example, an image of pedestrians on a street is perturbed, and the model classifies this modification as an empty street. The consequences

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

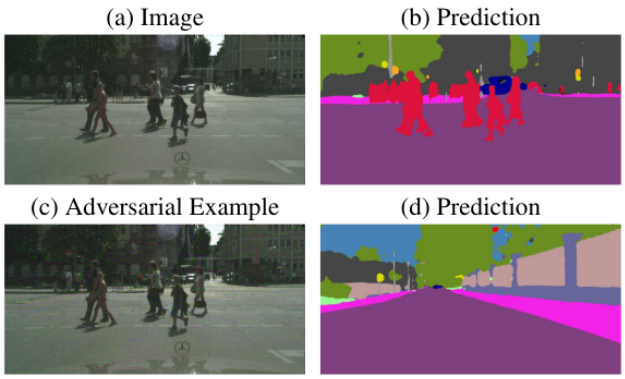

Figure 2: An adversarial example in a safety critical system published by Metzen et al. [12]

of a classification error in this system would be far greater, highlighting the importance of DNN verification tools to expose these adversarial examples.

In this paper, we make the following contributions.

(1) Select a set of networks as benchmarks that represent a variety of DNN architectures and that facilitate isolated analysis of network parameters.
(2) Evaluate the effects of network architectures and robustness properties on the network verification. More specifically, analyze the effects of the number of hidden layers, activation functions, and layer types, and degree of perturbation on the verification result and verification time.
(3) Compare the robustness verification performances of state-of-the-art DNN verifiers, α-β-CROWN [27] and NeuralSAT, [7] on the benchmarks in regards to verification result and verification time.

2 BACKGROUND

There are a variety of properties that can be verified by these DNN verifiers. These include but are not limited to temporal, fairness, robustness, and adversarial example detection. Temporal checks that the network responds to any data distribution shift that may happen as a result of time. Fairness ensures that protected attributes such as race, age, gender, or sexual orientation do not influence predictions with equal metrics. These two are not the focus of this work but illustrate the aspects of models that developers must consider. Robustness and adversarial example detection are defined by the model's ability to maintain accurate performance when given noise or perturbations to the image. We expect that the model predicts the same label for two inputs that hardly differ. If a model is not robust, it can be exploited with adversarial examples. Detecting these examples is not usually handled by the model itself. Rather a part of the verification process is identifying instances that are considered adversarial based on the amount of noise added. Our goal is to ensure that model can handle any noise from real world conditions or adversarial examples.

A robust model ensures that the model can both generalize well to unseen data and not be sensitive to environmental changes during inference. We define a value ε to determine the robustness. This ε can be seen as the radius of a ball that extends in all directions of the input dimensional space. Shown in Figure 3 [15], the blue point represents our input in space. ε extends the value of the point to create a ball where the input can exist in. For $\varepsilon << 1$, this window is very tight and represents a trivial problem to verify as it is extremely close to the original input. As we increase ε, the problem becomes harder as the verifier must explore this new input space. Eventually, ε reaches a point where it crosses the decision boundary causing an incorrect prediction. This is the orange point. The verifiers can then use attack strategies to easily identify those adversarial examples. Verifying robustness ensures that our model is not susceptible to these small perturbations in the input.

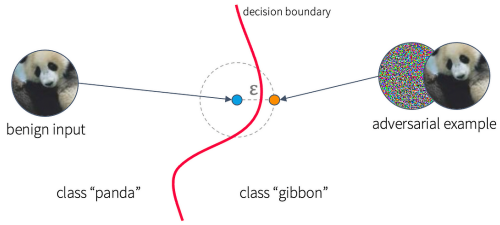

Figure 3: ε-radius Ball for Robustness [15]

For this work, we picked α-β-CROWN [27], [23], [18], [24], [20], [26], [25], [19], [14] and NeuralSAT [7], [8] as the two verifiers to analyze a variety of benchmarks on. We chose these specifically because they both provide GPU acceleration support and performed well on related verification competitions (α-β-CROWN - 1st, NeuralSAT - 4th at VNNComp 2023). They offer two different forms of verification, the former having a large learning curve and the former being easy to use right away.

α-β-CROWN is a powerful and efficient linear bound propagation framework with GPU optimized relaxation and parallel branch and bound methods. The property constraints are defined according to the input, per pixel for an image. These constraints form linear inequalities which are propagated backwards through the network. α-β-CROWN provides a lot of efficient support for these hard tasks. However, it requires a lot of complex parameter tuning for its bound relaxation, BaB splitting, attack iteration, and about 200+ others. A poorly tuned configuration can cause the performance of α-β-CROWN to dip reducing all of its benefits.

In contrast, NeuralSAT offers an out-the-box approach which takes the network and the input/output constraints representing the property to be verified. This is achieved by using Boolean abstraction of a propositional formula representing neuron activation status. It will then search for truth assignments in the input space. A general DPLL approach is taken by deciding an assignment to the boolean variables whose constraints are propagated for the satisfiability to be checked. NeuralSAT offers a lower barrier of entry which can generalize better to unseen models. This can prompt developers to utilize these tools.

One major limitation of the current verifiers is their limited network architecture support. So far, fully connected (FC), convolutional (Conv), pooling, residual connections, and batch normalization are the only supported layer types. This is because these layers differ in the way that they learn their parameters. FC layers use neurons with weighted edges and the activation functions to feed information forward. Convolution layers performs a convolution with a learnable filter which isn't a direct edge. As well, the activation functions act as another cosntraint to what network can be supported. Most DNN have historically used a mix of ReLUs and non-piecewise linear functions. Verifiers first only supported ReLUs due to non-ReLU's bound computation complexity making the verification too hard. Since then sigmoid, tanh, and other trigometric activation functions have gained support at the cost of verification time and complexity. These two parts cause a scabaility challenge that verifiers must face. The verifiers must be able to handle large volumes of data that typically exist in high dimensional input spaces. This create a large search and verify problem. Additionally, as more modern model components are added that take longer to train, the verification process becomes more complex and longer. Finding good benchmarks that add new network architectures is vital to accelerating the growth of verification.

3 RELATED WORK

A wide range of DNN verification tools have been developed to check for adversarial robustness properties, each with a different methodology and specialization. However, the growing number of tools has made it increasingly difficult for ML developers to decide which tool to use for their application. The International Verification of Neural Networks Competition (VNN-COMP) was introduced in 2020 to address this problem and facilitate a fair and objective comparison of DNN verification tools. The competition achieved this by standardizing a set of diverse benchmarks networks proposed by participants and industry practitioners that would expose strengths and weaknesses in state-of-the-art DNN verifiers. Bak et al. state that the performance of these verification tools are evaluated on a set of equally weighted benchmarks, each containing instances which are each comprised of a trained neural network, a timeout, and pre- and post-conditions. The scores for verifiers were computed based on the number of adversarial instances that were correctly held, correctly violated, or incorrectly labeled [2].

Brix et al. note that benchmark networks from the past four VNN-COMPs have evolved significantly, demonstrating the growing complexity of networks and broadening range of applications. In VNN-COMP 2020, three benchmark categories were used: fully connected (FC) networks with ReLU activations based on ACAS Xu and MNIST, FC networks with sigmoid and tanh activation functions based on MNIST, and convolutional networks based on MNIST and CIFAR10. In 2021, the competition added a network with max-pooling layers based on MNIST, residual networks based on CIFAR10, and large networks with sparse matricies based on database indexing. VNN-COMP 2022 saw the addition of FC networks with ReLU activations on reinforcement tasks, FC network in TILL format, complex U-Net networks with average-pooling ad softmax based on image segmentation as well as additional benchmarks for convolutional networks and residual networks. From having five simple FC and convolution network benchmarks in 2020 to having 12 diverse and complex benchmarks with up to 140 million parameters in 2022, the first three iterations of VNN-COMP showed significant development in benchmark selection [6]. Furthermore, VNN-COMP 2023 introduced a generative network and benchmarks with vision transformers and batch normalization layers. The applications of benchmarks put forth in VNN-COMP have also broadened. At its inception, most benchmarks had applications in image classification, but the 2024 benchmarks comprise of diverse applications from image generation to vision to power control [5]. Brix et al. state the challenge of creating good benchmarks: they must pushed the bounds of verification tools with novel network architectures or layers, and they must also be easy enough to be solved by all competing verification tools but not so difficult that no tool can solve them [6].

During the development of α-β-CROWN, the authors have focused on an evolving set of benchmarks. In the first iteration of this verifier in 2018, CROWN, Zhang et al. evaluate the tool on multi-layer perceptron (MLP) models trained on MNIST and CIFAR10 datasets using activation functions including ReLU, tanh, sigmoid, and arctan [27]. When Xu et al. created α-CROWN in 2021, an optimized version with improved intermediate layer bounds and final layer bounds, the tool was evaluated on the CIFAR10 dataset with three neural networks: Base, Wide, and Deep [24]. In 2021, Wang et al. developed β-CROWN which integrated ReLU split constraints in branch and bound into the original CROWN bound propagation procedure, testing it on the same benchmarks as Xu et al [20]. In 2022, Zhang et al. introduced another variant, GCP-CROWN, which increased the efficiency of bound propagation by using general cutting plane methods to strengthen bound tightness [25]. GCP-CROWN was largely evaluated on the VNN-COMP 2021 benchmarks, which included FC networks, ResNet, and Convolutional networks [2].

A similar selection of benchmarks was used during the development of NeuralSAT. In 2023, Duong et al. adapted the DPLL(T) algorithm in modern SMT solvers to develop NeuralSAT, which was tested on four benchmarks from VNN-COMP 2022: ACAS Xu, MNISTFC, CIFAR2020, and RESNET_A/B [7]. NeuralSAT was also tested on another benchmark, CIFAR_GDVB, which utilized the systematic DNN verification benchmark generator GDVB [22] to create 45 different DNNs using a single CIFAR network as a seed network. GDVB systematically varies the network architecture such as the number of layers and neurons per layer to generate new variants [7]. In 2024, Duong et al. made improvements to NeuralSAT by leveraging the linear behavior of neurons at intermediates states during verification computations, reducing combinatorial complexity. This tool was experimented on the same four benchmarks from VNN-COMP 2022: ACAS Xu, MNISTFC, CIFAR2020, and RESNET_A/B. In addition, the GDVB was used again to generate 38 networks from a single MNIST network with 3 layers, each with 1024 neurons [8].

As illustrated by Brix et al., the diverse benchmarks used in VNN-COMPs reflect the wide range of DNN architectures and applications. Also, allowing participating tool authors and industry practitioners to submit networks keeps the pool of benchmarks recent and relevant to safety critical applications [6]. More so than

a set of networks, Bak et al. mentions that these benchmarks standardizes the evaluation of state-of-the-art DNN verification tools [2]. This standardization invites newer DNN verifiers utilizing novel techniques to experiment on the same benchmarks that are used to evaluate state-of-the-art verifiers. As a result, the development of new and leading DNN verification tools are significantly based off of benchmarks from VNN-COMPs as seen in the development of α-β-CROWN and NeuralSAT by Zhang et al. and Duong et al. respectively [27] [7].

4 METHODOLOGY

4.1 Benchmarks

We aimed to include a variety of benchmarks that stress the verifiers and contained different network properties. Our selection criteria followed as such

- Recent models. We aimed to include models recently developed to track the progress of what the verifiers can support and show results.
- The network architecture needs to be supported by the verifiers. All layers and activation functions included need to be implemented by both verifiers. In order for the verifier to produce results for the best trained model, we didn't want to delete any layers. We decided that tweaking the network architecture in any way would invalidate the benchmark as .
- Multiple network architectures and properties are represented. We wish to include a different number of hidden layers, different layer types, different activation functions, and how the layers and activations are connected. The benchmarks that we selected give at least two networks of comparison for each network change.

A general overview of each benchmark is included in Table 1. This includes the number of hidden layers, the activation functions that are used, the various layer types, and the pretrained accuracy for each benchmark.

4.1.1 MNISTx2. [3]

Motivation: The MNIST dataset, while seemingly simple, poses a significant challenge for verifying neural networks. Its simplicity, consisting of 28x28 pixel black and white images of handwritten digits, makes it an ideal starting point for evaluating basic neural network architectures and the verifiers' abilities to correctly verify images. Our chosen architecture, comprising just 2 fully connected hidden layers with ReLU activation functions, serves as the baseline benchmark for our study. Both of our verifiers were easily able to verify and falsify instances for this model in a minimal amount of time.

This dataset serves as a good benchmark due to the inclusion of a wide range of writing styles, sizes, and shapes causing variability in the digits. This variability impacts the model's ability to generalize well to unseen data. The range of writing styles may also result in ambiguity of the digit, where a messy '7' may resemble a '1'. This ambiguity is an important aspect of learning robust features for the model which the verifiers are able to test in more detail. As well, the noise and distortion in the images can mimic real life conditions in which the model is placed. Lighting, shadows, or any imperfections from the image need to be handled by the model. Verifiers can help identify instances where imperfections lead to incorrect predictions.

Overall, these variations and noise factors stand as a good test for the robustness of models. These challenges provide benefit to our verification technique as we are focused on adding artificial noise to these images. The performance and ability to correctly verify and unverify instances can provide insight into improving the robustness of the network architecture. Verifying the neural network before deploying it can ensure that we are receiving correct predictions in light of any possible imperfections. As well, the model won't be susceptible to adversarial attacks from attackers.

Network: Takes in a (28x28) black and white image of a handwritten digit that is flatten to a 784 pixel tensor. The values are passed through 2 fully connected hidden layers with ReLU activations connecting them. An output in range of 0-9 predicting the digit is given.

4.1.2 MNISTx6. [3]

Motivation: Our first point of comparison is **how the number of hidden layers in a model affects the verification process**. Typically, increasing the hidden layer amount leads to more representation, better generalization, and increased expressiveness. These can aid the model when more distinct decision boundaries are required or can increase performance on unseen data. This network change can help alleviate some of the challenges that MNISTx2 faces. The robustness of the model can increase and more reliable predictions are made. However, this causes the verifying task to become harder.

This increased hidden layer network serves as analysis into how this can cause the branch and bound methods to take a lot more time to cover all of the neurons in the network. The time to verify or falsify an instance is shown to takes longer for both verifier. The verifying task becomes harder to cover more layers and neurons illustrating a potential drawback of BaB methods.

Network: Similar to MNISTx2. Takes in a (28x28) black and white image of a handwritten digit that is flatten to a 784 pixel tensor. The values are passed through 6 fully connected hidden layers with ReLU activations connecting them. An output in range of 0-9 predicting the digit is given.

4.1.3 Eran for MNIST. [17]

Motivation: The two previous MNIST benchmarks focused on ReLU activation functions. Those are piecewise linear activation functions whereas some more modern network architectures include non-piecewise linear functions. This includes Sigmoid, Tanh, or other trigonometric related functions. This acts as our second point of comparison, **how the activation function will affect the verification process**. ReLU offer computationally simple process but are unbounded. Non-piecewise functions can enhance the smoothness of optimization functions.

Network: This benchmark builds off of the previous 6 hidden layer MNIST network. It also takes in a (28x28) black and white image of a handwritten digit that is flatten to a 784 pixel tensor. The image is given to the 6 fully connected hidden layers with a width of 200 neurons. Now the model will use the Sigmoid[1] activation

[1]Any non-piecewise activation function may be used

	MNISTx2	MNIST x6	ERAN	CIFAR	GTSRB
Number of Hidden Layers	2	6	6	14	13
Activation Function(s)	ReLU	ReLU	Sigmoid	ReLU	Sign + SoftMax
Layer Type(s)	FC	FC	FC	ResNet	QConv + FC
Pretrained Accuracy	98.0%	96.8%	97.3%	69.2%	81.5%
Input Dimension	28x28	28x28	28x28	32x32	30x30
Output Dimension	10	10	10	10	43

Table 1: Overview of All Benchmarks

function between the layers. The MNIST digit prediction is given as output.

4.1.4 CIFAR10 ResNet. [21]

Motivation: All of the previous benchmarks were just fully connected layers with various adjustments made to the number of layers and activation functions. However, verifiers should be able to handle more generic architectures, such as ResNet, which exist in more practical prediction scenarios. ResNet indicates the first complex network that we analyze and is the third point of comparison, **how more complex layer types affect the verification process**. ResNet is a step to more modern models and the techniques that make image classification a more efficient process with increased performance.

The convolutional layers introduce a new form of information learning through their convolution filters. Verifiers need to handle different forms of parameter learning introduced in task specific layer types. This will enhance the verifier generalizability to new layers so that we do not lag behind model development.

Network: The ResNet structure with 2 residual blocks was used. The model takes in a (32x32) RGB image of an object. These pixels are passed through 5 convolutional layers and 2 fully connected layers with ReLU activation functions between residual blocks. An index of the predicted object is given.

4.1.5 German Traffic Sign Recognition Benchmark (GTSRB). [16]

Motivation: The German Traffic Sign Recognition Benchmark plays a crucial role in extending the verification process to safety critical application domains. Traffic signs are necessary for ensuring road safety and managing traffic flow. The recognition of these signs are vital for recent developments in autonomous driving vehicles that make decisions based on the signs that are recognized. For example, interpreting a stop sign as a speed limit sign can cause major harm when the person is unaware of their surroundings.

This benchmark and the challenges showcase the importance of verifying the model predictions in real time. Driving occurs at all points of the day where lighting, weather impacting the sensors, and regular image noise can impact the prediction. These models need to tuned across various regions and also verified to ensure safety. The safety critical domain makes this verification process even more important.

Network: Figure 4 outlines the more complex model architecture that was trained using the GTSRB. A Binary neural network (BNN) was used due to promise in computional and power consumption efficiency for resource constrained systems. QConvolutional layers, which binarize the convolutional layers, are used with the Sign activation function connnecting them. After 2 convolutions, a fully

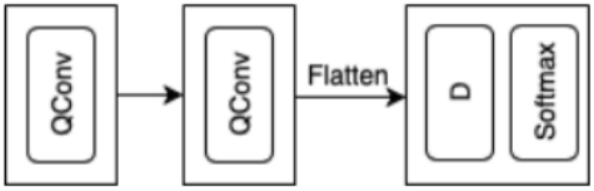

Figure 4: Model architecture for GTRSB dataset

connected and softmax layer are used to output the identified traffic sign. A (30x30) image is used with a range of 43 output classes of traffic signs. The model was trained and test on the 43 classes which only include the German signs. The other Belgian or Chinese traffic sign datasets were not included in the model that we tested and verified.

4.2 Generating Robustness Properties

In this work, we focused exclusively on local robustness properties. These robustness properties are defined as perturbations to each pixel in an image. Images are usually 2d sets of 3 RGB values which are set within [0, 255]. (The MNIST dataset images are only these 2d sets of one black and white pixel scaled between [0, 255]. The CIFAR and GTSRB are RGB so this is represented as a 3-dimensional array of pixels.). Part of preprocessing the data includes normalizing these pixels by uniformly scaling the image down so that the pixel are represented as floats from [0, 1].

From here, we can define a ϵ value which indicates the ϵ ball radius for that pixel at which it will be verified. The pixel is then given a lower and upper bound by simply subtracting and adding ϵ to the pixel, respectively. We clip the pixel to [0, 1] in order to ensure that verification stays in range. Now that we have the lower and upper bound defined, the constraint is given for that pixel such that we do not exceed either bound. This process is repeated for all pixels within an image to define a set of constraints over all pixels. In addition to these pixel constraints, the output constraints are defined from the correct label. The instance should not exist with a different label inside of the ϵ space. All constraints are put in the standard vnnlib format which can then be given to the verifier to determine if the instance is sat/unsat.

4.3 Sweeping-ϵ

The main method that we used to identify verifier abilities for a given benchmark was a sweeping-ϵ approach. As depicted in

Algorithm 1, we choose I number of images chosen from the benchmark's dataset that are correctly predicted using the Onnx model O. For all I images, a new instance i is created with a ε value. These ε are uniformly spread out from the starting ε to the ending ε for a determined ε count. For each ε_j we'll create a new instance of image i_j that uses the property as described in Section 4.2. Now that all properties are generated, the verifier can be run on instance i_j. The verifier would either return unsat (the instance was verified), sat (the instance was falsified or considered an adversarial example), or time out (the instance was being verified for too long). This time out was considered an unverified instance for our method because that means the problem was too hard to verify. A time out does not explicitly mean that the instance is an adversarial example but the instance became too hard.

Data: Onnx Model O, Verifier, Timeout T
Generate I number of correctly predicted images using O
for $i \in Instances$ **do**
  for $\varepsilon \in Uniform(\varepsilon_{start}, \varepsilon_{end})$ **do**
    Generate local robustness property p from ε
    Run verifier on property p using O for max of T seconds
  end
end

Algorithm 1: Conducting Sweeping-ε Overview

4.4 Metrics

We define three new metrics to determine the effectiveness of the verifier for a associated dataset.

(1) ε-Intersection: This occurs at the ε value which the number of verified instances equals that of unverified instances. As timeouts are considered unverified, this point indicates the level of perturbation at which the verifying problem is becoming too hard for the verifiers.
(2) Time Inflection: The time inflection is when our average time per instance inflects and begins to decrease. This is typically reached when all of the instances are timed out or unverified. This illustrates that the problems are becoming easily unverified due to the excessive noise.
(3) Unverified Rate of Change: This value is defined as the rate of change of the decline from mostly all verified to mostly all unverified instances. This value goes together with our ε-intersection and gives information to the user if a network has a hard decision boundary. A high rate of change tells us that just adding a little more noise will cause the instances to more easily be unverified.

Each benchmark generates its own unique set of properties that were fed to both verifiers. As the benchmarks varied in their complexity, we focused on showcasing the key metrics for each benchmark. Each benchmark was given a timeout that is lower than those used by the VNNCOMP. We decided these timeouts were necessary due to the computational complexity that these verifiers had at some ε values. We found this to be a necessary trade-off to analyze more instances per ε. Further, we wanted the ε-intersection to not exist at one extreme, for both the unverified and verified number of instances to plateau to all instances and no instances, respectively, and the time inflection point to exist. Thus, we set up each benchmark with different experimental settings. They mostly influence our range of ε which is seen in the Results section and analyzed in the Discussion section. These are outlined in Table 2.

5 RESULTS

5.1 MNIST

5.1.1 MNISTx2. Figure 5 shows α-β-CROWN verifying MNIST classification with a 2 hidden layers FFC + ReLU using a total of 40 instances. Figure 6 shows NeuralSAT verifying the same benchmark. The ε-intersection of both α-β-CROWN and NeuralSAT occurs between ε=0.0267 and ε=0.0356 (increments of ε were tested so unless the number of verified instances equals the number of unverified instances at one of the ε values we test, the exact ε-intersection cannot be determined and an ε range is given instead). This indicates that both verifiers found similar levels for perturbation before instances would be considered unverified. However, the verification time for each verifier vary. α-β-CROWN took a maximum time of 1.2s to verify all instances while NeuralSAT took more than 2s on most instances and took a maximum of 9s. In addition, the verification time for α-β-CROWN peaks near the ε-intersection while the verification time for NeuralSAT peaks after the ε-intersection when it handles more unverified instances. It is also important to note the time scale of the verification time for α-β-CROWN since its time plot takes a seemingly uncharacteristic sharp dip near the ε-intersection. Since the verification times of this graph are all quite low, small differences are magnified, so this dip may not be significant.

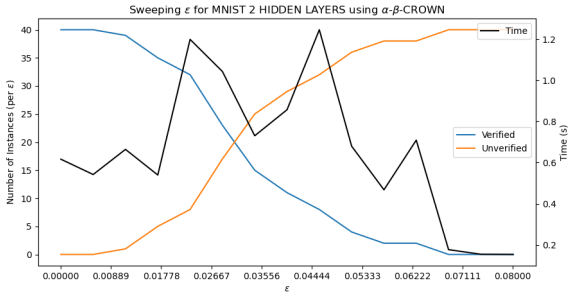

Figure 5: α-β-CROWN on MNISTx2 (MNIST Classification with 2 Hidden Layer FFC + ReLU)

5.1.2 MNISTx6. Figure 7 shows α-β-CROWN verifying MNIST classification with a 6 hidden layers FFC + ReLU using a total of 40 instances. Figure 8 shows NeuralSAT verifying the same benchmark. Both verifiers perform similarly on this benchmark as they share the same ε-intersection, ε=0.0267, demonstrating that both tools found the same level of perturbation for the instances before they became unverified. The verification times of both verifiers are similar, with both verifiers' times steadily increasing at $\varepsilon = 0.013$ when the numbered of unverified instances started to increases, peaking at around 80s after the ε-intersection, and then making a gradual decline as more unverified instances are handled.

	MNISTx2	MNIST x6	ERAN	CIFAR	GTSRB
Number of Instances	40	40	50	50	40
Starting ϵ	$\frac{.01}{255}$	$\frac{.01}{255}$	$\frac{.01}{255}$	$\frac{.01}{255}$	$\frac{.01}{255}$
Ending ϵ	$.08 + \frac{.01}{255}$	$.08 + \frac{.01}{255}$	$.04 + \frac{.01}{255}$	$.025 + \frac{.01}{255}$	$.008 + \frac{.01}{255}$
ϵ count	15	10	10	10	10
Timeout (seconds)	120	120	180	120	240

Table 2: Benchmark Experimentation Settings

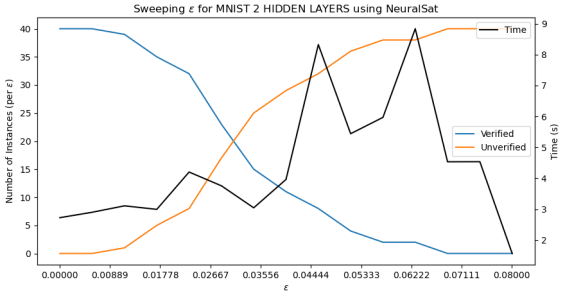

Figure 6: NeuralSAT on MNISTx2 (MNIST Classification with 2 Hidden Layer FFC + ReLU)

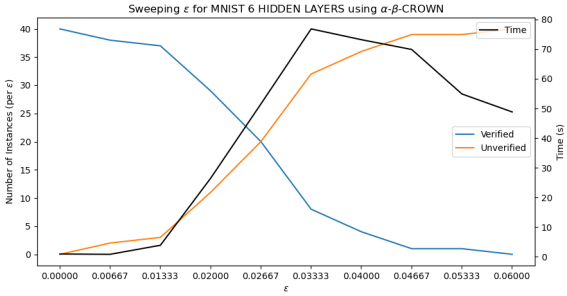

Figure 7: α-β-CROWN on MNISTx6 (MNIST Classification with 6 Hidden Layer FFC + ReLU)

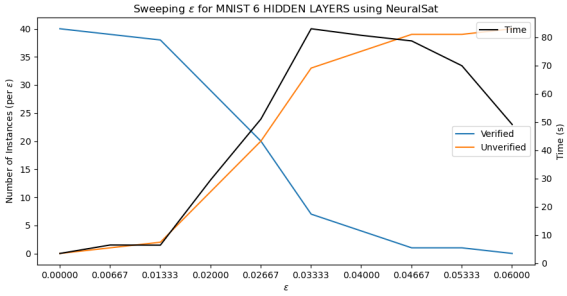

Figure 8: NeuralSAT on MNISTx6 (MNIST Classification with 6 Hidden Layer FFC + ReLU)

5.1.3 ERAN. Figure 9 shows α-β-CROWN verifying MNIST classification with a 6 hidden layers FFC + Sigmoid using a total of 50 instances. Figure 10 shows NeuralSAT verifying the same benchmark. Both tools perform similarly on this benchmark as the ϵ-intersection occurs between ϵ=0.0133 and ϵ=0.0178. The verification times are similar for both verifiers as the shape of the time plot matches the shape of the unverified instances plot. At ϵ values where there are more unverified instances, a majority of the computation time is at the specified 180s timeout, indicating that many of these instances were classified as unverified due to the timeout on both verifiers.

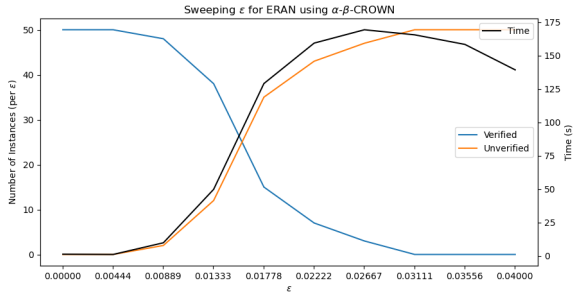

Figure 9: α-β-CROWN on ERAN (MNIST Classification with 6 Hidden Layer FFC + Sigmoid)

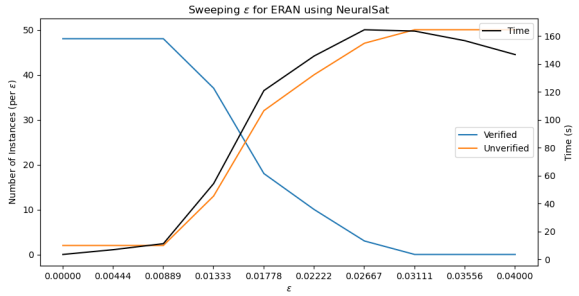

Figure 10: NeuralSAT on ERAN (MNIST Classification with 6 Hidden Layer FFC + Sigmoid)

5.2 CIFAR

Figure 11 shows the α-β-CROWN verification of the CIFAR10 dataset using the ResNet2b structure. A total of 50 instances were used. Figure 12 shows NeuralSAT verifying the same benchmark.

Both verifiers performed most equally on this benchmark, sharing a ε-intersection, $\varepsilon \approx .009$. The unverified rate of change and verification time rise remained similar between verifiers for ResNet. The ResNet architecture results differ from the previous MNIST results in that the number of unverified instances linearly increased. Before, this would be an exponential increase until all instances were unverified causing a plateau. The verifier more quickly made jumps to unverifying all the instances. We theorize that if the ε range was shrunk from 0 to .02 that a more exponential increase would be observed from both verifiers.

The observed time inflection point occurred at the time where almost all of the instances were deemed falsified. α-β-CROWN showed a larger magnitude decrease in time taken per instances after this inflection point. This illustrates that α-β-CROWN was able to falsified more of the highly perturbed images. NeuralSAT was still stuck trying to verify more of those.

Interestingly, both verifiers found 10 instances that were considered adversarial attacks at the beginning where $\varepsilon = \frac{.01}{255}$. This can account for an earlier ε-intersection point. We believe that even though the onnx model was able to correctly predict the instances when we were loading the data, the instance is extremely close to the decision boundary. It could be the case that the model was between two classifications with almost equal probabilities. Even with such a tight ε window, this caused the classification to flip to the other high probability, signifying an adversarial example. It's possible that the model was too weak and not accurate enough to ensure that verification was useful.

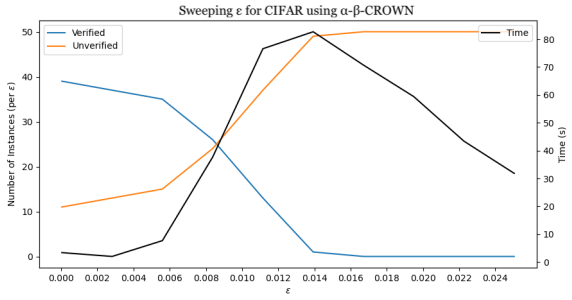

Figure 11: α-β-CROWN on CIFAR Classification with ResNet

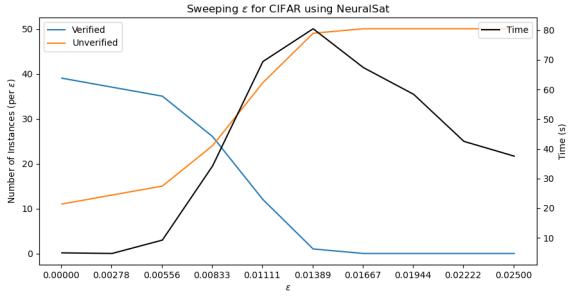

Figure 12: NeuralSAT on CIFAR Classification with ResNet

5.3 GTSRB

Figure 13 shows α-β-CROWN verifying the QConv 4 architecture with the GTRSB dataset for a total of 40 instances. The NeuralSAT verification of the same instances is shown in Figure 14. The verifier diverged in performance more in this benchmark as we let this harder verification task run for longer. α-β-CROWN showed to verify more instances in the ε's before the plateau and reached the plateau at $\varepsilon = .0044$ whereas NeuralSAT reached it at $\varepsilon = .0035$. This ultimaely came down to one more instance. Both verifiers had a similar decrease in time per instance as it reached the plateau. At the plateau, NeuralSAT was more easily to falsify adversarial examples shown by the lower time at the higher ε values.

Like the CIFAR dataset, 5 of the 40 instances were initially considered unverified within the tight starting ε. We hypothesize that this is because the architecture of model is so complex and its pretrained accuracy is already low. These two factors may cause any sort of change in the pixels to flip the bit which produces a falsified instance. The major difference seen with this dataset was that verification time was extremely high and at its peak to begin with. This showcases the difficulty in verifying these larger and more complex models at any perturbation value.

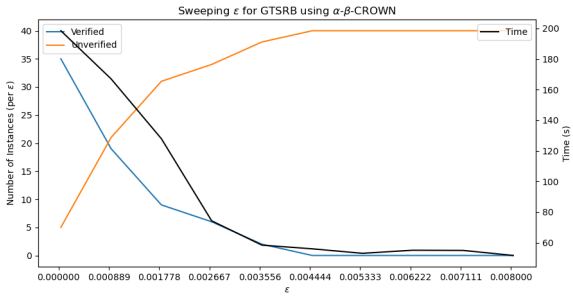

Figure 13: α-β-CROWN on GTSRB Classification with FC, ReLU, QConv, Sign

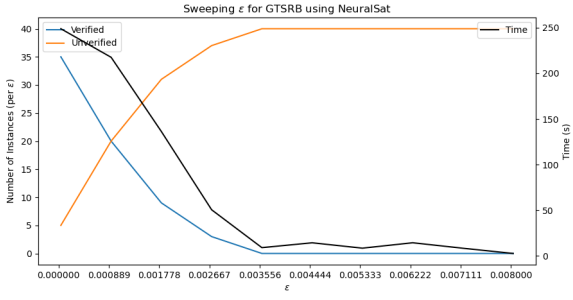

Figure 14: NeuralSAT on GTSRB Classification with FC, ReLU, QConv, Sign

6 DISCUSSION

6.1 Number of Hidden Layers

α-β-CROWN and NeuralSAT were evaluated on MNISTx2 and MNISTx6, which have 2 hidden layers and 6 hidden layers in their networks respectively. These benchmarks demonstrate how the number of hidden layers in a network affects the ε-intersection found by the verifiers, unverified rate of change, and verification time. The ε-intersection of MNISTx2 was between ε=0.0267 and ε=0.0356 and the ε-intersection of MNISTx6 was ε=0.0267. Based on these results, the ε-intersection decreases when the number of hidden layers increases. More hidden layers in a network produces more parameters for the model to train on and places the model in a more complex hypothesis class. While networks with more layers can capture more complex patterns in the data, having significantly more parameters can make complex networks more susceptible to noise than simpler networks. The increased sensitivity to noise of complex networks is seen in these results as the simplest network, MNISTx2, which is able to safely handle a larger ε degree of perturbation than MNISTx6.

The increasing number of hidden layers in MNIST based benchmarks can also be reflected in the verification times of both α-β-CROWN and NeuralSAT. MNISTx2 was much quicker to verify, with α-β-CROWN taking a maximum of 1.2s and NeuralSAT taking a maximum of 9s. MNISTx6 took longer to verify, with both verifiers taking a maximum of around 80s. These results show that the verification times for both verifiers increase significantly as the number of hidden layers in a network increases. While a more complex network with more parameters and layers may perform better, it incurs the cost of higher training time and also higher verification times due to increased computational complexity from higher dimensionality. As the network becomes more complex, constraint solvers in DNN verification tools experience an increased set of constraints and even higher-dimensional constraint spaces which make computation take longer.

In addition to ε-intersection and verification times, the unverified rate of change also differed across MNISTx2 and MNISTx6. As ε was increased, the number of verified instances on MNISTx2 steadily declined. However, on the MNISTx6 benchmark, the number of verified instances first gradually decreased and then sharply decreased. This may indicate that slight perturbations on the MNISTx2 benchmark had immediate effects on the safety of the result and larger perturbations had similar effects. On the other hand, slight perturbations on the MNISTx6 benchmark has less impact but larger perturbations had a more drastic impact on instance. This may be due to the fact that more complex networks are robust to minimal noise but after a ε threshold are significantly less robust than a simpler network at the same ε threshold.

6.2 Activation Functions

The MNISTx6 and ERAN benchmarks both contain 6 hidden layers but use the ReLU activation function and Sigmoid activation function respectively. The difference in activation function can be demonstrated in the ε-intersection, verification time and unverified rate of change. The ε-intersection of MNISTx6 was ε=0.0267 while the ε-intersection of ERAN was between ε=0.0133 and ε=0.0178. The ERAN network was able to handle a lesser degree of perturbation than the MNISTx6 network, which indicates that the Sigmoid activation function may be more susceptible to adversarial attacks than the ReLU activation function. Since the Sigmoid activation function, which is a non-piecewise function involving exponentiation and division operations, is more complex than the ReLU activation function, which is a piecewise linear function, networks that consists of Sigmoid activation functions may be more sensitive to added noise in inputs. This increase in sensitivity to perturbed instances can be demonstrated in the change in ε-intersection.

The change in activation function between the MNISTx6 and ERAN benchmarks is most apparent in the verification times for both α-β-CROWN and NeuralSAT. The verification time for MNISTx6 peaked at around 80s for both verifiers while the verifiers reached the 180s timeout on most unverified instances in the ERAN benchmark. The significant increase in verification time can be attributed to the increased computational complexity of the Sigmoid over the ReLU. The Sigmoid activation function's non-linearity makes constraint solving more computationally expensive than the piecewise linear activation function that is the ReLU, which can be seen in the significant increase in verification time for both verification tools. In addition, it is worth noting that many instances in the ERAN benchmark were deemed unverified due to timing out. A larger timeout may have allowed more instances to be verified by the verifiers, which would have also impacted the ε-intersection for this benchmark.

The difference in unverified rate of change between MNISTx6 and ERAN can also be attributed to the increased complexity of the Sigmoid over the ReLU. All instances in the MNISTx6 benchmark were unverified at ε=0.06 while all instances in the ERAN benchmark were unverified at ε=0.04. The unverified rate of change was higher in the ERAN benchmark as more instances got classified as unverified as the ε gradually increased, which reinforces the idea that the ERAN network was more sensitive to perturbation due to the inclusion of the Sigmoid activation function.

6.3 Layer Types

The latter 2 benchmarks introduced, ResNet2b for CIFAR10 and GTSRB, are best defined by their increasing complexity and application to harder problems. These benchmarks introduced new layer types that were previously restricted to fully connected layers. CIFAR introduced a basic convolution layer. The ε-intersection found for CIFAR was $\varepsilon \approx .009$ which signifies a decrease from both the normal fully connected layer type and inclusion of the sigmoid activation function. The convolutional layer is learnt in a different way where it attempts to learn spatially invariant aspects. This poses a more difficult verification process. As well, lighting and noise can affect the robustness of these models by altering those invariant spaces to a slightly different pixel value. If representative data is not included in the training process then the verifier will have a harder time verifying and falsifying instances. This leads to the increased time that we see in our results. Overall, the increased noise affects the ResNet model more than any of the previous models due to the increased complexity of convolution layers.

GTSRB was a further expansion on convolutional networks while using other activation functions. This benchmark established the

worst ε-intersection by far, $\varepsilon = .0008$, a 10x decrease from ResNet. As well, there was not a true time inflection point since the problems started out as hard to verify and only got easier to falsify. The GTSRB is the only benchmark which the time inflection point comes before the ε-intersection point. This fact demonstrates how complex architectures with different layer values can become very hard to verify.

This can also be related to GTSRB being the only benchmark to have more than 10 output classes. More decision boundaries are inherited by this model which increases the complexity to verify the layer types. This combined with complex layer types causes the problem to become harder to verify. However, progress towards these benchmarks are considered the most important for advancing verifiers and model development. It is crucial that we established effective guardrails to these models so we can ensure against adversarial attacks. As well, it is necessary to support more common layer types, which is the goal of including the ResNet and GTSRB benchmarks. If a generalized verification approach can be reached, we can verifying more diverse models and ensure every safety critical application is secured with a stark boundary.

6.4 α-β-CROWN and NeuralSAT

Overall, α-β-CROWN and NeuralSAT compared closely for all the chosen benchmarks. We hypothesize that if given more time, α-β-CROWN should verify more instances than the NeuralSAT counterpart. Seen through the ERAN and CIFAR benchmarks, α-β-CROWN can unverify more instances at the higher ε values before exceeding the timeout because verification takes on average a lower time per ε. If the timeout is removed, we expect a trend seen in the MNISTx2 benchmark to continue. This is where the time taken to execute all instances reaches an absolute peak. This peak may exist completely past the boundary of feasibility for execution time.

7 LIMITATIONS AND FUTURE WORK

This work was limited to the experimental settings that we chose. Ideally, more randomized instances and more ε values in our defined range would have been used. The general trends of these benchmarks would have still been seen, but the performance between verifiers could show more discrepancies with a more tuned approach. We were ultimately limited by computational power and time.

Additionally, the choice of benchmarks was focused mainly on image classification. More benchmarks exploring other application domains, especially more safety critical ones, can provide more insight into how these verifiers can integrate into the development work flow. Safety critical applications would benefit the most to ensure that our prediction is correct. A study into non-urgent medical imaging could be done where the model makes a prediction, and the verifier can run for an extensive amount of time to verify this. In general, more benchmarks can be added to this effort to further understand what benefits each verifier holds.

Lastly, we used α-β-CROWN and NeuralSAT as our two main verifiers to test the benchmarks on. However, other verifiers bring their own pros and cons and can be better at verifying certain tasks or checking different properties other than robustness. Further work would be to include other state of the art performing verifiers such as MN-BaB [9], Marabou [13], or Fairify [4].

8 CONCLUSION

We present a review of deep neural network verification using two state of the art verifiers, α-β-CROWN and NeuralSAT. We focused on the local robustness property which ensures that a model can maintain a correct prediction despite added noise or environmental change to the input. In conjunction, adversarial examples should be detected by the verifier to ensure that intentional and unintentional attacks are identified. We analyzed these abilities of the verifiers using set of benchmarks that are representative of various DNN architectures. This includes comparing how the number of hidden layers, activation functions used, and complex layer types can impact and alter the verification process. These benchmarks used the MNIST, CIFAR, and GTSRB datasets with pretrained models to generate robustness properties. These were defined by a ε value which determines how much each pixel is perturbed and bounded by. All pixel bound constraints with the same output constraints were fed to the verifier to verify or falsify.

The model complexity was increased through increased number of hidden layers, non-ReLU/non-piecewise linear activation functions, and non-fully connected layers. It was found that this increased complexity resulted in a longer verification time and more instances to not be verified at lower ε values. The MNISTx2 and MNISTx6 demonstrated that the verifier can easily verify small network neuron spaces. The work is divided on the GPUs and took less than 10 seconds for the hardest instance to verify. However, with more hidden layers and the same instances, it became a lot harder to verify seen by a distinct time inflection point of more than 80 seconds at about the same ε. The ERAN benchmark continued this trend by using the Sigmoid activation function on top of the 6 hidden layers. The non linear nature of the Sigmoid brought a longer verification time while pushing the ε-intersection from .0267 to .0177. This indicated that non-linearity causes less instances to be verified within the alloted time. The CIFAR and GTSRB benchmarks confirmed our findings that increased complexity makes verification extremely hard. The ResNet architecture introduced convolution layers which caused a low ε-intersection with the GTSRB network being even lower with a sharp decline in ability to verify instances.

We concertized the verification of deep neural network through the use of ε-intersection, time inflection, and the verified/unverified rate of change. We aim to increase verification when developing and deploying models by offering metrics to ensure robustness and guiding model improvements.

9 SUPPLEMENTARY MATERIAL

The artifact for this project can be found at https://github.com/sgsikorski/DNN_Analysis/. All plots presented can be found in /results using a standard sweeping_eps_BENCHMARK_INSTANCES_VERIFIER format. These were plotted with plot.py. The local robustness properties were generated according to generateProperties.py and wrote them to a standard vnnlib file which lived in a folder that housed the onnx file. All α-β-CROWN instances were run with a given configuration

file found in /configs. All NeuralSAT instances were run with a script found in /run_ns_instances

REFERENCES

[1] Vishal Dey Anupam Chattopadhyay Debdeep Mukhopadhyay Anirban Chakraborty, Manaar Alam. 2021. A survey on adversarial attacks and defences. In *CAAI Transactionson Intelligence Technology*, Vol. 6. 25–45. https://doi.org/10.1049/cit2.12028

[2] Stanley Bak, Changliu Liu, and Taylor Johnson. 2021. The Second International Verification of Neural Networks Competition (VNN-COMP 2021): Summary and Results. arXiv:2109.00498 [cs.LO]

[3] Alejandro Baldominos, Yago Saez, and Pedro Isasi. 2019. A survey of handwritten character recognition with mnist and emnist. *Applied Sciences* 9, 15 (2019), 3169.

[4] Sumon Biswas and Hridesh Rajan. 2023. Fairify: Fairness Verification of Neural Networks. In *ICSE'23: The 45th International Conference on Software Engineering* (Melbourne, Australia).

[5] Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. 2023. The Fourth International Verification of Neural Networks Competition (VNN-COMP 2023): Summary and Results. arXiv:2312.16760 [cs.LG]

[6] Christopher Brix, Mark Niklas Müller, Stanley Bak, Taylor T. Johnson, and Changliu Liu. 2023. First Three Years of the International Verification of Neural Networks Competition (VNN-COMP). arXiv:2301.05815 [cs.LG]

[7] Hai Duong, ThanhVu Nguyen, and Matthew Dwyer. 2024. A DPLL(T) Framework for Verifying Deep Neural Networks. arXiv:2307.10266 [cs.LG]

[8] Hai Duong, Dong Xu, ThanhVu Nguyen, and Matthew B. Dwyer. 2024. Harnessing Neuron Stability to Improve DNN Verification. arXiv:2401.14412 [cs.LG]

[9] Claudio Ferrari, Mark Niklas Muller, Nikola Jovanovic, and Martin Vechev. 2022. Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound. arXiv:2205.00263 [cs.LG]

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572 [stat.ML]

[11] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review* 37 (2020), 100270. https://doi.org/10.1016/j.cosrev.2020.100270

[12] T. Brox J. H. Metzen, M. C. Kumar and V. Fischer. 2017. Universal Adversarial Perturbations Against Semantic Image Segmentation. , 2755-2764 pages.

[13] Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. 2019. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*. Springer, 443–452.

[14] Suhas Kotha, Christopher Brix, J. Zico Kolter, Krishnamurthy Dvijotham, and Huan Zhang. 2023. Provably Bounding Neural Network Preimages. In *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 80270–80290. https://proceedings.neurips.cc/paper_files/paper/2023/file/fe061ec0ae03c5cf5b5323a2b9121bfd-Paper-Conference.pdf

[15] Klaus Leino. [n. d.]. Training Provably-Robust Neural Networks. https://towardsdatascience.com/training-provably-robust-neural-networks-1e15f2d80be2

[16] Andreea Postovan and Mădălina Eraşcu. 2023. Architecturing Binarized Neural Networks for Traffic Sign Recognition. *arXiv preprint arXiv:2303.15005* (2023).

[17] Anian Ruoss, Maximilian Baader, Mislav Balunović, and Martin Vechev. 2021. Efficient Certification of Spatial Robustness. arXiv:2009.09318 [cs.LG]

[18] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. 2019. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks. *Advances in Neural Information Processing Systems* 32 (2019), 9835–9846.

[19] Zhouxing Shi, Qirui Jin, J Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. 2023. Formal Verification for Neural Networks with General Nonlinearities via Branch-and-Bound. *2nd Workshop on Formal Verification of Machine Learning (WFVML 2023)* (2023).

[20] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. 2021. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in Neural Information Processing Systems* 34 (2021).

[21] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. 2021. Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Complete and Incomplete Neural Network Verification. *arXiv preprint arXiv:2103.06624* (2021).

[22] Dong Xu, David Shriver, Matthew B. Dwyer, and Sebastian Elbaum. 2020. Systematic Generation of Diverse Benchmarks for DNN Verification. In *Computer Aided Verification*, Shuvendu K. Lahiri and Chao Wang (Eds.). Springer International Publishing, Cham, 97–121.

[23] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. 2020. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems* 33 (2020).

[24] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. 2021. Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers. In *International Conference on Learning Representations*. https://openreview.net/forum?id=nVZtXBI6LNn

[25] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. 2022. General Cutting Planes for Bound-Propagation-Based Neural Network Verification. *Advances in Neural Information Processing Systems* (2022).

[26] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. 2022. A Branch and Bound Framework for Stronger Adversarial Attacks of ReLU Networks. In *Proceedings of the 39th International Conference on Machine Learning*, Vol. 162. 26591–26604.

[27] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 2018. Efficient Neural Network Robustness Certification with General Activation Functions. *Advances in Neural Information Processing Systems* 31 (2018), 4939–4948. https://arxiv.org/pdf/1811.00866.pdf

[28] Jin Zhang and Jingyue Li. 2020. Testing and verification of neural-network-based safety-critical control software: A systematic literature review. *Information and Software Technology* 123 (2020), 106296. https://doi.org/10.1016/j.infsof.2020.106296

[29] Xingyu Zhao, Alec Banks, James Sharp, Valentin Robu, David Flynn, Michael Fisher, and Xiaowei Huang. 2020. A Safety Framework for Critical Systems Utilising Deep Neural Networks. In *Computer Safety, Reliability, and Security*, António Casimiro, Frank Ortmeier, Friedemann Bitsch, and Pedro Ferreira (Eds.). Springer International Publishing, Cham, 244–259.