False Positives in Robustness Checking of Neural Networks

Mohammad Afzal[1,2], Ashutosh Gupta[1], R. Venkatesh[2] and S. Akshay[1]

[1]Indian Institute of Technology Bombay, Mumbai, India
[2]TCS Research, Pune, India

Abstract

Neural networks are increasingly used in safety-critical systems. To ensure their trustworthy deployment, we need to verify that they are robust against minor perturbations. But in doing so, the counterexamples found by state-of-the-art neural network verifiers for robustness may not be really meaningful. In fact, a counterexample that is considered a robustness violation reported by a neural network may not be a violation in the view of a domain expert. We refer to such cases as false positives (FP). In this work, we propose a new approach to evaluate the robustness of neural networks by considering the view of domain expert. Our goal is to evaluate the presence or absence of such FP in state of the art verifiers. However, doing this manually indeed may not scale. Thus, in our experiments, we evaluate the local robustness property based on the notions of FP and TP (True Positives), while approximating the domain expert using an ensemble of state-of-the-art neural network classifiers.

Keywords

Neural Network Verification, False Positive, Local Robustness Verification, Formal Methods

1. Introduction

Neural networks are increasingly used in safety-critical applications such as flight control [1], autonomous vehicles [2, 3, 4, 5], and medical diagnosis [6]. In the domain of computer vision, an adversarial example is an image that is visually similar to a given input image but is classified differently by the network. The work by Goodfellow et al. [7] demonstrated the existence of the adversarial examples in classical neural networks by slightly modifying an image in an algorithmic manner. This raises serious safety concerns for the use neural networks and has led to a vast research endeavor towards finding such adversarial examples. The machine learning community [8, 9, 10, 11, 12, 13, 14] has actively worked on finding such adversarial examples. Some of the well-known methods are fast gradient sign method (FGSM) [7] and projected gradient descent (PGD) [15]. However, these efforts typically fall short of verifying the absence of such examples.

One approach towards addressing this issue has been to use formal verification techniques. By leveraging formal methods, researchers provide certification of the absence of adversarial examples, if none exist [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]. These methods use abstract interpretation [32] or constraint solving [33] to analyze the neural network. Many of the works in this setting [34, 35, 36] focus on the local robustness verification problem, i.e., for a given neural network and an input image, the network is considered locally robust if all images close to the given image are classified the same as the original, where distance is defined in terms of a simple distance metric (e.g., L_1, L_2, or L_∞ norms). In the verification community, such images, which close to the original, but are classified differently are more often called counterexamples, and existing work focuses on either finding counterexamples or providing certification of their absence.

However, in doing so, existing works often lose the focus on evaluating the *genuineness* of the counterexample. This leads to false positives, i.e., counterexamples that exist from the perspective of the underlying specification, but may not be considered genuine from a domain expert's perspective. In

OVERLAY 2025, 7th International Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, October 26th, 2025, Bologna, Italy
afzal.2@tcs.com (M. Afzal); akg@cse.iitb.ac.in (A. Gupta); akshayss@cse.iitb.ac.in (S. Akshay)
https://afzalmohd.github.io/ (M. Afzal); https://www.cse.iitb.ac.in/~akg/ (A. Gupta); https://www.cse.iitb.ac.in/~akshayss/ (S. Akshay)
0000-0002-6173-3959 (M. Afzal); 0009-0003-7755-2006 (A. Gupta); 0000-0002-2471-5997 (S. Akshay)


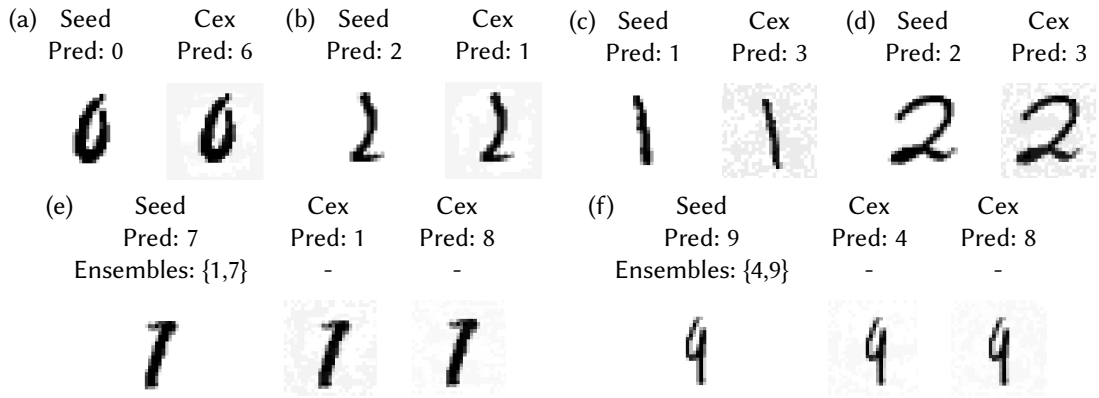

Figure 1: (a)–(b) Examples of false positives; (c)–(d) Examples of true positives. For each case, we show two images: the original image ("Seed") and the corresponding counterexample ("Cex") identified by the verifier. (e)–(f) Examples of true positives guided by ensemble classifiers. For each case, we show three images: the original image ("Seed"), the counterexample ("Cex") identified without ensemble guidance (middle), and the counterexample ("Cex Ensemble") identified with ensemble guidance (right).

this work, our aim is to identify false and true positives and study their prevalence in commonly used image classification benchmarks. Let us start by looking at a few motivating examples.

Consider the two images on the left of Figure 1(a)-(b), which are from the MNIST [37] dataset and have ground truth labels 0 and 2, respectively. The images on the right are counterexamples generated by the state-of-the-art verifier $\alpha\beta$-CROWN [38] for the image labeled 0 (misclassified as 6), and by the same verifier for the image labeled 2 (misclassified as 1). A domain expert may not consider these as actual *bugs* because the counterexample labeled as 6 also visually resembles a 6, and the network predicts it as either 0 or 6—which is reasonable. Similarly, the counterexample misclassified as 1 also resembles a 1, and the network predicts it as either 1 or 2, which is again acceptable. Since the network's predictions are visually justifiable, labeling them as counterexamples is, in fact, a false positive.

On the other hand, a counterexample may be considered a genuine counterexample, which we call "true positive", if the network misclassifies the image and the misclassification is not visually justifiable. In Figure 1(c), the image on the left is a seed image, and the image on the right is a counterexample reported by the $\alpha\beta$-CROWN verifier. The image is classified as 3 by the verifier and human expert would say that it is a 1. We present a similar example in Figure 1(d).

In this work, we recall and define the well-known notions of false positive (FP) and true positive (TP) in the context of local robustness verification of neural network image classifiers. Our modified definition requires a domain expert to classify the images and assign the ground truth labels, which we formally call an oracle. An important point to note is that in our setting, even an oracle may not always be able to classify the images since images may resemble many labels. Thus it may assign a set of possible labels to an image. As a result, we may use the oracle in two possible ways: (1) to determine whether a counterexample produced by the verifier is a true or false positive by checking whether the classification of the counterexample belongs to one of the oracle-provided labels; and (2) to guide the verifier to search only for counterexamples that are not classified as one of the oracle-provided labels.

For example, in Figure 1(e), the oracle may state that the original image could be either a 1 or a 7. We first ask the verifier to find a counterexample, and it reports one classified as 1 (middle image). We then ask the verifier to look specifically for a counterexample that is not classified as 1 or 7. If such a counterexample is found (right image), it qualifies as a true positive. Similarly, in Figure 1(f), the oracle identifies the original image as possibly 4 or 9. We ask the verifier to find a counterexample that is not classified as 4 or 9, and it reports 8—a true positive. Otherwise, it would have reported 4 as a counterexample, which would not be valid true positive.

It is an open secret that false positives exist in practice. However, we ask a more precise question: what is the rate of false positives in practice (RQ1)? To answer this, we evaluate the standard local robustness property with respect to false positives and true positives. To approximate the domain expert

(oracle), we develop an ensemble based approach that exploits multiple neural network classifiers, as we detail below.

We conduct experiments using state-of-the-art neural network verifier $\alpha\beta$-CROWN [38] on the MNIST [37] and CIFAR-10 [39] datasets. The ensemble consists 9 classifiers for MNIST and 13 for CIFAR-10, taken from the ERAN benchmark repository [40]. A label is considered valid if at least 30% of the classifiers agree on it. Our experiments reveal that approximately 6.9% and 1.95% of the counterexamples generated by the verifiers are false positives for the MNIST and CIFAR-10 datasets, respectively. We also manually checked whether the label assigned by the ensemble to each counterexample is accurate, which forms our second experimental research question (RQ2). We found that the false positive rates based on manual analysis were 12.86% for the MNIST benchmarks and 5.65% for the CIFAR-10 benchmarks. We also ask if there are counterexamples that are not classified as one of the the ensemble labels (RQ3), which would be considered true positives under the ensemble definition. To answer this, we modified the verifiers to search for such counterexamples and reran them on both the MNIST and CIFAR-10 benchmarks. We found that the false positive rate for MNIST benchmarks reduced significantly from 12.86% to 3.22% and true positives for the same benchmarks increase from 88.63% to 96.78%. In summary, our approach and novel analysis shows that the local robustness as is usually considered can be overtly conservative. And thus, verifiers are not able to declare a model trustworthy even when the model is actually trustworthy according to the domain experts.

2. Preliminaries and Definitions

In this section, we present the preliminaries, defines the oracle, and explains the classification of counterexamples as true positives and false positives in robustness verification. As shown in Section 3, many counterexamples under standard local robustness are false positives. We also provide an *ensemble based approach* to reduce false positives.

A neural network $N : \mathbb{R}^n \to \mathbb{R}^m$ is a function that takes an n-dimensional input and produces an m-dimensional output. Here, $N(x)[i]$ denotes the logit value corresponding to the i^{th} output neuron. The final decision $\hat{N}$ of the neural network is determined using the ARGMAX [41] function, which returns the index of the maximum output value: $\hat{N}(x) = \text{ARGMAX}(N(x))$.

Standard local robustness: Most prior works [34, 42, 43, 36, 44, 45] define the local robustness verification property as follows:

Definition 1. *For a neural network N, an input x, and an input perturbation value ϵ, N is* locally robust *if for every x' such that $dist(x', x) \leq \epsilon$, we have $\hat{N}(x') = \hat{N}(x)$.*

The above property requires that a neural network's classification remains unchanged under small perturbations ϵ of the input. Here, ϵ is a user defined parameter and $dist$ is a distance metric, often taken to be some L_p norm, where $p \in \{1, 2, \infty\}$. An input x' is considered a counterexample if $dist(x', x) \leq \epsilon$ and $\hat{N}(x') \neq \hat{N}(x)$.

Oracle, true positive, and false positive: An oracle is a function that takes an input x and assigns one or more labels to the input. The oracle models the domain expert's view of the input image x. Let us assume the neural network classifies an input x into one of the classes from the set $[m]$, where $[m] = \{0, 1, \ldots, m-1\}$.

Definition 2. *An oracle $\mathbb{O}$ is a function $\mathbb{O} : \mathbb{R}^n \to 2^{[m]}$, where $2^{[m]}$ is the power set of $[m]$. The oracle assigns one or more labels to the input image x.*

Intuitively, the oracle $\mathbb{O}$ assigns the ground truth labels to the input image x. It may assign more than one label to an image if the image visually resembles more than one class. For example, in Figure 1(a), the image on the left is a seed image x, which resembles both 0 and 6, we may have $\mathbb{O}(x) = \{0, 6\}$. Using the oracle, we can define the notions of true and false positives in the context of robustness verification of neural networks.

Definition 3. *An input counterexample x' is a* true positive *(resp.* false positive*) in a neural network N if $\hat{N}(x') \notin \mathbb{O}(x')$ (resp. $\hat{N}(x') \in \mathbb{O}(x')$).*

In Figure 1(a), the image on the left is a seed image x with $\hat{N}(x) = 0$, and the image on the right is a counterexample x' reported as per Definition 1 with $\hat{N}(x') = 6$. Since $\mathbb{O}(x') = \{0, 6\}$, we have $\hat{N}(x') \in \mathbb{O}(x')$. Therefore, it is not a genuine bug, and we consider it a false positive. A false positive does not violate the oracle's view of local robustness, as the oracle $\mathbb{O}$ considers the counterexample x' to be similar to the seed image x, but violates the standard property. On the other hand, a true positive violates both the oracle's view of local robustness and the standard local robustness property. In Figure 1(c), the image on the left is a seed image x with $\hat{N}(x) = 1$, and the image on the right is a counterexample x' reported as per the standard property with $\hat{N}(x') = 3$. Since $\mathbb{O}(x') = \{1\}$ and $\hat{N}(x') \notin \mathbb{O}(x')$, this indicates a true counterexample, as the oracle's classification and the neural network's classification are completely different. We can use true positives and false positives to evaluate the results of the standard local robustness property.

Ensemble-Based Approach to Reduce False Positives: As illustrated in Figure 1(a), the original image has a ground truth label of 0, but it also visually resembles as 6. Therefore, it is unreasonable to strictly require the classifier not to predict 6; rather, it should be allowed to classify the image as either 0 or 6. Motivated by this intuition, we propose using an ensemble of state-of-the-art classifiers for the corresponding dataset, which is formally defined in Section 3. Let $E(x)$ denote the set of labels assigned to the input x by the ensemble of classifiers.

Definition 4. *For a neural network N, an input x, and an input perturbation value ϵ, N is considered* ensemble-guided locally robust *at x if $\hat{N}(x) \in E(x)$ and for every x' such that $dist(x', x) \leq \epsilon$, we have $\hat{N}(x') \in E(x)$.*

We can modify the solver query for the ensemble-aware local robustness property as $\exists x'\ dist(x', x) \leq \epsilon \wedge \bigvee_{i=1, i \notin E(x)}^{m} \bigwedge_{j \in E(x)} N(x')[j] \leq N(x')[i]$. If this query is satisfiable, then we obtain a counterexample x' on which the network N disagrees with the ensemble prediction on the original input x.

The above definition is similar to the property (affinity robustness) presented in [46], where the expert provides a set of sets of possible labels among which the network must classify. In affinity robustness, the top-k classes refer to the k classes with the highest logit values ($N(x)$) in the network output. Affinity robustness requires that for some k, the top-k classes remain the same across both x and x', and moreover, the top-k classes must be a subset of one of the expert-provided set of labels. In contrast, our proposed local robustness property requires only that the top-1 (predicted) output of the neural network on input x' belongs to the set of labels provided by the ensembles. Our property is incomparable to the notion of affinity robustness.

3. Experiments

We evaluate the standard robustness property using the state-of-the-art (SOTA) verifier $\alpha\beta$-CROWN [38] by reporting the number of false positives and true positives on a standard class of benchmarks.

Benchmarks: We considered networks trained on the MNIST [37] and CIFAR-10 [39] datasets. All networks were obtained from VNNCOMP [47, 1, 48, 49]. Table 1 provides the details of these networks.

We randomly selected 1000 images from each dataset. For the neural networks trained on MNIST, we used two different values of input perturbation ϵ, resulting in approximately 2000 benchmark instances per network. For CIFAR-10 networks, we used a single value of ϵ, leading to approximately 1000 benchmark instances per network. The values of ϵ were chosen to match those used in VNNCOMP and are shown in the last column of Table 1. Many neural networks approximate the human decision-making process with the goal of reducing human effort. However, it is impractical for humans to act as oracles for all counterexample images. To address this, we propose using an ensemble of AI models as a proxy for the oracle, defined as follows.

Definition 5. *Let $\mathbb{A}$ be a set of AI models and agreement threshold k be a positive integer. We define an* ensemble oracle *$E(x)$ such that, for each x, $E(x) = \{y \mid |\{N \in \mathbb{A} \mid \hat{N}(x) = y\}| \geq k\}$.*

Table 1
Networks details

Category	Network name	#layers	#activation units	adv trained	ϵ-values
MNIST	mnist-net-256×2.onnx	2-FC	0.51K	No	0.03,0.05
	mnist-net-256×4.onnx	4-FC	1.02K	No	0.03,0.05
	mnist-net-256×6.onnx	6-FC	1.54K	No	0.03,0.05
CIFAR-10	cifar-base-kw.onnx	2-Conv, 2-FC	3.17K	Yes	0.03
	cifar-deep-kw.onnx	4-Conv, 2-FC	6.77K	Yes	0.03
	cifar-wide-kw.onnx	2-Conv, 2-FC	6.24K	Yes	0.03

Table 2
Ensemble of neural network classifiers used for the MNIST dataset

Network name	#layers	adv trained	adv training methods
mnist-relu-5-100.onnx	5-FC	Yes	DiffAI
ffnnRELU-Point-6-500.onnx	6-FC	No	-
ffnnRELU-PGDK-w-0.1-6-500.onnx	6-FC	Yes	PGD
ffnnRELU-PGDK-w-0.3-6-500.onnx	6-FC	Yes	PGD
convSmallRELU-Point.onnx	2-Conv, 2-FC	No	-
convSmallRELU-PGDK.onnx	2-Conv, 2-FC	Yes	PGD
convSmallRELU-DiffAI.onnx	2-Conv, 2-FC	Yes	DiffAI
convMedGRELU-Point.onnx	2-Conv, 2-FC	no	-
convBigRELU-DiffAI.onnx	4-Conv, 3-FC	Yes	DiffAI

Table 3
Ensemble of neural network classifiers used for the CIFAR-10 dataset

Network name	#layers	adv trained	arv training method
cifar-relu-9-200.onnx	10-FC	No	-
ffnnRELU-PGDK-w-0.0078-6-500.onnx	7-FC	Yes	PGD
convSmallRELU-Point.onnx	2-Conv, 2-FC	No	-
convSmallRELU-PGDK.onnx	2-Conv, 2-FC	Yes	PGD
convSmallRELU-DiffAI.onnx	2-Conv, 2-FC	Yes	DiffAI
convMedGRELU-Point.onnx	2-Conv, 2-FC	No	-
convMedGRELU-PGDK-w-0.0313.onnx	2-Conv, 2-FC	Yes	PGD
cifar-conv-maxpool.onnx	4-Conv, 2-Maxpool, 3-FC	no	-
convBigRELU-DiffAI.onnx	4-Conv, 3-FC	Yes	DiffAI
ResNetTiny-PGD.onnx	1-Conv, 5-res-blocks (3-Conv), 2-FC	Yes	PGD
ResNet18-PGD.onnx	1-Conv, 5-res-blocks (2-Conv), 3-res-blocks (3-Conv), 2-FC	Yes	PGD
ResNet34-DiffAI.onnx	1-Conv, 13-res-blocks (2-Conv), 3-res-blocks (3-Conv), 3-FC	Yes	DiffAI
SkipNet18-DiffAI.onnx	11-Conv, 3-res-blocks (3-Conv), 3-FC	Yes	DiffAI

The ensemble oracle $E(x)$ returns a set of labels for the input image x based on the agreement of the AI models in $\mathbb{A}$. The ensemble is defined such that it returns a label if at least k classifiers agree on that label. Here, k is a hyperparameter. If no label meets this criterion, we relax the requirement by decrementing the value of k. If k reaches 1, we select the label with the highest prediction confidence. The AI models can be of various types, such as neural networks, decision trees, or random forests. These models may be trained on different datasets, which might not always be publicly accessible. By leveraging such models, we aim to incorporate the collective knowledge of the community into our analysis. In this work, we use neural network classifiers as our AI models. For the MNIST dataset, we use $|A| = 9$ classifiers with a threshold $k = 3$, and for the more complex CIFAR-10 dataset, we use $|A| = 13$ classifiers with $k = 4$. The intuition behind these thresholds is that a label is accepted if approximately 30% of the networks agree on it. If we choose k to be large, we may disallow diversity of opinion and if we choose k to be small, a single member of ensemble may pollute the predictions. We use all the classifiers in ensembles from the ERAN repository [40], ranging from simple fully connected to complex classifiers, from simply trained to adversarially trained classifiers. To adversarially train the networks [40], DiffAI [50] and PGD [15] methods are used. DiffAI uses abstract interpretation to create a range of safe adversarial examples, helping the model become more robust in different input areas. PGD (Projected Gradient Descent) is an attack-based method that trains the model using adversarial examples made by slightly changing inputs through gradient steps within a fixed limit. Tables 2 and 3 show details about the selected neural network classifiers.

We conduct the following three experiments: we identify the false positives (FP) and true positives (TP) for the standard robustness property using the ensemble, we manually analyze the accuracy of

Table 4
Results obtained using the standard robustness property with $\alpha\beta$-CROWN, along with ensemble-based analysis of False Positives (FP) and True Positives (TP) for the MNIST (Table a) and CIFAR-10 (Table b) networks.

Result	mnist-net-256×2.onnx		mnist-net-256×4.onnx		mnist-net-256×6.onnx	
epsilons	0.03	0.05	0.03	0.05	0.03	0.05
FP	48	49	30	33	16	14
TP	647	832	310	551	297	493
Verified	294	96	518	160	402	102
Timeout	4	16	68	182	125	231

(a) MNIST results

Result	cifar-base-kw.onnx	cifar-deep-kw.onnx	cifar-wide-kw.onnx
epsilons	0.03	0.03	0.03
FP	12	18	45
TP	603	673	880
Verified	14	10	13
Timeout	38	36	62

(b) CIFAR-10 results

Table 5
MNIST results using the ensemble-based approach with $\alpha\beta$-CROWN, along with manual analysis of FP and TP

Result	mnist-net-256×2.onnx		mnist-net-256×4.onnx		mnist-net-256×6.onnx	
epsilons	0.03	0.05	0.03	0.05	0.03	0.05
FP	22	28	12	27	13	22
TP	660	846	324	577	321	522
Verified	307	103	551	174	431	116
Timeout	4	16	65	180	121	230

the decisions of the ensemble, and we construct the ensemble-based robustness property verifier and manually validate the resulting false positives and true positives.

False Positives Analysis Using Ensemble (RQ1): Tables 4(a) and 4(b) present the number of true positives (TP) and false positives (FP) identified using the ensemble base oracle defined above. We do not analyze false negatives or true negatives, as it is generally infeasible to determine in advance whether a neural network is locally robust around a given input. To empirically evaluate RQ1, we observed false positive rates of 6.9% for the MNIST benchmarks and 1.95% for the CIFAR-10 benchmarks. The FP rate is higher for the MNIST benchmarks compared to the CIFAR-10 benchmarks. A potential reason is that many MNIST images resemble more than one label, whereas CIFAR-10 images tend to be more distinct. Overall, we observe that the number of false positives is significant and merits further consideration.

Manual Analysis of the Ensemble (RQ2): To evaluate the accuracy of the ensemble, we conducted a manual analysis of the false positives (FPs) and true positives (TPs) reported in Tables 4(a) and 4(b). Since it is impractical to manually examine all positive cases, we focused on the first column of each table (`mnist-net-256×2.onnx` and `cifar-base-kw.onnx`, with $\epsilon = 0.03$). For MNIST, we found that 13 out of 48 reported FPs were actually TPs, and 44 out of 647 reported TPs were actually FPs. For CIFAR-10, 4 out of 12 reported FPs were actually TPs, and 27 out of 603 reported TPs were actually FPs. Overall, this results in a false positive rate of 12.86% for MNIST and 5.67% for CIFAR-10 on the selected benchmark networks. This analysis demonstrates that although an approximation of the oracle can reduce human effort, it is not perfect.

Ensemble-based Property Analysis (RQ3): We guide the robustness property using the previously described ensemble, allowing misclassifications into ensemble-predicted labels (Section 2), and use humans as oracles to analyze counterexamples. Since human evaluation for all counterexamples is impractical, we limit this analysis to the `mnist-net-256×2.onnx` network with $\epsilon = 0.03$. Table 5 summarizes the number of FPs and TPs. We observed that the false positive rate dropped from 12.86% to 3.22%, while the true positive rate increased from 88.63% to 96.78%.

4. Conclusion

In this paper, we considered true and false positives in the context of the local robustness using oracles. We used an ensemble of classifiers as an approximation of the oracle and analyzed the robustness of neural networks with respect to this ensemble. We showed that the standard local robustness property can report many false positives under this oracle and proposed a new ensemble-guided local robustness property that can help reduce the number of false positives.

References

[1] S. A. Emami, P. Castaldi, A. Banazadeh, Neural network-based flight control systems: Present and future, Annual Reviews in Control 53 (2022) 97–137.

[2] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to end learning for self-driving cars, CoRR abs/1604.07316 (2016). URL: http://arxiv.org/abs/1604.07316. arXiv:1604.07316.

[3] A. Gupta, A. Anpalagan, L. Guan, A. S. Khwaja, Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues, Array 10 (2021) 100057.

[4] P. S. Chib, P. Singh, Recent advancements in end-to-end autonomous driving using deep learning: A survey, IEEE Transactions on Intelligent Vehicles 9 (2023) 103–118.

[5] S. Kuutti, R. Bowden, Y. Jin, P. Barber, S. Fallah, A survey of deep learning applications to autonomous vehicle control, IEEE Transactions on Intelligent Transportation Systems 22 (2020) 712–733.

[6] F. Amato, A. López, E. M. Peña-Méndez, P. Vaňhara, A. Hampl, J. Havel, Artificial neural networks in medical diagnosis, Journal of applied biomedicine 11 (2013) 47–58.

[7] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL: http://arxiv.org/abs/1412.6572.

[8] Y. Song, R. Shu, N. Kushman, S. Ermon, Constructing unrestricted adversarial examples with generative models, Advances in neural information processing systems 31 (2018).

[9] I. Dunn, H. Pouget, D. Kroening, T. Melham, Exposing previously undetectable faults in deep neural networks, in: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2021, pp. 56–66.

[10] I. Dunn, L. Hanu, H. Pouget, D. Kroening, T. Melham, Evaluating robustness to context-sensitive feature perturbations of different granularities, arXiv preprint arXiv:2001.11055 (2020).

[11] I. Dunn, H. Pouget, T. Melham, D. Kroening, Adaptive generation of unrestricted adversarial inputs, arXiv preprint arXiv:1905.02463 (2019).

[12] J. Hayes, G. Danezis, Learning universal adversarial perturbations with generative models, in: 2018 IEEE Security and Privacy Workshops (SPW), IEEE, 2018, pp. 43–49.

[13] S. Baluja, I. Fischer, Learning to attack: Adversarial transformation networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[14] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, D. Song, Generating adversarial examples with adversarial networks, arXiv preprint arXiv:1801.02610 (2018).

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=rJzIBfZAb.

[16] A. Lomuscio, L. Maganti, An approach to reachability analysis for feed-forward relu neural networks, CoRR abs/1706.07351 (2017). URL: http://arxiv.org/abs/1706.07351. arXiv:1706.07351.

[17] M. Fischetti, J. Jo, Deep neural networks and mixed integer linear optimization, Constraints An Int. J. 23 (2018) 296–309. URL: https://doi.org/10.1007/s10601-018-9285-6. doi:10.1007/s10601-018-9285-6.

[18] S. Dutta, S. Jha, S. Sankaranarayanan, A. Tiwari, Output range analysis for deep feedforward neural networks, in: NASA Formal Methods Symposium, Springer, 2018, pp. 121–138.

[19] C.-H. Cheng, G. Nührenberg, H. Ruess, Maximum resilience of artificial neural networks, in: International Symposium on Automated Technology for Verification and Analysis, Springer, 2017, pp. 251–268.

[20] G. Katz, C. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An efficient smt solver for verifying deep neural networks, in: International conference on computer aided verification, Springer, 2017, pp. 97–117.

[21] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, et al., The marabou framework for verification and analysis of deep neural networks, in:

International Conference on Computer Aided Verification, Springer, 2019, pp. 443–452.
[22] R. Ehlers, Formal verification of piece-wise linear feed-forward neural networks, in: International Symposium on Automated Technology for Verification and Analysis, Springer, 2017, pp. 269–286.
[23] X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety verification of deep neural networks, in: International conference on computer aided verification, Springer, 2017, pp. 3–29.
[24] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, J. Z. Kolter, Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification, Advances in Neural Information Processing Systems 34 (2021) 29909–29921.
[25] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, C. Hsieh, Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers, CoRR abs/2011.13824 (2020). URL: https://arxiv.org/abs/2011.13824. arXiv:2011.13824.
[26] H. Zhang, S. Wang, K. Xu, L. Li, B. Li, S. Jana, C. Hsieh, J. Z. Kolter, General cutting planes for bound-propagation-based neural network verification, CoRR abs/2208.05740 (2022). URL: https://doi.org/10.48550/arXiv.2208.05740. doi:10.48550/arXiv.2208.05740. arXiv:2208.05740.
[27] S. Wang, K. Pei, J. Whitehouse, J. Yang, S. Jana, Formal security analysis of neural networks using symbolic intervals, in: 27th {USENIX} Security Symposium ({USENIX} Security 18), 2018, pp. 1599–1614.
[28] S. Wang, K. Pei, J. Whitehouse, J. Yang, S. Jana, Efficient formal safety analysis of neural networks, in: Advances in Neural Information Processing Systems, 2018, pp. 6367–6377.
[29] Y. Y. Elboher, J. Gottschlich, G. Katz, An abstraction-based framework for neural network verification, in: International Conference on Computer Aided Verification, Springer, 2020, pp. 43–65.
[30] P. Yang, R. Li, J. Li, C.-C. Huang, J. Wang, J. Sun, B. Xue, L. Zhang, Improving neural network verification through spurious region guided refinement, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2021, pp. 389–408.
[31] X. Lin, H. Zhu, R. Samanta, S. Jagannathan, Art: Abstraction refinement-guided training for provably correct neural networks, in: 2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020, IEEE, 2020, pp. 148–157. URL: https://doi.org/10.34727/2020/isbn.978-3-85448-042-6_22. doi:10.34727/2020/isbn.978-3-85448-042-6_22.
[32] P. Cousot, R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1977, pp. 238–252.
[33] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, C. Tinelli, cvc4, in: Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23, Springer, 2011, pp. 171–177.
[34] G. Singh, T. Gehr, M. Mirman, M. Püschel, M. T. Vechev, Fast and effective robustness certification, in: S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018, pp. 10825–10836. URL: https://proceedings.neurips.cc/paper/2018/hash/f2f446980d8e971ef3da97af089481c3-Abstract.html.
[35] G. Singh, T. Gehr, M. Püschel, M. T. Vechev, Boosting robustness certification of neural networks, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019. URL: https://openreview.net/forum?id=HJgeEh09KQ.
[36] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, L. Daniel, Efficient neural network robustness certification with general activation functions, Advances in neural information processing systems 31 (2018).
[37] L. Deng, The mnist database of handwritten digit images for machine learning research [best of the web], IEEE signal processing magazine 29 (2012) 141–142.
[38] H. Zhang, K. Xu, S. Wang, et al., alpha-beta-CROWN: Complete and incomplete neural network verification, https://github.com/Verified-Intelligence/alpha-beta-CROWN, ???? Accessed: 2025-07-17.
[39] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).

[40] M. N. Müller, G. Singh, M. Balunovic, G. Makarchuk, A. Ruoss, F. Serre, M. Baader, D. D. Cohen, T. Gehr, A. Hoffmann, J. Maurer, M. Mirman, C. Müller, M. Püschel, P. Tsankov, M. Vechev, Eran: Eth robustness analyzer for neural networks,
urlhttps://github.com/eth-sri/eran, ???? Secure, Reliable, and Intelligent Systems Lab (SRI), ETH Zurich. Accessed: 2025-07-17.
[41] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, volume 4, Springer, 2006.
[42] G. Singh, T. Gehr, M. Püschel, M. Vechev, An abstract domain for certifying neural networks, Proceedings of the ACM on Programming Languages 3 (2019) 1–30.
[43] G. Singh, R. Ganvir, M. Püschel, M. Vechev, Beyond the single neuron convex barrier for neural network certification, Advances in Neural Information Processing Systems 32 (2019).
[44] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, C.-J. Hsieh, Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=nVZtXBI6LNn.
[45] D. Zhou, C. Brix, G. A. Hanasusanto, H. Zhang, Scalable neural network verification with branch-and-bound inferred cutting planes, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.
[46] K. Leino, M. Fredrikson, Relaxing local robustness, Advances in Neural Information Processing Systems 34 (2021) 17072–17083.
[47] S. Bak, C. Liu, T. Johnson, The second international verification of neural networks competition (vnn-comp 2021): Summary and results, arXiv preprint arXiv:2109.00498 (2021).
[48] C. Brix, S. Bak, C. Liu, T. T. Johnson, The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results, 2023. URL: https://arxiv.org/abs/2312.16760. arXiv:2312.16760.
[49] C. Brix, S. Bak, T. T. Johnson, H. Wu, The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results, arXiv preprint arXiv:2412.19985 (2024).
[50] M. Mirman, T. Gehr, M. Vechev, Differentiable abstract interpretation for provably robust neural networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 3578–3586.