Verifying rich robustness properties for neural networks

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

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

Abstract. Robustness is a important problem in AI alignment and safety, with models such as neural networks being increasingly used in safety-critical systems. In the last decade, a large body of work has emerged on local robustness, i.e., checking if the decision of a neural network remains unchanged when the input is slightly perturbed. However, many of these approaches require specialized encoding and often ignore the confidence of a neural network on its output. In this paper, our goal is to build a generalized framework to specify and verify variants of robustness in neural network verification. We propose a specification framework using a simple grammar, which is flexible enough to capture most existing variants. This allows us to introduce new variants of robustness that take into account the confidence of the neural network in its outputs. Next, we develop a novel and powerful unified technique to verify all such variants in a homogeneous way, viz., by adding a few additional layers to the neural network. This enables us to use any state-of-the-art neural network verification tool, without having to tinker with the encoding within, while incurring an approximation error that we show is bounded. We perform an extensive experimental evaluation over a large suite of 8870 benchmarks having 138M parameters in a largest network, and show that we are able to capture a wide set of robustness variants and outperform direct encoding approaches by a significant margin.

1 Introduction

Neural networks are increasingly being used in safety-critical applications such as autonomous vehicles, medical diagnosis, and speech recognition [1,2,3]. However, as is well known e.g., in the image classification setting [4], a small perturbation in the input image may lead to misclassifying the output of the network, even if the perturbed image looks exactly like the original image. Such examples are called adversarial examples and there is a long line of work [5,6,7] that focuses on generating such examples. However, failure to generate adversarial examples does not guarantee that they do not exist, and one often resorts to formal methods techniques for such guarantees.

One such paradigm has emerged around local (i.e., around a given input) robustness verification for image classification, where different techniques have been developed, including sound but incomplete techniques such as [8,9,10,11]

as well as less scalable but complete techniques such as [12,13,14,15,16]. The success of these approaches can be judged by the VNN-COMP competition for robustness verification of neural networks [17], which is now in its 5th edition, and has several competing solvers working on neural networks of fairly large size, with the largest network having approximately 13.16 Million activations.

Different Robustness Variants Different applications require different variants of robustness (e.g., see [18]). Further, many existing works on robustness verification regard answers returned by classifiers as binary, ignoring the fact that classifiers provide a confidence in the counterexample in terms of the classification probability, which is computed by the well known SOFTMAX function [19]. For instance, in Figure 1(a-b) an image from the CIFAR-10 [20] dataset, is misclassified after an input perturbation, but with a very low confidence. Should such a network be called non-robust, if all misclassifications are such low confidence ones? This motivates a more *relaxed* notion of robustness which would allow such examples. At the other extreme, one could argue for a *stronger* notion of robustness, saying that the network in Figure 1(c-e), should be called non-robust since under minor input perturbations, the confidence in the classification has vastly changed (even if no misclassification is reported). Finally, we may consider a Lipschitz-continuity motivated smoothness criterion [18,21,22,23] which says that a neural network is not *smooth-robust* if there are counterexamples with varying confidences as depicted in Figure 1(f-g).

To be able to handle variants such as those given above, the first step is to have a common way to specify all such variants. In this work, we use a simple grammar via arbitrary Boolean combinations of linear inequalities and show that it captures all the above introduced variants and many more. But when confidence is included in the property definition, it becomes necessary to approximate the confidence (SOFTMAX) function to analyze these properties effectively, which we do, with formal guarantees on the error.

Difficulties in encoding. The main difficulty that remains is whether such rich properties modeled as Boolean combinations of linear inequalities can directly be encoded in state-of-the-art tools. In fact, as standardized in VNN-COMP, robustness properties are specified in a special VNNLIB format as pre- and post-conditions that a neural network must satisfy. Although VNNLIB supports arbitrary Boolean combinations of linear constraints, most state-of-the-art neural network verifiers are optimized for simpler post-conditions which typically involve only disjunctions or conjunctions of linear atoms of over the outputs. Thus, verifying properties with multi-layered conjunctions, disjunctions, or combinations of both poses significant challenges: (i) Modifying the verifier's code to handle complex post-conditions requires a deep understanding of its implementation, which can be challenging for users unfamiliar with the codebase, i.e. $\alpha\beta$-CROWN [24] (ii) The source code for some commercial verifiers may not be publicly available, restricting the ability to make modifications. (iii) Even when access is available, adapting the code for each new property format is time-consuming and prone to errors. (iv) Constraint-based tools, i.e., MARABOU [25],

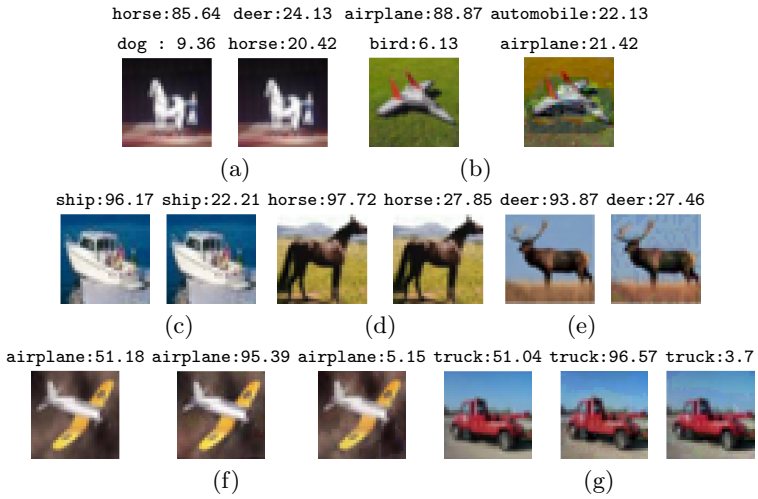


Fig. 1: **Relaxed Robustness: (a-b)** The network `convBigRELU-PGD.onnx` correctly classified the original image (left) as `horse` (and resp. `airplane`) with high confidence. With an input perturbation of 16/255, we can find misclassified images (right) but with low confidence. In fact, it turns out all counterexamples have low confidence and hence verification succeeds under the relaxed robustness criterion with an 80% confidence threshold, while state-of-the-art verifiers would have declared this network non-robust. **Strong robustness: (c-e)** The network `convBigRELU-PGD.onnx` classified the original image (left) of class `ship/horse/deer` with very high confidence and we found images (right) within perturbation of 16/255, such that the confidence drops drastically, although the class remains same. These images are robust with respect to the standard and relaxed robust criteria but not robust with respect to the strong robust criteria if confidence is allowed to fall upto 30%. **Smoothness: (f-g)** The left image, labeled as `Airplan/Truck`, is taken from the Cifar-10 dataset and is classified correctly with a medium confidence of $\sim 50\%$ by the neural network `cifar10-2-255.onnx`. Under an $\epsilon = 16/255$ perturbation, we obtain images with much higher and lower confidences, showing drastic variations. The bottomline is that the above requirements may vary across applications, and users can define many more requirements tailored to specific needs.

allow such properties to be encoded directly as constraints, but users are then limited to a particular solver only and lack to achieve the scalability. (v) Advanced techniques like Projected Gradient Descent (PGD) attacks, which are highly effective at finding counterexamples in neural networks, cannot always be applied directly due to the complexity of certain post-conditions.

Simplifying post-conditions by adding layers. To overcome these challenges, we propose technique that simplifies arbitrary post-conditions by appending a few additional layers to the neural network. This transformation converts the post-condition into a straightforward form, such as $y \geq 0$ or $y > 0$, where y is the out-

put of an added node in the neural network, while maintaining low error bounds. This intricate transformation generates new layers that encode parts of the formulas and composes the outputs of these layers according to the Boolean operations in the formulas. Since most verifiers support the ReLU activation function, we employ ReLU to model the Boolean operations in the post-conditions. The output of a sum of ReLUs can model conjunction or disjunction: if all inputs are negative, the output is zero; if any input is positive, the output is positive. Since we use ReLU operators to model all Boolean operations, conjunctions and disjunctions interpret input signals in opposite ways i.e., for a conjunction, a negative input yields 1 and a positive input yields 0, while for a disjunction, a positive input yields 0 and a negative input yields 1. To enable composition of conjunction outputs into disjunctions and vice versa, we introduce a novel technique that reverses the outputs using a flip operation, while maintaining low error bounds. These simplified post-conditions can be verified using any state-of-the-art verifier as a black box, eliminating the need for code modifications and ensuring seamless integration with existing tools.
Summarizing, *our main contributions* are:

- We define a grammar for post-conditions that capture a rich set of robustness properties for neural networks. Our grammar allows us to capture existing notions from the literature such as strong [18] and top-k robustness [26].
- Our framework also allows us to define novel notions of relaxed robustness, that incorporate confidence by distinguishing low-confidence counterexamples during robustness analysis. Since the confidence is obtained from the output of the SOFTMAX function, we approximate of the effect of the SOFTMAX, so that it can be encoded into Boolean combinations of linear constraints, with formal guarantees on the error incurred by the approximation.
- We provide a new technique to analyze all such old and new variants. Instead of changing the encoding for each variant, our idea is to provide a generalized encoding (for all properties that can be expressed using this grammar) by adding layers to the neural network. Our encoding enables the use of state-of-the-art neural network verifiers like $\alpha\beta$-CROWN, PyRAT.
- We perform a wide set of experiments over benchmarks ranging from $0.51K$ to $13.16M$ non-activation units. Our evaluation shows that we can use our technique to verify different robustness, and that we outperform the direct encodings that have been tried in the literature [18,27].

Structure of the paper. We define preliminaries in Section 2. Section 3 introduces different notions of confidence-based properties, the SOFTMAX approximation, and the grammar. In Section 4, we present our unifying technique to verify properties by encoding post-conditions as layers. Section 5 considers non-confidence-based properties we can capture. Section 6 reports our experiments demonstrating the effectiveness of our translation, and we conclude in Section 7. Missing proof details and additional experiments can be found in the Appendix.

Related Work. In addition to the works already discussed, we discuss a few recent developments. The paper [27] recently introduced softmax-based confidence

and its verification focusing on computing bounds on the output of the softmax function based on the bounds of its input. Our approximation is different, as explained in Section 3.1 (and App. A), and is fine-tunable with respect to the desired confidence level and usable with any neural network verified, not just constraint-based ones. [28] introduces convex bounds on the softmax function, providing both lower and upper bounds that are compatible with convex optimization. A case-study surveying different robustness variants is presented in [18] which includes strong and smooth robustness. The strong definition in [18] however, requires the logit value of the classified class should not drop below a certain threshold. We apply the same condition using confidence values, which are normalized between 0 and 100. Smoothness is an instance of Lipschitz continuity in neural networks [23,22,21,18,29], which bounds each class's logit values using a Lipschitz constant. Ours is a simplified version as we bound the confidence of the seed image's class by a threshold. The paper [30] performs robustness verification for cross-entropy loss functions, but there is a subtle difference, as we do not use the logarithm of the softmax output. Furthermore, the technique used in that paper is based on a *Satisfiability Modulo Convex Programming* [30] approach, whereas we use $\alpha\beta$-CROWN, which relies on a different suite of solver techniques, including PGD-attack [31], CROWN [32], α-CROWN [16], β-CROWN [15], among others. Other papers [33,34] have also explored encoding constraints into neural networks. The paper [33] appends a user-requirement as a layer to the neural network and trains the network to satisfy these requirements, however, their approach is limited to propositional formulae, and was not applied to post training robustness verification. The framework [34] simplifies neural networks by statically replacing unsupported operations with supported ones, enabling the underlying verifier to handle them. It also simplifies properties by decomposing them into smaller sub-properties that can be verified individually by the verifier. Finally, [35] introduced the concepts of top-k relaxed robustness and affinity robustness. As we show in Section 5, these can also be captured by our approach.

2 Preliminaries

A neural network is composed of layers $l_0, l_1, \ldots, l_k$, where k represents the number of layers. Each layer contains an ordered set of neurons. A neuron n_{ij} represents the j-th neuron of layer l_i, and $|l_i|$ represents the number of neurons in layer l_i. Layers l_0 and l_k are the input and output layers, respectively; all other layers are hidden layers. We assume that input and output layers do not have any activation function, while each hidden layer l_i has an activation function σ_i. Some well-known non-linear activation functions include: ReLU: $\sigma_i(y) = \max(0, y)$; LeakyReLU: $\sigma_i(y) = \max(\alpha y, y)$, where α is the slope hyperparameter for the negative input region ($y \leq 0$); Sigmoid: $\sigma_i(y) = \frac{1}{1+e^{-y}}$; and Tanh: $\sigma_i(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$. We consider any activation function as long as it is supported by state of the art verifiers. Each layer l_i, except the input layer, has a weight matrix $W_i \in \mathbb{R}^{|l_i| \times |l_{i-1}|}$ and a bias vector $B_i \in \mathbb{R}^{|l_i|}$. Let

Φ_i denote the function representing the computation from layer l_0 to l_i. Then, $\Phi_i(x) = \boldsymbol{\sigma}_i(W_i\Phi_{i-1}(x) + B_i)$, where $\boldsymbol{\sigma}_i$ denotes the vectorized version of the scalar activation function σ_i, and $\Phi_0(x) = x$. For the output layer l_k, the network output is given by $N(x) = \Phi_k(x) = W_k\Phi_{k-1}(x) + B_k$. $N_i(x)$ represents the logits value of i^{th} output neuron. The final decision $\hat{N}$ of a neural network is determined using the ARGMAX [36] function, which returns the index/class of the maximum output value: $\hat{N}(x) = \text{ARGMAX}(N(x))$. Thus, a neural network $N : \mathbb{R}^n \to \mathbb{R}^m$ is a function that takes an input of n dimensions and produces an output of m dimensions, i.e., $n = |l_0|$ and $m = |l_k|$.

To construct a verification query, we define predicates P and Q over the input and output layers, respectively. A *verification query* is defined as a triple $\langle N, P, Q\rangle$. We say that the query holds if, for every input x such that $x \models P$, it follows that $N(x) \models Q$, i.e., $\forall x, P(x) \implies Q(N(x))$. If the query does not hold, then there exists a counterexample x such that $x \models P$ and $N(x) \not\models Q$. For instance, let us see how the standard (local) robustness definition can be written as a verification query in the above form.

Standard Local Robustness. For a given image x^* (sometimes called the seed image), a neural network N, and an input perturbation parameter ϵ, the local robustness property [37,38,11,39,32], states that for all images x close to x^*, the network's prediction for x should be the same as its prediction for x^*. We represent the closeness as $\texttt{dist}(x^*, x) = \|x^* - x\|_p$, where $p \in \{1, 2, \infty\}$. Formally:

$$\forall x\ \texttt{dist}(x^*, x) \leq \epsilon \implies \hat{N}(x^*) = \hat{N}(x) \tag{1}$$

The above property thus corresponds to the verification query $\langle N, P, Q\rangle$, where the predicate P is $\texttt{dist}(x^*, x) \leq \epsilon$ and Q is $\hat{N}(x^*) = \hat{N}(x)$. We call x a *misclassified* input if $\hat{N}(x^*) \neq \hat{N}(x)$.

Softmax-based confidence computation in Neural Networks. In a classification neural network, each output neuron is associated with a class, and the output of the neuron is the logit value for that class. The softmax function is used to convert these logits into a probability distribution over the classes, which indicates the confidence of the network in its prediction. This is formally defined in Eq 2 below, where c represents the class for which we compute the confidence, and y_i denotes the logit value for the i^{th} dimension (class i) of the output layer. This definition is adopted from [19,28,27]

$$\text{CONF}((y_1, .., y_m), c) = 100 \cdot \text{SOFTMAX}((y_1, .., y_m), c) = \frac{100 \cdot e^{y_c}}{\sum_{i=1}^{m} e^{y_i}}. \tag{2}$$

3 Modeling Different Robustness Variants

The standard robustness definition (in Eq 1) does not capture the confidence of the network in its prediction. However, as discussed in the Introduction, it is often meaningful to consider variants that take into account confidence. The main challenge here (as already noted in e.g, [27,18]) is that softmax-based confidence (as defined in Eq 2) involves exponentials. Hence to integrate it with the existing

solver technology (which primarily focuses on linear constraints or their Boolean combinations), one has to approximate them.

In this section, we present a list of robustness properties that are confidence-aware (some new and some inspired by earlier work), express them as modified verification queries, and also explain how they can be approximated. Then, we will present a simple grammar in which all these definitions can be expressed and which allows us to capture other (even non-confidence based) variants.

3.1 Relaxed Robustness

Let us first discuss a weaker notion of robustness that we called *relaxed robustness*, where the intuition is to ignore low-confidence counterexamples in determining robustness of the network, as shown in Figure 1(a-b). The main idea is that for all images x close to given image x^*, their classification should be the same, unless the confidence of the network on x is less than a certain threshold. Incorporating the requirement in Eq 1 gives us Eq 3 which permits low-confidence counterexamples to bypass the strict robustness criteria. Let τ be the confidence level threshold, a tunable user specified parameter.

$$\forall x \; \texttt{dist}(x^*, x) \leq \epsilon \implies \Big(\textsc{Conf}(N(x), \hat{N}(x)) < \tau \vee \hat{N}(x^*) = \hat{N}(x)\Big) \tag{3}$$

Hence, for the above property, P is as in Eq 1, but we have a new post-condition $Q_{rel} = \textsc{Conf}(N(x), \hat{N}(x)) < \tau \vee \hat{N}(x^*) = \hat{N}(x)$. Thus if the query $\langle N, P, Q_{rel}\rangle$ holds, we say that N is τ-relaxed robust (or just relaxed robust when clear from context), and otherwise, i.e., if there exists x satisfying the negation, $\neg Q_{rel}$, we say that there exists a misclassified counterexample x with confidence at least τ. Since computing $\textsc{Conf}(N(x), \hat{N}(x))$ (a non-linear function) precisely is difficult, we approximate it. Let $y_1, \ldots, y_m = N(x)$, and define $t = \hat{N}(x)$, implying $y_t = \max_{i=1}^{m}(y_i)$. Additionally, let $y_{t'}$ denote the second-highest output: $y_{t'} = \max_{i=1, i\neq t}^{m}(y_i)$. We have the following claim for any $\tau \geq 50$, which leads to our approximation. Let $\delta = -\ln(\frac{100}{\tau} - 1)$

Claim 1 *If $y_t < y_{t'} + \delta$ holds, then $\textsc{Conf}(N(x), \hat{N}(x)) < \tau$*

Proof. Assume LHS of the claim holds, i.e., $y_t < y_{t'} + \delta$. By removing all exponential terms from the denominator except for e^{y_t} and $e^{y_{t'}}$, we obtain $\textsc{Conf}(y_1, ...y_m, t) \leq \frac{100e^{y_t}}{e^{y_t}+e^{y_{t'}}}$. After scaling the right-hand side by $\frac{1}{e^{y_t}}$, we get $\textsc{Conf}(y_1, ...y_m, t) \leq \frac{100}{1+e^{-(y_t - y_{t'})}}$. Since $\delta > y_t - y_{t'}$, we have $\textsc{Conf}(y_1, ...y_m, t) < \frac{100}{1+e^{-\delta}}$. Since $\delta = -\ln(\frac{100}{\tau} - 1)$, $\textsc{Conf}(y_1, ...y_m, t) < \tau$. Finally, noting that $(y_1, \ldots, y_m) = N(x)$, and from definition of t, we get $\textsc{Conf}(N(x), \hat{N}(x)) < \tau$.

Claim 2 *If $y_t \geq y_{t'} + \delta$ holds, then $\textsc{Conf}(N(x), \hat{N}(x)) \geq \frac{100}{1+(m-1)e^{-\delta}}$.*

Proof. Let LHS of the claim be true. Since $y_{t'}$ is the second maximum, we can derive $\textsc{Conf}(y_1, ...y_m, t) \geq \frac{100e^{y_t}}{e^{y_t}+(m-1)e^{y_{t'}}}$, replacing e^i by $e^{t'}$ for all $i, i \neq t$. After scaling the fraction in the RHS by e^{y_t}, we obtain $\textsc{Conf}(y_1, ...y_m, t) \geq \frac{100}{1+(m-1)e^{y_{t'} - y_t}}$. Since $y_t \geq y_{t'} + \delta$, we obtain $\textsc{Conf}(y_1, ...y_m, t) \geq \frac{100}{1+(m-1)e^{-\delta}}$.

The above approximation is similar to the one presented in [27], but our derivation and proofs are rather different (see Appendix A for more details). Also, where they use the constraint $y_t < y_{t'} + \delta \wedge t \neq \hat{N}(x)$ to derive an approximation of the second-highest output, we compare y_t with every other logit value: $\bigwedge_{j=1,j\neq t}^{m} y_t \geq y_j + \delta$. Similarly, we apply a disjunction to determine the maximum y_t. To ensure misclassification, we impose the condition $t \neq \hat{N}(x)$. Thus, we obtain $\neg Q'_{rel}$ as follows.

$$\bigvee_{i=1,i\neq\hat{N}(x)}^{m} \quad \bigwedge_{j=1,j\neq i}^{m} (y_i \geq y_j + \delta) \tag{4}$$

where, y_i, y_j are the i^{th} and j^{th} outputs in $N(x)$, and $\delta = -\ln(\frac{100}{\tau} - 1)$. Note that the conjunct $(\hat{N}(x^*) \neq \hat{N}(x))$ in Eq 3 is already subsumed in Eq 4 above. Thus we obtain the following theorem:

Theorem 1. *Given verification query* $\langle N, P, Q'_{rel}\rangle$,

1. *if the query holds true, then* $\langle N, P, Q_{rel}\rangle$ *holds, i.e., there is no misclassified counterexample in* P *with confidence greater than* $\frac{100}{1+e^{-\delta}} = \tau$.
2. *else, there is a misclassified input* $x \models P$ *with confidence greater than or equal to* $\frac{100}{1+(m-1)e^{-\delta}}$.

Proof. 1. Suppose the query holds true, which implies that Eq 4 is unsatisfiable. This means that $\forall i, i \neq \hat{N}(x), \exists j, j \neq i, \quad y_i < y_j + \delta$. Consequently, there exists some $j \neq max$ such that $y_{\max} < y_j + \delta$, where $y_{\max} = \max(y_i)$. This further implies $y_{\max} < y_{\text{smax}} + \delta$, since $y_j \leq y_{\text{smax}}$, where y_{smax} is the second highest value in the output layer. By Claim 1, we conclude that no counterexample exists with confidence greater than $\frac{100}{1+e^{-\delta}}$.

2. Suppose the query does not hold, which implies that Eq 4 is satisfiable. This means there exists some y_i (where $y_i \neq y_{\hat{N}(x)}$) that is greater than all other logit values by a margin δ, with $\delta \geq 0$. Thus, $y_{\max} \geq y_{\text{smax}} + \delta$. By Claim 2, this counterexample of class max has confidence greater than $\frac{100}{1+(m-1)e^{-\delta}}$.

Theorem 1 ensures the soundness of our approximation (part 1.) and provides a lower bound on the confidence of any counterexample (part 2.). Specifically, if the derived query Q'_{rel} holds, then no counterexample exists with confidence greater than the threshold τ. Otherwise, any counterexample must have a confidence of at least $\frac{100}{1+(m-1)e^{-\delta}}$, which denote as τ_{lb}.

In Figure 2, we illustrate the estimation of the lower bound, by plotting confidence vs the gap between the top two outputs. The thick curve plots the user defined thresholds, and the dashed curve plots the lower bound approximation. For a target confidence threshold τ on the Y-axis, a horizontal line to the user defined threshold curve finds the δ, while the vertical line from the intersection crosses the lower bound curve at τ_{lb}; the lower bound for any counterexample.

3.2 Strong Robustness

The second variant of robustness, that we call *strong robustness*, aims to capture cases where the confidence on the seed image is high, but drops below a

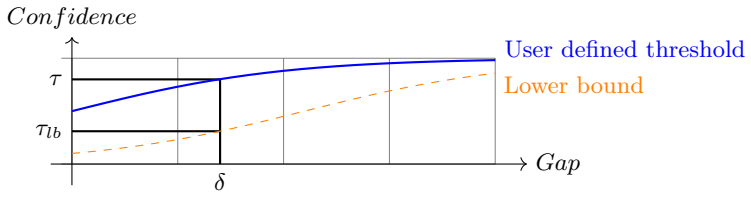


Fig. 2: Behavior of lower bound τ_{lb} and the user defined threshold τ approximations of $softmaxC$.

certain threshold within an ϵ-bounded perturbation, regardless of whether the classification label changes. In another words, a significant drop in confidence itself indicates a weakness in the underlying network. Recall that an example of such robustness was illustrated in Figure 1(c-e), where the confidence for a seed image of class `ship` dropped from 96.17% to 22.21%. The definition of strong robustness, which generalizes the notion introduced in [18], can now be formally defined as follows. Let τ_1 and τ_2 be two threshold values such that $\tau_1 > \tau_2$.

$$\forall x\ \texttt{dist}(x^*, x) \leq \epsilon \implies \Big(\textsc{Conf}(N(x), \hat{N}(x^*)) > \tau_2 \wedge \hat{N}(x^*) = \hat{N}(x)\Big) \\ \vee\ \textsc{Conf}(N(x^*), \hat{N}(x^*)) < \tau_1 \tag{5}$$

The above equation asserts that for any x in the ϵ-neighborhood of x^*, if confidence of the network on x^* is at least τ_1, then x must be classified the same as x^* with a high confidence (above threshold τ_2). Thus, we get the query $\langle N, P, Q_{str}\rangle$ where P remains the same, while $Q_{str} = (\textsc{Conf}(N(x), \hat{N}(x^*)) > \tau_2 \wedge \hat{N}(x^*) = \hat{N}(x)) \vee \textsc{Conf}(N(x^*), \hat{N}(x^*)) < \tau_1$. Note that $\textsc{Conf}(N(x^*), \hat{N}(x^*)) < \tau_1$ can be computed beforehand since x^* is a given seed image. Thus, we will only need to approximate the condition $\textsc{Conf}(N(x'), \hat{N}(x)) > \tau_2$, which we do as follows.

Let $t = \hat{N}(x^*)$ and $y_1, \ldots, y_m = N(x)$, with $y_{t'} = \max_{i=1, i\neq t}^{m}(y_i)$ being the second maximum. For any τ_2, let $\delta = -\ln(\frac{1}{m-1}(\frac{100}{\tau_2} - 1))$, where $\delta \geq 0$. Now we can show (with proof in Appendix B)

Claim 3 *If $y_t > y_{t'} + \delta$, then $\textsc{Conf}(N(x'), t) > \tau_2 \wedge t = \hat{N}(x)$, and otherwise, $\textsc{Conf}(N(x'), t) \leq \frac{100}{1+e^{-\delta}}$.*

The constraint $y_t > y_{t'} + \delta$ allows us to defines the new post-condition Q'_{str}, where instead of explicitly computing $y_{t'}$, we perform comparisons with each y_i using disjunction. Formally, we define $Q'_{str} = \neg\left(\bigvee_{i=1, i\neq t}^{m} y_t \leq y_i + \delta\right)$ and obtain a revised verification query with following guarantees.

Theorem 2. *Given verification query $\langle N, P, Q'_{str}\rangle$,*

1. *if the query holds, then $\langle N, P, Q_{str}\rangle$ holds, i.e., all inputs in P are correctly classified with confidence greater than $\frac{100}{1+(m-1)e^{-\delta}} = \tau_2$.*
2. *else, there is an example $x \models P$ such that x is either misclassified or has confidence less than $\frac{100}{1+e^{-\delta}}$.*

Proof. 1. Suppose the query holds, i.e., $\forall i \neq t,\ y_t > y_i + \delta$. This implies that $y_t > y_{t'} + \delta$, where $y_{t'} = \max_{i \neq t}(y_i)$ and $\delta \geq 0$. Hence, y_t is the maximum logit, and no misclassification occurs. Furthermore, by one direction of Claim 3, the confidence is greater than $\frac{100}{1+(m-1)e^{-\delta}}$, which corresponds to τ_2.
2. Suppose the query does not hold, i.e., $\exists i \neq t$ such that $y_t \leq y_i + \delta$. This implies that $y_t \leq y_{t'} + \delta$, where $y_{t'} = \max_{i \neq t}(y_i)$ and $\delta \geq 0$. By the other direction of Claim 3, the confidence is less than or equal to $\frac{100}{1+e^{-\delta}}$. Since $y_t \leq y_{t'} + \delta$ does not imply $y_t \leq y_{t'}$, misclassification may or may not occur.

3.3 Smoothness

A third variant is smoothness, an instance of Lipschitz continuity in neural networks [23,22,21]. The work in [18] introduces the concept of Lipschitz robustness, which bounds each class's logit values using a Lipschitz constant. We propose a simplified instance of Lipschitz robustness, where we ask that, within an ϵ-perturbation, the confidence of network should not exhibit significant variations wrt the seed image. As illustrated in Figure 1(f-g) in the Introduction, a seed image of class `Truck` was classified with 51.04% confidence, but under ϵ perturbations, the confidence varied dramatically from 3.7% to 96.57%, violating the smoothness criterion. Such drastic changes in confidence within an ϵ-bounded perturbation indicate potential issues with the underlying network. Formally, let $t = \hat{N}(x^*)$, and $\tau > 0$ a threshold, then we have:

$$\forall x\ \texttt{dist}(x^*, x) \leq \epsilon \implies |\textsc{Conf}(N(x^*), t) - \textsc{Conf}(N(x), t)| < \tau \tag{6}$$

As before, $\textsc{Conf}(N(x^*), t)$ can be pre-computed as x^* is a given seed image; so we can fix $\textsc{Conf}(N(x^*), t) = C$, a constant. Substituting this, we obtain a verification query $\langle N, P, Q_{sm}\rangle$, where $Q_{sm} = C - \textsc{Conf}(N(x), t) > -\tau \wedge C - \textsc{Conf}(N(x), t) < \tau$, with P unchanged. Now again to approximate the non-linear constraint in Q_{sm} let $y_1, \ldots, y_m = N(x)$, $y_{t'} = \max_{i=1, i\neq t}^{m}(y_i)$, $\delta_1 = -\ln\left(\frac{100}{C+\tau} - 1\right)$, and $\delta_2 = -\ln\left(\frac{1}{m-1}\left(\frac{100}{C-\tau} - 1\right)\right)$. We have:

Claim 4 *If $y_t < y_{t'} + \delta_1$ and $y_t > y_{t'} + \delta_2$ holds, then $\textsc{Conf}(N(x'), t) < \tau_1 \wedge \textsc{Conf}(N(x'), t) > \tau_2$. Otherwise, i.e., if $y_t \geq y_{t'} + \delta_1$ or $y_t \leq y_{t'} + \delta_2$ holds, then $\textsc{Conf}(N(x'), t) \geq \frac{100}{1+(m-1)e^{-\delta_1}}$ or $\textsc{Conf}(N(x'), t) \leq \frac{100}{1+e^{-\delta_2}}$.*

Again $y_t < y_{t'} + \delta_1 \wedge y_t > y_{t'} + \delta_2$ is our derived post-condition Q'_{sm}. Eq 7 shows the encoding of the $\neg Q'_{sm}$. The first part of the equation encodes $y_t \geq y_{t'} + \delta_1$, since $y_{t'} = \max_{i=1, i\neq t}^{m}(y_i)$, which is equivalent to checking $\forall i, i \neq t, y_t \geq y_i + \delta_1$. The second part of the equation encodes $y_t \leq y_{t'} + \delta_2$, which is equivalent to checking $\exists i, i \neq t, y_t \leq y_i + \delta_2$. Thus,

$$\neg(Q'_{sm}) = \bigwedge\nolimits_{i=1, i\neq t}^{m} y_t \geq y_i + \delta_1 \vee \bigvee\nolimits_{i=1, i\neq t}^{m} y_t \leq y_i + \delta_2, \tag{7}$$

It is then easy to show a similar guarantee on the approximation.

Theorem 3. *Given a verification query $\langle N, P, Q'_{sm}\rangle$:*

1. *If the query holds, then* $\langle N, P, Q_{sm}\rangle$ *holds, i.e., for all inputs* $x \models P$*, the confidence remains bounded as* $|C - \textsc{Conf}(N(x), t)| < \tau$*.*
2. *Otherwise, there exists* $x \models P$ *such that the confidence is either greater than* $\frac{100}{1+(m-1)e^{-\delta_1}}$ *or less than* $\frac{100}{1+e^{-\delta_2}}$*.*

3.4 A grammar for different robustness variants

We have seen three variants of robustness. One way to check these variants is to have separate encodings for each, i.e., change the neural network encoding manually and obtain different robustness algorithms (as done e.g., in [18,27]). But can we have a common technique that can handle all three? This leads us to two simple yet important observations: First, all three are defined by only changing the post-condition of the verification query. Second, after the approximation described, all three can be written as a Boolean combination of linear constraints. This motivates us to define a simple grammar on the post-conditions that unifies all the variants (similar to the standard *Satisfiability Modulo Linear Real Arithmetic* [40], but we use this grammar in the context of post-conditions). The following grammar describes the rules for the post-conditions considered in our analysis. *LE* denotes the linear expressions, and *LC* represents the constraints on *LE* defined using conditional operators. The final property is represented by *PC* (for PostCond), which denotes the Boolean combinations of *LC* and is essentially a quantifier-free linear arithmetic formula.

$$\begin{aligned}
&\text{LE} ::= c_1x_1 + c_2x_2 + \cdots + c_mx_m + b, \forall i \in [m], c_i, b \in \mathbb{R}\\
&\text{LC} ::= \text{LE} > 0 \mid \text{LE} \geq 0 \mid \text{LE} < 0 \mid \text{LE} \leq 0\\
&\text{PC} ::= \text{PC} \wedge \text{PC} \mid \text{PC} \vee \text{PC} \mid \text{LC} \mid \neg\text{LC}
\end{aligned}$$

We let $\mathcal{PC}$ denote the set of post-conditions generated using the above grammar. It is then easy to see that $Q, Q_{rel}, Q_{str}, Q_{sm}$ are all in $\mathcal{PC}$, i.e., derivable in this grammar. As we show next, the main reason to do so is that the post-condition can algorithmically be encoded into additional layers of the neural network, which allows us to use any state-of-the-art neural network verifier.

4 Encoding Mechanism via Additional Layers

In this section, we provide an encoding for all robustness variants definable by the grammar, such that they can be integrated in state-of-the-art robustness verification engines. We begin by noting that the International Verification of Neural Networks Competition (VNN-COMP) [17] standardizes both the neural network format (ONNX) and the property format (VNNLIB). All neural network verifiers participating in VNN-COMP take as input a neural network in ONNX format and a property file in VNNLIB format. Importantly, the VNNLIB format encodes the pre-condition P and the negation of the post-condition Q of the neural network N, expressed as an arbitrary Boolean combination of linear constraints, i.e., exactly in the grammar that we described in the previous section.

However, most state-of-the-art neural network verifiers are optimized for *simplified post-conditions*, typically involving only disjunctions or conjunctions of linear atoms over the outputs of the network. In what follows, we call a post-condition *simplified* if it is either atomic $y > 0$, $y \geq 0$, $y < 0$, or $y \leq 0$, where y is one of the outputs of N or conjunction/disjunction of the atomics.

In this section, we show how we can convert post-conditions into simplified ones *by just appending a few additional layers* to the neural network, without having to change the encoding within the verifiers. In doing so, we only add a linear number of neurons in the size of the post-condition and number of the layers added is at most the depth of the formula (represented as a Directed Acyclic Graph) in the post-condition. As we will see, while it is easy to do this for conjunctions, it is non-trivial when both conjunctions and disjunctions are present, and we achieve a solution with an approximation factor.

Translating Conjunctions. Translating conjunctions is relatively simple (as also done in [34]). Suppose we have a conjunction $\bigwedge_{i=0}^{n} LE_i \leq 0$, where nodes $y_1, \ldots, y_m$ represent the original network's output nodes and $LE_i = \sum_{j=1}^{m} w_{ij} y_j + b_j$, say. Then, we observe that defining a new output as $y = \sum_{i=0}^{n} \text{ReLU}(LE_i)$ can be encoded easily as an additional layer, as depicted in Figure 3(a). The nodes $LE_1, LE_2, \ldots, LE_n$ represents the actions of corresponding linear expressions, which are followed by ReLU nodes. We immediately obtain:

Lemma 1. $y \leq 0 \iff \bigwedge_{i=0}^{n} LE_i \leq 0$

Proof. ($\implies$) Since $y = \sum_{i=0}^{n} \text{ReLU}(LE_i)$, if $y \leq 0$ then $\forall i$, $\text{ReLU}(LE_i) = 0$, because ReLU's output is always ≥ 0. Therefore, $\forall i, LE_i \leq 0$ by the definition of ReLU. Therefore, $\bigwedge_{i=0}^{n} LE_i \leq 0$ holds. ($\Longleftarrow$)Since $\bigwedge_{i=0}^{n} LE_i \leq 0$, we conclude $\bigwedge_{i=0}^{n} \text{ReLU}(LE_i) = 0$. Therefore, $\sum_{i=0}^{n} \text{ReLU}(LE_i) = 0$. Since $y = \sum_{i=0}^{n} \text{ReLU}(LE_i)$, $y \leq 0$. □

The number of ReLU nodes added is just the number of clauses in the property, e.g., in Figure 3(a), there are a total of n clauses; therefore, n ReLU nodes are added. We can also deduce that $y > 0 \Leftrightarrow \bigvee_{i=0}^{n} LE_i > 0$, which represents the encoding of the disjunction.

The general translation. When we have both disjunctions and conjunctions in the post-condition, we cannot directly feed the output of conjunctions to the input of a disjunction. The reason is that in the case of conjunction, the condition $y = 0$ is interpreted as true, whereas $y > 0$ is interpreted as false, while for disjunction, the roles are reversed. A straightforward approach to handle this asymmetry is to convert the post-condition into Disjunctive Normal Form (DNF) and execute each clause in parallel. However, this method suffers from several drawbacks: (i) the conversion to DNF may cause exponential blowup, (ii) information derived from one clause cannot be exploited by another, and (iii) the underlying verifier codebase may still need modifications to accommodate each clause. Another way to encode disjunctions is by using products, however, this introduces highly non-linear constraints, which are very inefficient to handle. Although all three post-conditions above are expressed in DNF form, this is not always the case.

For instance, for the top-k properties introduced in Section 5 the post-conditions are not in DNF (also see Appendix C Equations 15 and 18)

Therefore, we take an alternate approach via a transformation $flip(b, v) = b - v$ of a signal v for some $b > 0$ (defined later), which allows us to feed output of disjunction to conjunction and vice-versa. Our translation will also introduce non-linear constraints but only as *Relu*'s which can be easily modeled in verifiers. In the following, let $\dagger \in \{\vee, \wedge\}$ and if $\dagger = \vee$, then $\bar{\dagger} = \wedge$, if $\dagger = \wedge$, then $\bar{\dagger} = \vee$. Wlog., we also assume that the post-condition is given in Negation Normal Form, with negations pushed inside the linear inequalities, with $\wedge$ and $\vee$ flattened (i.e., $\wedge$ and $\vee$ alternate when traversing the formula top-down). Then, $\dagger \in \{\vee, \wedge\}$, for any post-condition $Q \in \mathcal{PC}$ and a parameter $\eta \in \mathbb{R}$, we define the gadget $V(\dagger, Q, \eta)$ as an expression made up of $flip$s, $Relu$, and summation functions. Hence for every such expression we can build a circuit $\mathfrak{C}_{V(\dagger,Q,\eta)}$ that can be integrated with the Neural network using additional layers (for $flip$ we use -1 weight on edges, while $Relu$ and summation are standard operators in neural networks). Thus, given a set of input values to Q, $V(\dagger, Q, \eta)$ evaluates to a real output value y, satisfying some properties, as we shall describe below. Both V and its circuit are defined inductively based on the structure of the post-condition. For atomic formulas, we construct the gadgets V as given below, one for each value of $\dagger \in \{\wedge, \vee\}$:

$$V(\wedge, LE \leq 0, \eta) = LE + \eta \qquad V(\vee, LE \leq 0, \eta) = -LE + \eta \tag{8}$$

The following gives the gadget for $Q = Q_1 \dagger .. \dagger Q_k$ (where $\dagger \in \{\vee, \wedge\}$), whose circuit is pictorially illustrated in Figure 3(b).

$$V(\dagger, Q, \eta) = \sum\nolimits_{i=1}^{k} Relu(flip(b(\dagger, k, \eta), V(\bar{\dagger}, Q_i, \eta))) \tag{9}$$

$$\text{where,} b(\dagger, k, \eta) = \begin{cases} \eta(1 + 1/k) & \text{if } \dagger = \wedge \\ 2 * \eta & \text{if } \dagger = \vee \end{cases} \tag{10}$$

The main intuition of the above construction is that after crossing every disjunct/conjunct, the direction of the bounds flip, i.e., upper bound on an output becomes a lower bound and vice versa. This allows us to propagate bounds when the circuit has disjuncts and conjuncts. Before formalizing this intuition and proving the properties of the translation, we present in Figure 3(c), an illustrative example. In this example, the negation of the post-condition $\neg Q$ is $((y_1 + y_2 \leq 0 \wedge y_2 \leq 0) \vee (y_1 - y_3 \leq 0 \wedge y_3 \leq 2))$, with atomic formulae $y_1 + y_2 \leq 0$, $y_2 \leq 0$, and so on. The parameter η in the input constraint is set to 0.2 (user-defined). The part of circuit before the first dashed vertical line are the gadgets for atomic formulae. For instance, $y_1 + y_2 \leq 0$ is converted into $-y_1 - y_2 + 0.2$, which is represented by the red part of the circuit. Thus, the output at the red line satisfies $0.2 \leq -y_1 - y_2 + 0.2$ which is equivalent to $y_1 + y_2 \leq 0$. The reason to have this flipped representation is that $y_1 + y_2 \leq 0$ and $y_2 \leq 0$ are connected with conjunction, but the formulation of Eq 9 expects the atomic formulae to be connected with disjunction, thus we apply the disjunctive rule in right had side

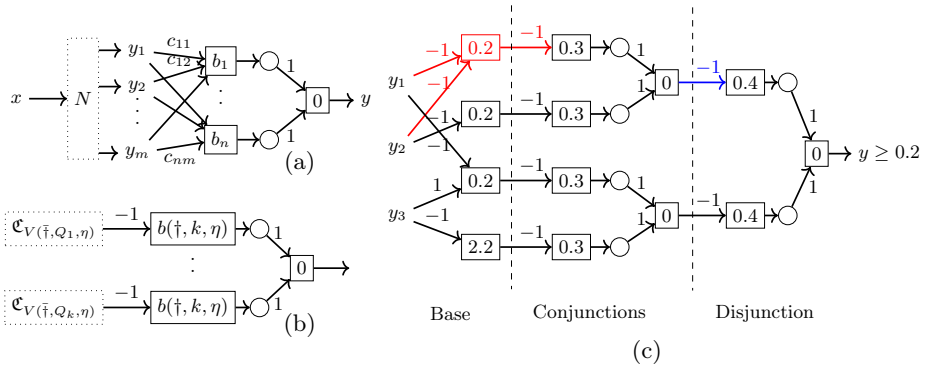


Fig. 3: (a) Neural network N appended with neural network layer that encodes either $\bigwedge_{i=0}^{n} LE_i \leq 0$ and its negation $\bigvee_{i=0}^{n} LE_i > 0$, where $LE_i = \sum_{j=1}^{m} c_{ij} y_j + b_i$. The circular nodes are ReLU and the square nodes nodes are linear combinations. (b) The circuit for $\mathfrak{C}_{V(\dagger,Q,\eta)}$ (c) Translation of post-condition $\neg((y_1 + y_2 \leq 0 \wedge y_2 \leq 0) \vee (y_1 - y_3 \leq 0 \wedge y_3 \leq 2))$ using our scheme and $\eta = 0.2$.

of of (8). The resulting outputs are connected with conjunctions, and the gadget between the two vertical lines corresponds to these conjunctions. Since this gadget represents conjunction, the value of b is determined by the conjunction case in Eq 10, i.e., $b = \eta\left(1 + \frac{1}{k}\right) = 0.2\left(1 + \frac{1}{2}\right) = 0.3$. In other words, at the blue line the output will be less than or equal to 0.2 if $(y_1 + y_2 \leq 0 \wedge y_2 \leq 0)$ (as shown more generally in Lemma 2 below). Finally, the conjunctions are connected with disjunctions, represented by the gadget after the second vertical line, which is also constructed following Eq 10, where $b = 2\eta = 0.2 \times 2 = 0.4$.

Our choice of $b(\dagger, k, \eta)$ is crucial for ensuring the desired properties of the gadget. The following technical lemma (with proof in Appendix D) establishes how can we propagate the bounds on the signal $V(\dagger, Q, \eta)$ to the sub-formulas Q_i and vice-versa, which will help us relating bounds on the inputs of the gadget and the bounds on the output of the gadget.

Lemma 2. *For $\eta > 0$ and $\beta > 0$, the following holds for Q.*

1. $\forall i.\ \eta \leq V(\vee, Q_i, \eta) \Rightarrow V(\wedge, Q, \eta) \leq \eta$.
2. $\exists i.\ V(\wedge, Q_i, \eta) \leq \eta \Rightarrow \eta \leq V(\vee, Q, \eta)$.
3. $V(\wedge, Q, \eta) \leq \beta \Rightarrow \forall i.\ (1 + 1/k)\eta - \beta \leq V(\vee, Q_i, \eta)$.
4. $\beta \leq V(\vee, Q, \eta) \Rightarrow \exists i.\ V(\wedge, Q_i, \eta) \leq 2\eta - \beta/k$.

Using the first two parts of the above lemma (the other two parts will be used for providing error bounds), we are now ready to state the lemma that establishes the correctness of our translation scheme. The first two cases demonstrate correctness for atomic formulae, while Cases 3 and 4 establish correctness for general formulae.

Lemma 3. *For a given post condition Q and an $\eta > 0$, the following holds*

1. If $Q = LE \leq 0$, $V(\wedge, Q, \eta) \leq \eta \Leftrightarrow Q$
2. If $Q = LE \leq 0$, $V(\vee, Q, \eta) \geq \eta \Leftrightarrow Q$
3. If Q is conjunctive, i.e., $Q = Q_1 \wedge ... \wedge Q_k$, then $Q \Rightarrow V(\wedge, Q, \eta) \leq \eta$
4. If Q is disjunctive, i.e., $Q = Q_1 \vee ... \wedge Q_k$, then $Q \Rightarrow V(\vee, Q, \eta) \geq \eta$

Proof. We prove this lemma by induction.

- **Base case:**
  1. If $Q = LE \leq 0$, then by definition, $V(\wedge, Q, \eta) = LE + \eta$. Given that $V(\wedge, Q, \eta) \leq \eta$, we derive: $LE + \eta \leq \eta \Leftrightarrow LE \leq 0$.
  2. If $Q = LE \leq 0$, then by definition, $V(\vee, Q, \eta) = -LE + \eta$. Given that $V(\vee, Q, \eta) \geq \eta$, we derive: $-LE + \eta \geq \eta \Leftrightarrow LE \leq 0$.
- **Inductive step:** Assume Lemma 3 holds for all formulae of depth d, denoted as Q^d. Consider a formula of depth $d+1$, given by: $Q^{d+1} = Q_1^d \dagger \cdots \dagger Q_k^d$.
  3. If $\dagger = \wedge$, Q_i^ds are true. Due to the induction hypothesis, $\forall i,\ \eta \leq V(\vee, Q_i^d, \eta)$. By the part 1 of Lemma 2, we have: $\forall i,\ \eta \leq V(\vee, Q_i^d, \eta) \implies V(\wedge, Q^{d+1}, \eta) \leq \eta$. Therefore, $V(\wedge, Q^{d+1}, \eta) \leq \eta$ holds.
  4. If $\dagger = \vee$, Q_i^d is true for some i. Due to the induction hypothesis, $\exists i,\ V(\wedge, Q_i^d, \eta) \leq \eta$. By the part 2 of Lemma 2, we have: $\exists i,\ V(\wedge, Q_i^d, \eta) \leq \eta \implies \eta \leq V(\vee, Q^{d+1}, \eta)$. Therefore, $\eta \leq V(\vee, Q^{d+1}, \eta)$ holds.

Using the above, we can finally define $N' = \mathit{Append}(N, V(\dagger, Q, \eta))$ to be the neural network obtained by attaching the circuit $\mathfrak{C}_{V(\dagger,Q,\eta)}$ after the neural network N with y being the final output.More precisely, the inputs of $V(\dagger, Q, \eta)$ are connected to the outputs of N and the output y of $V(\dagger, Q, \eta)$ is the only output of N'. Then, from the above lemma, we can prove our first theorem regarding using it as a verification query.

Theorem 4. *Consider neural network N, pre-condition P, a post-condition $\neg Q$, and $\eta > 0$. Let $N' = \mathit{Append}(N, V(\dagger, Q, \eta))$ be the neural network obtained by attaching $\mathfrak{C}_{V(\dagger,Q,\eta)}$ after neural network N and y be the final output.*

1. *If Q is disjunctive, $\langle N', P, y < \eta \rangle \Rightarrow \langle N, P, \neg Q \rangle$*
2. *If Q is conjunctive, $\langle N', P, y > \eta \rangle \Rightarrow \langle N, P, \neg Q \rangle$*

Proof. The following is the proof of the first part. Let us consider the successful query $\langle N', P, y < \eta \rangle$ to the solver. We know $\neg(y < \eta)$ is infeasible. Therefore, $\eta \leq V(\vee, Q, \eta)$ is infeasible. Therefore due to the contrapositive of the third part of Lemma 3, we conclude that post-condition Q cannot be satisfied. Therefore, query $\langle N, P, \neg Q \rangle$ holds. Similarly, we can prove the second part using the last part of Lemma 3.

Integrating All Components: Soundness. Now, we establish the soundness of our framework by integrating the two main components: confidence approximation (as defined in Section 3) and the encoding mechanism (as defined in this section). While our framework remains sound as long as the soundness conditions of the first component hold, we can show how soundness is preserved across all examples presented in Section 3.

Theorem 5. *Let N be a neural network, P a pre-condition, Q any of the three post-conditions specified in Section 3). Let N' be the appended neural network and Q' be the simplified post-condition obtained through confidence approximation. If the query $\langle N', P, Q' \rangle$ holds, then the original query $\langle N, P, Q \rangle$ also holds.*

Proof. Let Q'' denote an intermediate post-condition obtained after applying confidence approximations, but prior to full simplification and encoding. From Theorems 1, 2, and 3, we have: $\langle N, P, Q'' \rangle \implies \langle N, P, Q \rangle$. Furthermore, from Theorem 4, we have: $\langle N', P, Q' \rangle \implies \langle N, P, Q'' \rangle$. and hence together, we conclude $\langle N', P, Q' \rangle \implies \langle N, P, Q \rangle$.

Bound on the error in counterexamples. If the query $\langle N', P, y < \eta \rangle$ fails, the verifier generates a counterexample input x to N' that violates $y < \eta$. Since the neural network translation is approximate, x may not violate $\neg Q$ in N. Moreover, the following Theorem (with proof in Appendix D) uses Lemma 2 (3., 4.) to bound the error. Let $Q[\eta]$ be a formula obtained by replacing each $LE \leq 0$ by $LE \leq \eta$ in Q.

Theorem 6. *If $\eta \leq V(\vee, Q, \eta)$ is satisfiable and Q is DNF (disjunctive normal form), we can conclude that $Q[2\eta]$ is satisfiable.*

The above theorem indicates that the satisfying assignment may violate Q by the margin of 2η. We must choose η as small as possible but not smaller than the minimum precision of the underlying solver, which will lead to the numerical instability of the solver. The above theorem requires Q to be in DNF. However, we can prove a similar theorem if Q is in CNF (conjunctive normal form). Since all our properties are in CNF or DNF, we can use the above theorem to bound the error of the counterexample for the properties that we are interested in.

For the simplicity of the presentation, we only considered non-strict inequalities above. In Appendix E, we provide the versions of the theorems that handle strict inequalities.

5 Beyond the Confidence-based Properties

Top-k robustness: Till now, we have considered robustness variants involving confidence, now we show that even other variants can easily be encoded in our framework. We consider the notion of top-k robustness introduced in [35]. Let $N(x, k)$ denote the k-th highest logit value of the network output, and define $N^k(x) = \{i \mid N_i(x) \geq N(x, k)\}$, the set of classes with the top-k highest logits. Top-k robustness is then defined as $\forall x'\ \text{dist}(x, x') \leq \epsilon \implies N^k(x) = N^k(x')$. An example shown in Figure 6 in the Appendix. Its negation can be expressed as:

$$\exists x'\ dist(x, x') \leq \epsilon \wedge N^k(x) \neq N^k(x')$$

Letting $y_1, ..., y_m = N(x')$, we can capture this, i.e., $N^k(x) \neq N^k(x')$ constraint using $\bigvee_{i=0}^{m} \bigvee_{j \in N^k(x), j \neq i}^{m} y_i > y_j$. Which indeed is in our desired format of the grammar, with the total number of disjunctive clauses being $(m - k) \cdot k$.

The same paper [35] also defines a relaxed variant (*top-k-relaxed*) robustness: $\forall x'\ \text{dist}(x, x') \leq \epsilon \implies \exists k \leq K\ :\ N^k(x) = N^k(x')$. Another variant, *top-k-affinity*, permits the user to specify the set of classes in which misclassification is allowed. All three variants of top-k robustness are captured by our grammar, more details of which, due to lack of space, are presented in Appendix C. We will now present experimental results for this property as well as those introduced earlier in the next section.

6 Experiments

Our implementation encodes our post-conditions into additional neural network layers as described in the previous section and then invokes a Neural network verification engine. For experiments, we selected the state-of-the-art tool $\alpha\beta$-CROWN [11,41,42,39], a portfolio verifier, which has consistently ranked 1^{st} in VNN-COMP 2021-2024. Our experiments below are designed to address the following research questions: **RQ1:** Does our layer-based approach work/scale for different variants of robustness for *large* VNN-COMP benchmarks? **RQ2:** How does our approach compare across *varying thresholds* for the different robustness variants? **RQ3:** How does our approach compare wrt a direct encoding of properties in *a constraint-based state-of-the-art solver* e.g.,MARABOU? In addition, to show the empirical impact or meaningfulness of our different robustness variants (introduced in Section 3), other than the images in the Introduction, we present more instances in Figures 6, 7, 9, 10, 11, 12, and 14 in Appendix C and F.

Intuitively, RQ1 aims to demonstrate *scalability* for each of the (richer) properties we consider, in comparison to other tools. In RQ2, we study the effect of changing thresholds on performance. While it is expected that higher thresholds yield more "safe" cases (e.g., in relaxed robustness), this experiment serves as a sanity check to validate that the framework behaves consistently across variants and demonstrates flexibility in supporting different thresholds. RQ3 compares our encoding framework with direct encoding. We performed all experiments on a GPU machine: Tesla V100-SXM2-32GB, 9 vCPUs, and 32GB RAM, and set the value of η to 1×10^{-4}.

Benchmarks: We conducted experiments on four different datasets: MNIST [43], CIFAR-10 [20], Traffic Sign Recognition (TSR) [44], and IMAGENET [45]. All networks used in the experiments, along with their properties (VNNLIB files), were taken from VNN-COMP 2021 to VNN-COMP 2024. The provided VNNLIB files correspond to standard robustness properties. We have listed our benchmarks in Table 1. Our experiments covered a diverse set of benchmarks, ranging from fully connected networks to complex architectures, including convolutional layers, max-pooling layers, residual blocks, and ReLU activations, including standard and adversarially trained models. *Additionally, the network sizes ranged from small architectures with* 512 *ReLUs to large networks with* 11.16M *ReLUs. Each neural network, along with its corresponding property (VNNLIB) file, represents a single benchmark. In total, we evaluated 8,870 benchmarks in our experiments.* More details are provided in Appendix F.1.

Table 1: Networks details

Category	**Network name**	#layers	#activation units	adv trained
MNIST	mnist-net-256×2.onnx	2-FC	0.51K	No
	mnist-net-256×4.onnx	4-FC	1.02K	No
	mnist-net-256×6.onnx	6-FC	1.54K	No
CIFAR-10	cifar-base-kw.onnx	2-Conv, 2-FC	3.17K	Yes
	cifar-deep-kw.onnx	4-Conv, 2-FC	6.77K	Yes
	cifar-wide-kw.onnx	2-Conv, 2-FC	6.24K	Yes
	cifar10-2-255.onnx	3-Conv, 2-FC	49.15K	Yes
	cifar10-8-255.onnx	2-Conv, 2-FC	16.39K	Yes
	convBigRELU-PGD.onnx	4-Conv, 3-FC	62.46K	Yes
	resnet-2b.onnx	2-res-blocks (5-Conv, 2-FC)	6.24K	Yes
	resnet-4b.onnx	4-res-blocks (9-Conv, 2-FC)	14.45K	Yes
GTSRB	net-1	2-QConv, 1-FC	-	No
	net-2	3-QConv, 3-BN, 2-Maxpool, 2-FC	-	No
	net-3	3-QConv, , 3-BN, 3-Maxpool, 2-FC	-	No
IMAGENET	vggnet-16	13-Conv, 5-Maxpool, 3-FC	13.16M	No

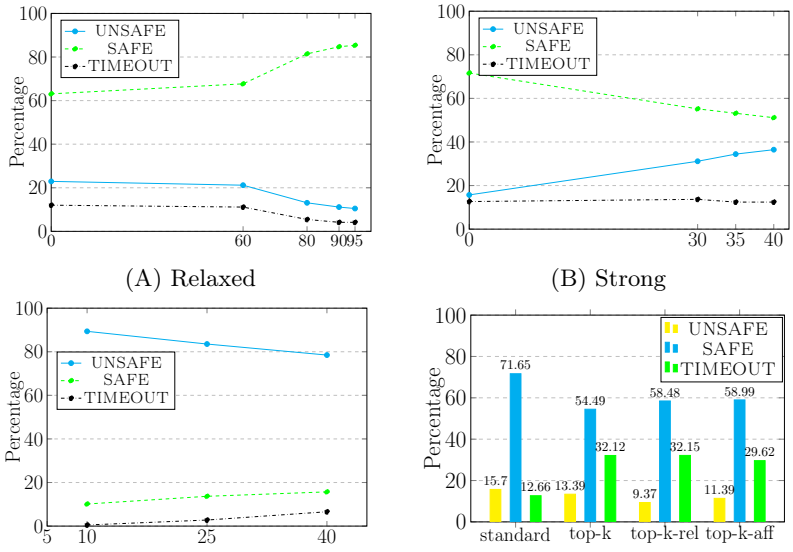
Evaluating robustness variants and comparing across thresholds (RQ1, RQ2): We conducted experiments on different robustness properties, as explained in Section 3. We present a summary of results below, with a detailed analysis for each dataset is provided in Appendix F.2.

Relaxed robustness: As defined in Eq 3 of Section 3, we need a user defined confidence level τ to analyse this property. We took five different confidence thresholds $\tau \in \{0, 60, 80, 90, 95\}$, where $\tau = 0$ corresponds to standard robustness. For a given threshold value τ, a result of SAFE indicates either no misclassification or a misclassification with confidence below $\tau\%$, whereas a result of UNSAFE indicates a counterexample with confidence above $\tau\%$.

The results in Figure 4A show that as we increase the confidence level, the safe (verified) cases increase and unsafe (counterexamples) decrease. This is because increasing confidence means we report counterexamples only if it has confidence greater than the threshold τ, which means less numbers of counterexamples. Also, TIMEOUT cases are decreasing, as increasing confidence reduces search space, and most of the cases are verified by the abstraction-based module like CROWN [39] and MILP based bound tightening, within $\alpha\beta$-CROWN.

Strong robustness: We took the threshold value τ_1 as 57.5 for the seed image in Eq 5, because it is the average value of all the confidences on the seed images of the properties. We took three different values of the confidences thresholds $\tau_2 = 30, 35, 40$. Intuitively we want our model to be robust against the misclassification, at the same time confidence should not go below τ_2. As shown in Figure 4B, the counterexamples increase as we increase the confidence τ_2, because higher the threshold means higher chances of confidence to fall below the threshold, with almost no increase in time taken.

Smoothness: We considered varying threshold values of 10, 25, and 40 for smoothness. Intuitively, a threshold of 10 means that if, within an ϵ perturbation, the confidence of the seed image's class fluctuates within ± 10 of its original confidence, the smoothness property holds (SAFE); otherwise, it does not hold (UNSAFE). Figure 4C illustrates how smoothness changes as the threshold varies.



(A) Relaxed (B) Strong

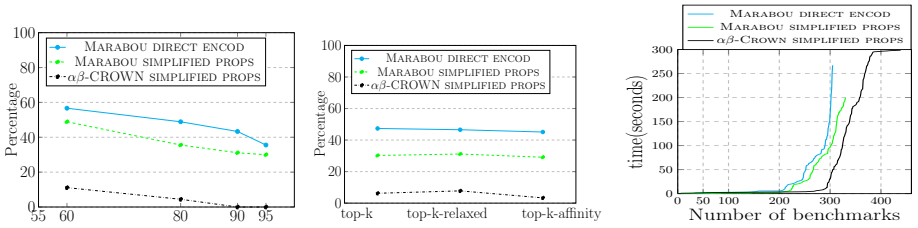
(C) Smoothness (D) Top-k

Fig. 4: Figures 4A, 4B, and 4C show the confidence thresholds on the x-axis and the percentage of SAFE, UNSAFE, and TIMEOUT instances on the y-axis. Figure 4D presents a comparison between standard robustness and top-k robustness, including top-k relaxed robustness and top-k affinity robustness. For each robustness metric, the left/middle/right bars represent the percentage of UNSAFE, SAFE, and TIMEOUT cases, respectively.

Increasing the smoothness threshold allows for greater variations in the output, resulting in more SAFE cases and fewer UNSAFE cases. However, the number of timeout cases slightly increases. A possible reason is that a lower threshold implies even slight changes in output confidence can lead to UNSAFE cases.

Top-k, top-k-relaxed, and top-k-affinity robustness: In both variations of top-k robustness, no threshold is involved, so the number of benchmarks remains the same as in standard robustness. We observed in previous experiments that for the GTSRB dataset properties, the confidence values for all seed images are 100%, meaning the confidence for all other classes is 0%. As a result, top-k properties cannot be applied to these benchmarks. More details are provided in Appendix F.2. Figure 4D compares standard robustness with top-k, top-k relaxed, and top-k affinity robustness. The number of UNSAFE cases decreases from standard robustness to the other three robustness definitions.

Comparing with a direct encoding (RQ3): To compare our layer-based encoding with a direct encoding, for each of the specific richer properties considered, we encoded them directly with the constraint-based solver MARABOU using its Python interface. We compared this with our layer-based encoding on MARABOU (i.e., an appended network with a simplified property) as well as $\alpha\beta$-CROWN. Since MARABOU is a CPU-based solver [17], we conducted the experiments on an Intel(R) Xeon(R) Gold 6314U CPU @ 2.30GHz with 64GB RAM machine. We observed that MARABOU ran out of memory for many CIFAR-10



(A) Relaxed robustness (% of timeout cases vs. confidence thresholds)

(B) Top-k,top-k-relaxed,and top-k-affinity robustness

(C) Cactus plots for only solved benchmarks

Fig. 5: (A-B)Comparison of the constraint-based solver MARABOU with and without the simplified property, alongside $\alpha\beta$-CROWN with the simplified property. (C) the x-axis shows the number of benchmarks solved, ordered by increasing solving time, and the y-axis shows the time taken to solve them.

benchmarks, so we restricted this comparison to MNIST benchmarks. Figure 5A presents the comparison with the relaxed robustness property, using the same confidence threshold as in previous experiments. In Figure 5B, we also compare the performance with respect to top-k relaxed and top-k affinity robustness. Comparisons with strong robustness and smoothness properties are provided in the Appendix F.3. Figure 5C shows the cactus plot of the comparison. Our results indicate that $\alpha\beta$-CROWN using our layer-based encoding, outperforms MARABOU, both with the direct encoding and with the layer-based one. This is perhaps not surprising, as $\alpha\beta$-CROWN is a portfolio tool incorporating efficient techniques such as PGD-attack [31] and CROWN [11]. However, as an ablation study, we can see that MARABOU with our layer-encoding does perform better than MARABOU without it (i.e., with the direct encoding). *This comparison demonstrates that our framework not only enables the encoding of richer properties but also is efficient compared to directly encoding the properties.*

7 Conclusion

In this paper, we considered different variants of robustness in neural networks that incorporate SOFTMAX-based confidence values. We introduced a grammar of post-conditions that captures all these variants and provided a unifying framework to transform them into a gadget that can be appended to an existing neural network, thereby simplifying arbitrary properties. This enables the use of verifiers such as $\alpha\beta$-CROWN. Our experiments demonstrate that our approach is more efficient than directly encoding the properties as constraints into state-of-the-art constraint-based solvers.

References

1. Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai

Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. volume abs/1604.07316, 2016.
2. Filippo Amato, Alberto López, Eladia María Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis. volume 11, pages 47–58. Elsevier, 2013.
3. Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.
4. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
5. Isaac Dunn, Hadrien Pouget, Daniel Kroening, and Tom Melham. Exposing previously undetectable faults in deep neural networks. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 56–66, 2021.
6. Jamie Hayes and George Danezis. Learning universal adversarial perturbations with generative models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 43–49. IEEE, 2018.
7. Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
8. Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, volume 1, page 3, 2018.
9. Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin 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.
10. Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018.
11. Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.
12. Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.
13. Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.
14. Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International conference on computer aided verification*, pages 3–29. Springer, 2017.
15. Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

16. Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *CoRR*, abs/2011.13824, 2020.
17. Christopher Brix, Stanley Bak, Taylor T Johnson, and Haoze Wu. The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results. *arXiv preprint arXiv:2412.19985*, 2024.
18. Marco Casadio, Ekaterina Komendantskaya, Matthew L Daggitt, Wen Kokke, Guy Katz, Guy Amir, and Idan Refaeli. Neural network robustness as a verification property: a principled case study. In *International conference on computer aided verification*, pages 219–231. Springer, 2022.
19. Goodfellow Ia. Deep learning/ian goodfellow, yoshua bengio and aaron courville, 2016.
20. Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
21. Blerta Lindqvist. A novel method for function smoothness in neural networks. *IEEE Access*, 10:75354–75364, 2022.
22. Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. A case for new neural network smoothness constraints. 2020.
23. Zi Wang, Gautam Prakriya, and Somesh Jha. A quantitative geometric approach to neural-network smoothness. *Advances in Neural Information Processing Systems*, 35:34201–34215, 2022.
24. alpha-beta-crown. `https://github.com/Verified-Intelligence/alpha-beta-CROWN`, 2025. Accessed: 2025-06-21.
25. Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, et al. Marabou 2.0: a versatile formal analyzer of neural networks. In *International Conference on Computer Aided Verification*, pages 249–264. Springer, 2024.
26. Klas Leino and Matt Fredrikson. Relaxing local robustness. *Advances in Neural Information Processing Systems*, 34:17072–17083, 2021.
27. Anagha Athavale, Ezio Bartocci, Maria Christakis, Matteo Maffei, Dejan Nickovic, and Georg Weissenbacher. Verifying global two-safety properties in neural networks with confidence. In *International Conference on Computer Aided Verification*, pages 329–351. Springer, 2024.
28. Dennis Wei, Haoze Wu, Min Wu, Pin-Yu Chen, Clark Barrett, and Eitan Farchi. Convex bounds on the softmax function with applications to robustness verification. In *International Conference on Artificial Intelligence and Statistics*, pages 6853–6878. PMLR, 2023.
29. Matt Jordan and Alexandros G Dimakis. Exactly computing the local lipschitz constant of relu networks. *Advances in Neural Information Processing Systems*, 33:7344–7353, 2020.
30. Nikhil Naik and Pierluigi Nuzzo. Robustness contracts for scalable verification of neural network-enabled cyber-physical systems. In *2020 18th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pages 1–12. IEEE, 2020.
31. Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.
32. Duo Zhou, Christopher Brix, Grani A Hanasusanto, and Huan Zhang. Scalable neural network verification with branch-and-bound inferred cutting planes. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

33. Eleonora Giunchiglia, Alex Tatomir, Mihaela Cătălina Stoian, and Thomas Lukasiewicz. Ccn+: A neuro-symbolic framework for deep learning with requirements. *International Journal of Approximate Reasoning*, 171:109124, 2024. Synergies between Machine Learning and Reasoning.
34. David Shriver, Sebastian Elbaum, and Matthew B Dwyer. Dnnv: A framework for deep neural network verification. In *International Conference on Computer Aided Verification*, pages 137–150. Springer, 2021.
35. Klas Leino and Matt Fredrikson. Relaxing local robustness. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17072–17083. Curran Associates, Inc., 2021.
36. Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
37. Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.
38. Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin Vechev. Beyond the single neuron convex barrier for neural network certification. *Advances in Neural Information Processing Systems*, 32, 2019.
39. Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021.
40. Daniel Kroening and Ofer Strichman. *Decision procedures*, volume 5. Springer, 2008.
41. Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.
42. Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32:9835–9846, 2019.
43. Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.
44. Andreea Postovan and Mădălina Eraşcu. Architecturing binarized neural networks for traffic sign recognition. *arXiv preprint arXiv:2303.15005*, 2023.
45. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
46. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
47. Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.
48. Mislav Balunovic and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2020.
49. Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

50. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
51. Bob Bixby. The gurobi optimizer. *Transp. Re-search Part B*, 41(2):159–178, 2007.

Appendix

A Softmax Approximation

We present a theoretical comparison between our softmax approximation and the approximation introduced in [27]. Below is the definition of the softmax function. We omit the factor of 100 from the equation, as it was used solely to convert probabilities into percentages:

$$\text{Softmax}(y_1, \ldots, y_m, c) = \frac{e^{y_c}}{\sum_{i=1}^{m} e^{y_i}}.$$

The paper [27] introduced lower and upper bounds for the softmax function as follows. Let c be the index of the maximum value, i.e., $c = \arg\max_{i=1}^{m} y_i$:

$$\text{Sig}(y_c - \max_{i=1, i\neq c}^{m}(y_i) - \log(m-1)) \leq \text{Softmax}(y_1, \ldots, y_m, c) \leq \text{Sig}(y_c - \max_{i=1, i\neq c}^{m}(y_i)) \tag{11}$$

where $\text{Sig}(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and max denotes the usual maximum operation.

For analysis, consider the lower bound:

$$\begin{aligned} lb &= \text{Sig}(y_c - \max_{i=1, i\neq c}^{m}(y_i) - \log(m-1)) \\ &= \frac{1}{1 + e^{-y_c + \max_{i=1, i\neq c}^{m}(y_i) + \log(m-1)}} \\ &= \frac{1}{1 + e^{-y_c + \max_{i=1, i\neq c}^{m}(y_i)} e^{\log(m-1)}} \\ &= \frac{1}{1 + (m-1)e^{-y_c + \max_{i=1, i\neq c}^{m}(y_i)}} \\ &= \frac{1}{1 + (m-1)e^{-(y_{\max} - y_{\text{smax}})}}. \end{aligned}$$

Here, we define

$$\begin{aligned} y_{\max} &= \max_{i=1}^{m} y_i, \\ y_{\text{smax}} &= \max_{i=1, i\neq \max}^{m} y_i. \end{aligned}$$

Intuitively, $y_{\max}$ and y_{smax} represent the maximum and second maximum values of y_i, respectively.

We define the lower bound from Section 3 as follow:

$$lb' = \frac{1}{1+e^{-\delta}},$$

where $\delta = -\log\left(\frac{1}{th} - 1\right)$ and τ is a user-defined confidence threshold.

The key difference between our analysis and the one presented in [27] is that our approach incorporates a user-defined confidence threshold, whereas the previous approximation is based solely on the input to softmax variables y_i.

Now, let us determine when our lower bound approximation is tighter than the one from [27]:

$$\begin{aligned} lb' &> lb \\ \delta &> y_{\max} - y_{\text{smax}} \\ -\log\left(\frac{1}{th} - 1\right) &> y_{\max} - y_{\text{smax}} \\ \frac{1}{th} - 1 &< e^{-y_{\max}+y_{\text{smax}}} \\ \tau &> \frac{1}{1+e^{-y_{\max}+y_{\text{smax}}}}. \end{aligned} \tag{12}$$

Eq 12 establishes the condition under which our lower bound for the SOFTMAX function is tighter. Applying a similar analysis for the upper bound, we find that

$$\tau < \frac{1}{1+e^{-y_{\max}+y_{\text{smax}}}}. \tag{13}$$

Thus, depending on whether we are bounding the upper or lower limit, we can appropriately select τ to achieve a tighter approximation.

B Proof from Section 3

Proof (of Claim 3 part 1). The following proof is similar to the proof for Claim 2. Let LHS of the claim be true. Since $y_{t'} = \max_{i=1, i\neq t}^{m}(y_i)$, we can derive $\textsc{Conf}(y_1, ...y_m, t) \geq \frac{100e^{y_t}}{e^{y_t}+(m-1)e^{y_{t'}}}$, replacing e^i by $e^{t'}$ for all $i, i \neq t$. After scaling the fraction in the RHS by e^{y_t}, we obtain $\textsc{Conf}(y_1, ...y_m, t) \geq \frac{100}{1+(m-1)e^{y_{t'}-y_t}}$. Since $y_t \geq y_{t'} + \delta$, we obtain $\textsc{Conf}(y_1, ...y_m, t) \geq \frac{100}{1+(m-1)e^{-\delta}}$. Since $\delta = -\ln(\frac{1}{m-1}(\frac{100}{\tau_2} - 1))$, $\textsc{Conf}(y_1, ...y_m, t) > \tau_2$.

Proof (of Claim 3 part 2). The following proof is similar to the proof for Claim 1. Assume LHS of the claim holds, i.e., $y_t \leq y_{t'} + \delta$. By removing all exponential terms from the denominator except for e^{y_t} and $e^{y_{t'}}$, we obtain $\textsc{Conf}(y_1, ...y_m, t) \leq \frac{100e^{y_t}}{e^{y_t}+e^{y_{t'}}}$. After scaling the right-hand side by $\frac{1}{e^{y_t}}$, we get $\textsc{Conf}(y_1, ...y_m, t) \leq \frac{100}{1+e^{-(y_t-y_{t'})}}$. Since $\delta \geq y_t - y_{t'}$, we have $\textsc{Conf}(y_1, ...y_m, t) \leq \frac{100}{1+e^{-\delta}}$.

Proof (of Claim 4 part 1). Assume LHS of the claim holds:

1. $y_t < y_{t'} + \delta_1$ holds, proof is similar to the Claim 1, except in Claim 1 y_t is maximum, which is not ncessesary here. By removing all exponential terms from the denominator except for e^{y_t} and $e^{y_{t'}}$, we obtain $\textsc{Conf}(y_1, ...y_m, t) \leq \frac{100e^{y_t}}{e^{y_t}+e^{y_{t'}}}$. After scaling the right-hand side by $\frac{1}{e^{y_t}}$, we get $\textsc{Conf}(y_1, ...y_m, t) \leq \frac{100}{1+e^{-(y_t-y_{t'})}}$. Since $\delta_1 > y_t - y_{t'}$, we have $\textsc{Conf}(y_1, ...y_m, t) < \frac{100}{1+e^{-\delta_1}}$. Since $\delta_1 = -\ln(\frac{100}{\tau_1} - 1)$, $\textsc{Conf}(N(x'), t) < \tau_1$.
2. $y_t > y_{t'} + \delta_2$ holds, by Claim 3 part 1, we conclude that $\textsc{Conf}(N(x'), t) > \tau_2$.

By the above two cases we conclude that $\textsc{Conf}(N(x'), t) < \tau_1 \wedge \textsc{Conf}(N(x'), t) > \tau_2$ holds.

Proof (of Claim 4 part 2). Assume LHS of the claim holds:

1. $y_t \geq y_{t'} + \delta_1$ holds, by Claim 2, we conclude that $\textsc{Conf}(N(x'), t) \geq \frac{100}{1+(m-1)e^{-\delta_1}}$ holds.
2. $y_t \leq y_{t'} + \delta_2$ holds, by Claim 3 part 2, we conclude that $\textsc{Conf}(N(x'), t) \leq \frac{100}{1+e^{-\delta_2}}$

By the above two cases we conclude that $\textsc{Conf}(N(x'), t) \geq \frac{100}{1+(m-1)e^{-\delta_1}}$ or $\textsc{Conf}(N(x'), t) \leq \frac{100}{1+e^{-\delta_2}}$.

Proof (of Theorem 3).

Let $y_t < y_{t'} + \delta_1 \wedge y_t > y_{t'} + \delta_2$ is our derived post-condition Q'_{sm}. Eq 7 shows the encoding of the $\neg Q'_{sm}$. First part of the equation encodes $y_t \geq y_{t'} + \delta_1$, since $y_{t'} = \max_{i=1, i\neq t}^{m}(y_i)$, which is equivalent to checking $\forall i, i \neq t, y_t \geq y_i + \delta_1$, which is first part of the equation. The second part of the equation encodes $y_t \leq y_{t'} + \delta_2$, which is equivalent to checking $\exists i, i \neq t, y_t \leq y_i + \delta_2$.

1. If the query holds, then by Claim 4 part 1 and the above explanation of Eq 7, we can conclude $\textsc{Conf}(N(x'), t) < \tau_1 \wedge \textsc{Conf}(N(x'), t) > \tau_2$. By the above discussion $\tau_1 = C + \tau$ and $\tau_2 = C - \tau$, it follows that $\textsc{Conf}(N(x'), t) < C + \tau \wedge \textsc{Conf}(N(x'), t) > C - \tau$, which implies $|C - \textsc{Conf}(N(x'), t)| < \tau$.
2. If the query does not hold, then by Claim 4 part 2 and the above explanation of Eq 7, we conclude that $\textsc{Conf}(N(x'), t) \geq \frac{100}{1+(m-1)e^{-\delta_1}}$ or $\textsc{Conf}(N(x'), t) \leq \frac{100}{1+e^{-\delta_2}}$.

C Top-k Robustness

We consider the notion of top-k robustness introduced in [35]. Suppose a function N computes the logits values at the neural network's output layer, where $N_i(x)$ represents the value at the i^{th} dimension. Let $N(x, k)$ denote the k^{th} highest value of the logits. The function $N^k(x) = \{i \mid N_i(x) \geq N(x, k)\}$ intuitively represents the set of classes with the top k highest values of the logits. Top-k robustness is then defined as:

$$\forall x' \; dist(x, x') \leq \epsilon \implies N^k(x) = N^k(x')$$

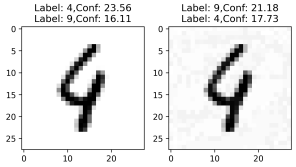


Fig. 6: **Top-k and Top-k-relaxed**: The image on the left, labeled as 4, is taken from the MNIST dataset and correctly classified with a confidence of 23.56% by the neural network `mnist-net-256x6.onnx`. The tool $\alpha\beta$-CROWN finds a counterexample (right), misclassified as label 9, within an input perturbation of 0.05.The counterexample has a confidence of 21.18%. The top-k property allows misclassification within the top-k predictions of the original image. For $k = 2$, the original image has top-2 predictions: labels 4 and 9 (corresponding to the highest and second-highest confidence scores). This example is top-2 as well as top-2-relaxed robust but does not satisfy standard robustness or strong robustness criteria.

See Figure 6 for an example. Its negation can be expressed as:

$$\exists x' \; dist(x, x') \leq \epsilon \wedge N^k(x) \neq N^k(x')$$

Letting $y_1, ..., y_m = N(x')$, we can capture this, i.e., $N^k(x) \neq N^k(x')$ constraint using

$$\bigvee_{i=0}^{m} \bigvee_{j \in N^k(x), j \neq i}^{m} y_i > y_j \tag{14}$$

which indeed is in our desired format format for disjunction in the grammar, with the total number of disjunctive clauses is $(m - k) \cdot k$.

Theorem 7. *Eq* (14) $\iff N^k(x) \neq N^k(x')$

Proof. ($\Longrightarrow$) If Eq 14 holds, for a fixed k there exist indices i and j with $i \notin N^k(x)$ and $j \in N^k(x)$ such that $y_i > y_j$. By defintion of $N^k(x)$, $i \in N^k(x')$, but $i \notin N^k(x)$ this means $N^k(x') \neq N^k(x)$.

($\Longleftarrow$) If for a fixed k, $N^k(x) \neq N^k(x')$ holds and $|N^k(x)| = |N^k(x')|$ then there must exist an index i such that $i \in N^k(x')$ but $i \notin N^k(x)$. Consequently, there also exists an index $j \in N^k(x)$ such that $j \notin N^k(x')$. This implies that j is replaced by i in $N^k(x')$, which further implies that $y_i > y_j$. Which is Eq 14.

C.1 Top-k Relaxed Robustness:

Further, in the same paper [35], a relaxed variant of top-K robustness was also defined as

$$\forall x' \; dist(x, x') \leq \epsilon \implies \exists k \leq K : N^k(x) = N^k(x')$$

The negation of the above post-condition is:

$$\bigwedge_{k=1}^{K} N^k(x) \neq N^k(x')$$

Using the above, the full negation of the post-condition is expressed as:

$$\bigwedge_{k=1}^{K} \left(\bigvee_{i=1, j \in N^k(x), j \neq i}^{m} y_i > y_j \right) \tag{15}$$

The above constraints are in Conjunctive Normal Form (CNF) and hence expressible in our grammar and encodable using the procedure that we explain in Section 4.

Theorem 8. *Eq 15* $\iff \bigwedge_{k=1}^{K} N^k(x) \neq N^k(x')$

Proof. ($\Longrightarrow$) If Eq 15 holds, then for every $k \in K$ there exist indices i and j with $i \notin N^k(x)$ and $j \in N^k(x)$ such that $y_i > y_j$. By defintion of $N^k(x)$, $i \in N^k(x')$, but $i \notin N^k(x)$ this means $N^k(x') \neq N^k(x)$ for all $k \in K$.

($\Longleftarrow$) If $\bigwedge_{k=1}^{K} N^k(x) \neq N^k(x')$ holds and $|N^k(x)| = |N^k(x')|$ then for each k there must exist an index i such that $i \in N^k(x')$ but $i \notin N^k(x)$. Consequently, there also exists an index $j \in N^k(x)$ such that $j \notin N^k(x')$. This implies that j is replaced by i in $N^k(x')$, which further implies that $y_i > y_j$. Which is Eq 15.

Theorem 15 establishes the equivalence between the encoding constraints in Eq 15 and the negation of the post-condition $\bigwedge_{k=1}^{K} N^k(x) \neq N^k(x')$ for relaxed Top-k robustness. This equivalence confirms the correctness of our encoding.

C.2 Top-k-affinity Robustness:

The paper [35] also introduced the concept of affinity robustness. The intuition behind this robustness is to incorporate expert knowledge into robustness evaluation. Experts provide sets of similar categories, and the network is considered robust as long as misclassifications remain within these predefined sets. The key idea is to allow misclassification within similar categories but not across completely different ones.

For instance, if an input image of a pine tree is misclassified as a palm tree, this might be acceptable. However, it should not be misclassified as a mammal or another unrelated category. Similarly, a tiger could be misclassified as a leopard but not as an elephant. Suppose experts define these sets as $\mathbb{S}$, where each set $S \in \mathbb{S}$ represents a collection of classes within which misclassification is allowed.

The definition of affinity robustness is given as:

$$\begin{aligned} &\forall x' \; dist(x, x') \leq \epsilon \implies \\ &\quad \exists k \leq K \; S \in \mathbb{S} : N^k(x) = N^k(x') \wedge N^k(x) \subseteq S \end{aligned} \tag{16}$$

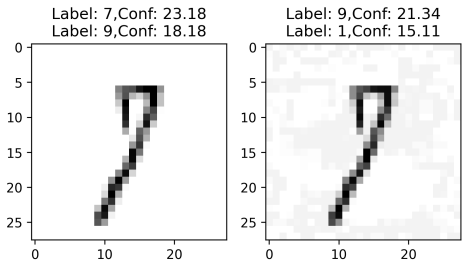


Fig. 7: **Top-k-affinity Robustness**: The image on the left, labeled as 7, is taken from the MNIST dataset and correctly classified with a confidence of 23.18% by the neural network `mnist-net-256x6.onnx`. A state-of-the-art verification tool, $\alpha\beta$-CROWN, finds a counterexample (right), misclassified as label 9, within an input perturbation of 0.05 (consistent with the perturbation used in VNN-COMP). The counterexample has a confidence of 21.34%. In affinity robustness, the user provides prior knowledge about acceptable misclassifications. Suppose the user specifies the following affinity groups: $\{\{1,7,9\},\{2,7\},\{0,8\},\{4,8\},\{3,5\},\{6\}\}$. This means, a class-7 image may be misclassified as 1, 2, or 9. A class-0 image may only be misclassified as 8. And a class-6 image must not be misclassified at all. According to this specified misclassification knowledge, the above example is affinity robust as well as top-k relaxed robust. However, the images in Figure 6 are not affinity robust under the same misclassification constraints because 4 is not allowed to misclassified in class 9. Additionally, the above image does not satisfy standard or strong robustness criteria.

The negation of the post-condition can be written as follow:

$$\bigwedge_{k=1}^{K} \bigwedge_{S\in\mathbb{S}} \left(N^k(x) \neq N^k(x') \vee N^k(x) \nsubseteq S\right). \tag{17}$$

For a given image x, the condition $N^k(x) \nsubseteq S$ can be determined beforehand. For all paired $\langle k, S\rangle$, if $N^k(x) \nsubseteq S$ is satisfied, the entire constraint is satisfied due to the disjunction. Therefore, we only need to construct constraints for those pairs $\langle k, S\rangle$ where $N^k(x) \subseteq S$ holds. Let K' and S' denote the sets of pairs $\langle k, S\rangle$ for which $N^k(x) \subseteq S$ is satisfied. The new constraints for the post-condition can be written as follows:

$$\bigwedge_{\langle k,S\rangle\in(K',S')} N^k(x) \neq N^k(x').$$

Using equation above, this can be rewritten as:

$$\bigwedge_{\langle k,S\rangle\in(K',S')}\left(\bigvee_{i=1,j\in N^k(x),j\neq i}^{m} y_i \geq y_j\right) \tag{18}$$

The above constraints are expressed in conjunctive normal form (CNF) and can be encoded into a neural network using the procedure described in Section 4.

Theorem 9. *Eq 18* $\iff \bigwedge_{\langle k,S\rangle\in(K',S')} N^k(x) \neq N^k(x')$

Proof. ($\Longrightarrow$) If Eq 18 holds, then for every $k, S \in (K', S')$ there exist indices i and j with $i \notin N^k(x)$ and $j \in N^k(x)$ such that $y_i > y_j$. By defintion of $N^k(x)$, $i \in N^k(x')$, but $i \notin N^k(x)$ this means $N^k(x') \neq N^k(x)$ for all $k, S \in (K', S')$.

($\Longleftarrow$) If $\bigwedge_{\langle k,S\rangle\in(K',S')} N^k(x) \neq N^k(x')$ holds and $|N^k(x)| = |N^k(x')|$ then for each $k, S \in (K', S')$ there must exist an index i such that $i \in N^k(x')$ but $i \notin N^k(x)$. Consequently, there also exists an index $j \in N^k(x)$ such that $j \notin N^k(x')$. This implies that j is replaced by i in $N^k(x')$, which further implies that $y_i > y_j$ for each $k, S \in (K', S')$. Which is Eq 18.

D Section 4: Proofs of Correctness

Proof (Proof for Lemma 2).

The following is the proof of the four parts.

1. We assume for each i, $\eta \leq V(\vee, Q_i, \eta)$. After a rewrite, $(1+1/k)\eta - V(\vee, Q_i, \eta) \leq \eta/k$. After applying definition of b and $flip$, $flip(b(\wedge, k, \eta), V(\vee, Q_i, \eta)) \leq \eta/k$ for each i. Due to the definition of V, we conclude $V(\wedge, Q, \eta) \leq \eta$.
2. We assume for some i, $V(\wedge, Q_i, \eta) \leq \eta$. After a rewrite, $\eta \leq 2\eta - V(\vee, Q_i, \eta)$. After applying definition of b and $flip$, $\eta \leq flip(b(\vee, k, \eta), V(\wedge, Q_i, \eta))$ for some i. Due to the definition of V, we conclude $\eta \leq V(\wedge, Q, \eta)$.
3. If $V(\wedge, Q, \eta) \leq \beta$, we conclude $flip(b(\wedge, k, \eta), V(\vee, Q_i, \eta)) \leq \beta$. After applying definition of b, $flip((1 + 1/k)\eta, V(\vee, Q_i, \eta)) \leq \beta$ for each i. After expanding definition of $flip$, $(1 + 1/k)\eta - V(\vee, Q_i, \eta) \leq \beta$. After simplification, $(1 + 1/k)\eta - \beta \leq V(\vee, Q_i, \eta)$.
4. If $\beta \leq V(\vee, Q, \eta)$, we conclude $\beta/k \leq flip(b(\wedge, k, \eta), V(\wedge, Q_i, \eta))$ for some i. After applying definition of b, $\beta/k \leq flip(2\eta, V(\wedge, Q_i, \eta))$. After expanding definition of $flip$, $\beta/k \leq 2\eta - V(\wedge, Q_i, \eta)$. After simplification, $V(\wedge, Q_i, \eta) \leq 2\eta - \beta/k$.

Proof (Proof for Theorem 6). Since Q is DNF, $Q = Q_1 \vee ... \vee Q_k$. Due to part four of Lemma 2, there is $V(\wedge, Q_i, \eta) \leq 2\eta - \eta/k = (2 - 1/k)\eta$. Let $Q_i = LE_1 \leq 0 \wedge ... \wedge LE_l \leq 0$. Due to part three of Lemma 2 for each LE_i, $(1+1/l)\eta - (2-1/k)\eta \leq V(\vee, LE_i \leq 0, \eta)$. After simplification, $(-1 + 1/k + 1/l)\eta \leq V(\vee, LE_i \leq 0, \eta)$. Therefore, $(-1+1/k+1/l)\eta \leq -LE_i + \eta$. Therefore, $LE_i \leq (2-1/k-1/l)\eta \leq 2\eta$.

Proof (Proof for Theorem 5). Let Q'' denote an intermediate post-condition obtained after applying confidence approximations, but prior to full simplification and encoding. From Lemmass 1, 2, and 3, we have:

$$\langle N, P, Q'' \rangle \implies \langle N, P, Q \rangle.$$

Furthermore, from Theorem 4, we have:

$$\langle N', P, Q' \rangle \implies \langle N, P, Q'' \rangle.$$

By transitivity of implication, it follows that:

$$\langle N', P, Q' \rangle \implies \langle N, P, Q \rangle.$$

E Support of strict inequalities

In this section, we present the version of theorem from section 4 that support strict inequalities. The proof of theorems work in the similar lines as we have provided for the earlier theorems.

We can rewrite Lemma 2 as follows to support strict inequalities. Let $<^* \in \{<, \leq\}$. Let us define V for $<^*$:

$$V(\wedge, LE <^* 0, \eta) = LE + \eta \qquad V(\vee, LE <^* 0, \eta) = -LE + \eta$$

Lemma 4. *For $\eta > 0$ and $\beta > 0$, the following holds for Q.*

1. $\forall i.\ \eta <_i^* V(\vee, Q_i, \eta) \Rightarrow V(\wedge, Q, \eta) <^* \eta$.
   If for some i, $<_i^* = <$, then $<^* = <$. Otherwise, $<^* = \leq$.
2. $\exists i.\ V(\wedge, Q_i, \eta) <^* \eta \Rightarrow \eta <^* V(\vee, Q, \eta)$.
3. $V(\wedge, Q, \eta) <^* \beta \Rightarrow \forall i.\ (1 + 1/k)\eta - \beta <^* V(\vee, Q_i, \eta)$.
4. $\beta <^* V(\vee, Q, \eta) \Rightarrow \exists i.\ V(\wedge, Q_i, \eta) <^* 2\eta - \beta/k$.

Let us recursively define $<_Q^*$.

1. For $Q = LE <^* 0$, $<_Q^* = <^*$.
2. For conjunctive Q
$$<_Q^* = \begin{cases} \leq & \forall i.\ <_{Q_i} = \leq \\ < & Otherwise \end{cases}$$
3. For disjunctive Q
$$<_Q^* = \begin{cases} < & \forall i.\ <_{Q_i} = < \\ \leq & Otherwise \end{cases}$$

Lemma 5. *For a given post condition Q and an $\eta > 0$, the following holds*

1. If $Q = LE \leq 0$, $V(\wedge, Q, \eta) \leq \eta \Leftrightarrow Q$
2. If $Q = LE \leq 0$, $V(\vee, Q, \eta) \geq \eta \Leftrightarrow Q$
3. If $Q = LE > 0$, $V(\wedge, Q, \eta) < \eta \Leftrightarrow Q$

4. If $Q = LE > 0$, $V(\vee, Q, \eta) > \eta \Leftrightarrow Q$
5. If Q is conjunctive, $Q \Rightarrow V(\wedge, Q, \eta) <^*_Q \eta$
6. If Q is disjunctive, $Q \Rightarrow \eta <^*_Q V(\vee, Q, \eta)$

Let $\overline{<} = \leq$ and $\overline{\leq} = <$.

Theorem 10. *Let us consider a neural network N, pre-condition P, a post-condition $\neg Q$, and $\eta > 0$.*

1. *If Q is disjunctive, $\langle N', P, y \overline{<^*_Q} \eta \rangle \Rightarrow \langle N, P, \neg Q \rangle$*
2. *If Q is conjunctive, $\langle N', P, \eta \overline{<^*_Q} y \rangle \Rightarrow \langle N, P, \neg Q \rangle$*

F Experiments

In this section, we show the detailed analysis of the Experiments in section 6.

F.1 Benchmarks

We briefly explain each benchmark as follows:
MNIST: We utilized benchmarks from VNN-COMP 2022, which included a total of three networks and 30 VNNLIB files, resulting in 90 benchmarks. These benchmarks were constructed with l_∞ input perturbations of $\epsilon \in \{0.03, 0.05\}$ in 15 randomly choosen images from mnist dataset. All networks are fully connected with ReLU activation functions, featuring between $0.51K$ and $1.54K$ ReLU activations.
CIFAR-10: We selected the first six networks, ranging from `cifar-base-kw.onnx` to `convBigRELU-PGD.onnx`, from VNN-COMP 2022. These networks include convolutional layers, fully connected layers, and ReLU activation functions. Additionally, the networks `resnet-2b` and `resnet-4b`, both residual networks [46] (ResNet), were sourced from VNN-COMP 2021. These residual networks contain two and four residual blocks, respectively. All networks for this dataset were adversarially trained either using COLT [47] or using the method described in [48]. The dataset includes a total of 305 benchmarks, with l_∞ input perturbations ϵ ranging from $\frac{1}{255}$ to $\frac{16}{255}$.
GTSRB: These benchmarks are sourced from VNN-COMP 2024 and are based on binary neural networks (BNNs) trained on the German Traffic Sign Recognition Benchmark (GTSRB) dataset [49]. This multi-class dataset comprises images of German road signs spanning 43 classes, presenting challenges for both humans and models due to factors such as perspective changes, shading, color degradation, and varying lighting conditions. The networks include QConv layers (which binarize the corresponding convolutional layers), Batch Normalization (BN), Max Pooling (MP), and Fully Connected (FC) layers. A total of 45 benchmarks are available for this dataset, created using l_∞ input perturbations with parameter epsilon values of $\{1, 3, 5, 10, 15\}$. These networks do not use explicit activation functions but instead employ a `sign` operator, which converts the input to 1, -1, or 0 if the input is positive, negative, or zero, respectively.

IMAGENET-1K: To analyze the effect on scalability when using appended networks, we used the VGGNET-16 architecture [50], the only one from VNN-COMP that runs on imagenet. The network consists of convolutional layers, ReLU activation functions, and max pooling layers, with a total of $138M$ parameters and $13.16M$ ReLU activations. The properties were generated by applying perturbations on single input pixels to all 150528 input pixels using l_∞ perturbations. The post condition was generated wrt target robustness, where we check for misclassification wrt a fixed target class.

F.2 RQ1-2

Next, we show a detailed analysis of each definition with respect to different datatsets.

Relaxed robustness Figure 8 presents the analysis across different datasets. For the MNIST benchmarks, the number of SAFE cases increases while the number of UNSAFE cases decreases as we raise the confidence threshold. This is intuitive, as a higher confidence threshold makes the definition more relaxed, leading to more SAFE cases. Notably, we observe zero TIMEOUT cases beyond a confidence threshold of 80%. Since these are small networks, $\alpha\beta$-CROWN employs the Gurobi [51] solver for verification. As the confidence threshold increases, the search space tightens, allowing all benchmarks to be verified either by CROWN [39] or the MILP bound-tightening technique. A similar trend is observed for the CIFAR-10 benchmarks—higher confidence thresholds make the definitions more relaxed, resulting in more SAFE cases, which can be efficiently verified by incomplete methods like CROWN.

We observed surprising behavior with the GTSRB dataset, as shown in Figure 8. The number of SAFE, UNSAFE, and TIMEOUT cases remained constant, despite increasing the confidence threshold. There were a total of 45 benchmarks available for this category, resulting in 45 benchmarks for each confidence threshold. For each threshold, we found 42 UNSAFE, 3 TIMEOUT, and 0 SAFE cases. This behavior raised curiosity, prompting us to experiment with a confidence threshold of 98%, but the same pattern persisted. Additionally, we noticed that the confidence for all seed images was 100%. This behavior raises serious concerns about the vulnerability of these networks. If both the seed image and counterexample have near-100% confidence, it is a red flag for these networks. Confidently making incorrect decisions is more dangerous than doing so with lower confidence, as a user might intervene to validate decisions made with lower confidence.

In the IMAGENET-1K benchmarks, there are a total of 18 benchmarks, implying 18 benchmarks for each confidence threshold. We observed on a GPU machine that many of the benchmarks ran out of memory with 32GB of GPU RAM. As a result, we had to run them on a CPU machine: Intel(R) Xeon(R) Gold 6314U CPU @ 2.30GHz with 64GB of RAM. For the standard robustness property with a confidence of 0%, we observed 1 UNSAFE, 6 SAFE, and 11

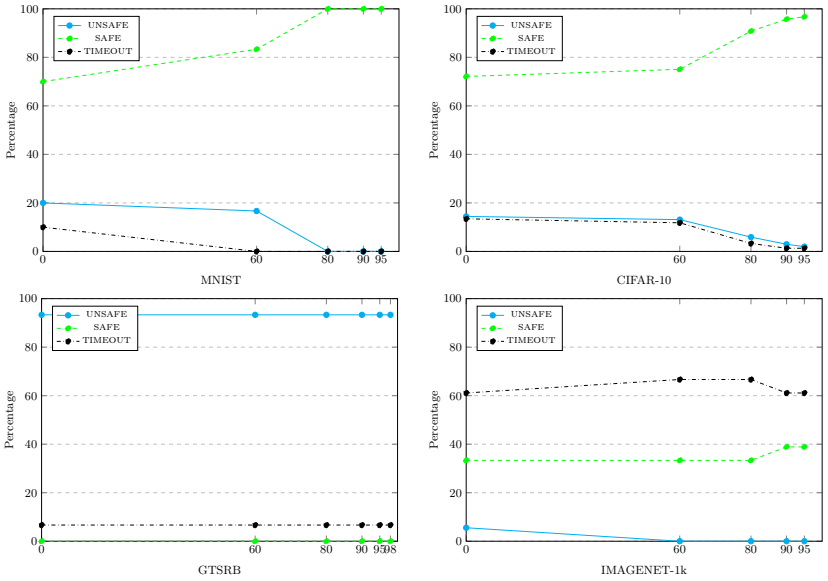


Fig. 8: Analysis of relaxed robustness with respect to each dataset separately.

TIMEOUT cases. At confidence levels of 60% and 80%, the 1 UNSAFE case was converted to a TIMEOUT. At 90% and 95% confidence, it was converted to a SAFE case. The 11 TIMEOUT cases remained as TIMEOUT for all confidence levels.

Figures 9, 10, and 11, and 12 provide more insightful examples of the relaxed robustness definition.

Strong robustness Figure 13 shows the strong robustness behavior with varying threshold levels. In addition to SAFE, UNSAFE, and TIMEOUT, we also introduced a new attribute to the graphs, namely TRIVIAL SAFE. The TRIVIAL SAFE

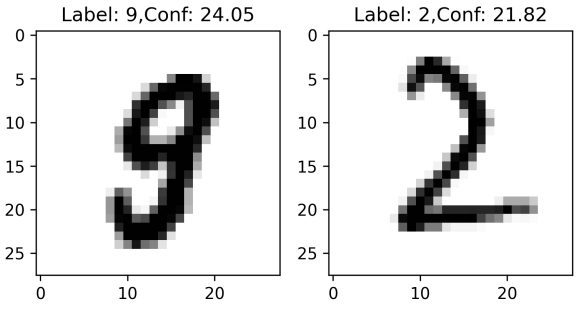


Fig. 9: The above images were verified using relaxed robustness with a confidence threshold of 80%.

Fig. 10: The image on the left was correctly classified as label AIRPLAN by the network `convBigRELU-PGD.onnx` with a confidence of 91.37%, but it was misclassified as `automobile` with a confidence of 22.13% within an ϵ perturbation of 0.007 under the standard robustness property. The same image was then verified by relaxed robustness with a confidence threshold of 90%.

Fig. 11: The left image is taken from the German Traffic Sign Recognition Benchmark (GTSRB) and belongs to the class `Speed limit (80 km/h)`. It is correctly classified by `net-1` (Table 1) with 100% confidence. However, under an ϵ-perturbation of 5/255, it is misclassified as `Speed limit (120 km/h)` with a high confidence of 99.99%, highlighting a potential vulnerability in the network. The right image belongs to the class `Right-of-way at the next intersection` and is classified with 100% confidence. With the same ϵ-perturbation, it is misclassified as `Beware of ice/snow` with a confidence of 98.19%.

cases occur when the premise of the strong robustness condition in Eq 5 becomes false, because the confidence of the seed image is below the threshold. As mentioned in the experimental section of the main paper, we set a 57% confidence threshold for the seed images, as it represents the average confidence across all seed images and datasets. We observed that all the MNIST benchmarks became trivially safe, as the confidence of all seed images was below 57%. Therefore, we conducted a separate analysis for the MNIST dataset with varying threshold values.

We took 22% as the threshold on seed images for mnist benchmarks, and threshold values 15, 17, and 20 on the right side thresholds of the strong robustness equation. We found that after confidence threshold 15% the SAFE and TIMEOUT cases become 0, only UNSAFE and TRIVIAL SAFE cases remained. This implies these networks are not strong robust with respect to the given threshold values.

For the CIFAR-10 dataset, we used the same threshold values mentioned in the experimental section of the main paper. As we increased the confidence threshold on the right-hand side of the implication in the strong robustness

Fig. 12: The image on the left is correctly classified as CARPENTERS KIT with 94.0% confidence by the VGGNET-16 network. However, within an epsilon perturbation of $1e-5$, it is misclassified as ABACUS with 38.23% confidence. This image was verified under relaxed robustness with a 95% confidence threshold.

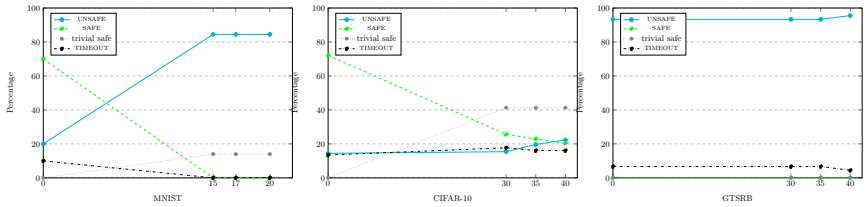


Fig. 13: Analysis of strong robustness with respect to each dataset separately.

equation, the number of UNSAFE cases increased, while the number of SAFE cases decreased. This trend is intuitive, as a higher confidence requirement increases the likelihood of falsifying the condition. The number of TIMEOUT cases remained almost unchanged.

We used the same threshold values as those for the CIFAR-10 dataset for the GTSRB dataset. The number of TRIVIAL SAFE cases is zero since all seed images have a confidence of 100%. We observed almost the same behavior as in relaxed robustness, except for one case that converted from TIMEOUT to UNSAFE at the confidence threshold of 40%.

Smoothness Figure 15 illustrates the behavior of the smoothness property across all three datasets. We observed that for the MNIST and GTSRB datasets, almost 99% of the benchmarks did not satisfy the smoothness property, indicating highly unsmooth behavior in these networks. The Figure 14 shows the timeout case. The CIFAR-10 benchmarks performed reasonably well, with the number of SAFE cases increasing as the smoothness threshold increased. This trend is intuitive since a higher smoothness threshold allows for greater variations in the output. Notably, we observed a slight increase in TIMEOUT cases, which may be due to the higher complexity of this property compared to re-

Fig. 14: This image from the GTSRB benchmark was classified correctly `Turn right ahead (33)` with 100% confidence by the network $net - 3$ in Table 1. However, this case resulted in a timeout in all three definitions: relaxed robustness, strong robustness, and smoothness.

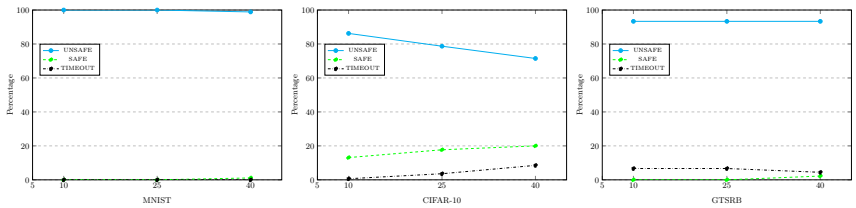


Fig. 15: Analysis of smoothness with respect to each dataset separately.

laxed and strong robustness. Unlike the other properties, smoothness requires verifying both upper and lower bounds of the output confidence.

Top-k robustness Affinity robustness requires a predefined set of classes in which misclassification is allowed (prior knowledge). For MNIST, this affinity set is defined as: $\{\{0, 8\}, \{4, 9\}, \{1, 9, 7\}, \{2\}, \{3\}, \{5\}, \{6\}\}$. Intuitively, this means that 0 is allowed to be misclassified as 8 but not as any other class, while 9 is allowed to be misclassified as 4, 1, or 7 only. Similarly, 2 is not allowed to be misclassified into any other class. For CIFAR-10, we define the affinity set as: $\{\{bird\}, \{airplane, automobile, ship, truck\}, \{cat\}\ \{deer, dog, horse\}, \{dog\}, \{frog\}\}$. Here, machines (airplane, automobile, ship, truck) are allowed to be classified among themselves, and the same applies to animals (deer, dog, horse). The affinity set is user-defined and may vary depending on the user or application. We took $k = 2$ for both mnist and cifar-10 benchmarks.

Figure 16 compares the results on the MNIST and CIFAR-10 datasets separately. As mentioned in the experiment section of the main paper, all the seed images in the GTSRB dataset have 100% confidence, making top-k analysis inapplicable to those benchmarks.

For the MNIST benchmarks, the results align with intuition: SAFE cases increase while UNSAFE cases decrease as we move from standard robustness to affinity and then to top-k relaxed robustness. This trend is likely due to the relatively small network sizes, making it easier to find solutions.

On the other hand, for the CIFAR-10 benchmarks, while SAFE cases are theoretically expected to increase, we observed a rise in TIMEOUT cases instead.

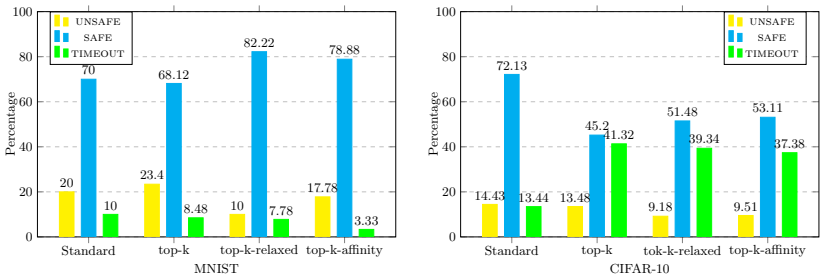


Fig. 16: Top-K: The left figure shows the comparison on MNIST benchmarks, right figure shows comparison on CIFAR-10 benchmarks

A potential reason for this could be the increased complexity of the problem, specifically the presence of conjunctions of disjunctions, making verification more computationally challenging.

We wish to recall that Figure 4D compares standard robustness with top-k relaxed and top-k affinity robustness. The number of UNSAFE cases decreases from standard robustness to the other two robustness definitions. Although both versions of top-k robustness are more relaxed than standard robustness, the number of SAFE cases also decreases due to an increase in the number of TIMEOUT cases. One possible reason for the increase in timeouts is the added complexity of the constraints in both top-k robustness definitions. In standard robustness, the property consists of a disjunction of atomic properties, whereas in top-k robustness, it involves a conjunction of disjunctive clauses. This requires additional layers to be appended to the existing neural network, increasing computational complexity.

F.3 RQ3

Figure 17 compares three approaches: MARABOU DIRECT ENCOD, which directly encodes the property using the Marabou interface; MARABOU SIMPLIFIED PROPS, which encodes the simplified property with an appended network using the Marabou interface; and $\alpha\beta$-CROWN SIMPLIFIED PROPS, which applies $\alpha\beta$-CROWN to the simplified property and appended networks.

Figure 17 specifically compares these methods on strong robustness and smoothness, while comparisons on other properties are presented in the experiment section of the main paper. We observed that $\alpha\beta$-CROWN resulted in almost zero TIMEOUT cases for both properties. Additionally, using state-of-the-art tools on appended networks with simplified properties, as converted by our framework, showed improved performance compared to directly encoding the original properties with Marabou.

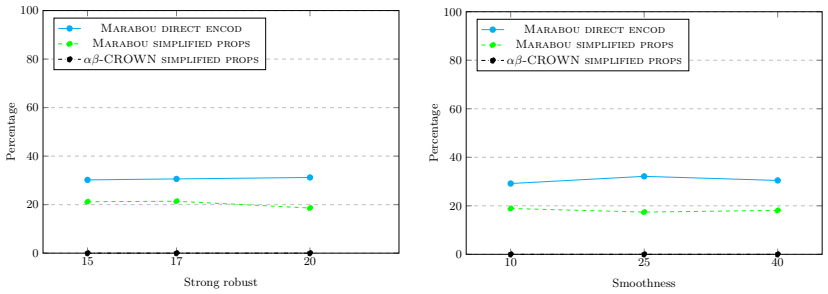


Fig. 17: Comparison of the constraint-based solver MARABOU with and without the simplified property, alongside $\alpha\beta$-CROWN with the simplified property. The y-axis represents the percentage of TIMEOUT cases, and the x-axis represents different threshold values. The left figure shows the comparison with strong robustness, while the right figure shows the comparison with smoothness.