Experimental Evaluation of Neural Network Verification Tools

1. Goal

Assessing the Performance of State-of-the-Art Verifiers Across VNN COMP Iterations.

2. Problem formalization

Neural Network Verification Problem----We consider the neural network verification problem defined as follows:

Given an **input specification** $\phi \subseteq \mathbb{R}^{d_{\text{in}}}$ (also called *pre-condition*), an **output specification** $\psi \subseteq \mathbb{R}^{d_{\text{out}}}$ (also called *post-condition*), and a **neural network** $N : \mathbb{R}^{d_{\text{in}}} \mapsto \mathbb{R}^{d_{\text{out}}}$, we aim to prove that the pre-condition implies the post-condition:

$$ \forall x : x \vDash \phi \Rightarrow N(x) \vDash \psi $$

Adversarial Robustness in Image Classification: One particularly popular property is the robustness to adversarial ℓ_{∞}-norm bounded perturbations in image classification.

There, the network N computes a numerical score $y \in \mathbb{R}^{d_{\text{out}}}$ corresponding to its confidence that the input belongs to each of the d_{out} classes for each input $x \in \mathbb{R}^{d_{\text{in}}}$. The final classification c is then computed as:

$$ c = \arg\max_{i} N(x)_i $$

In this setting, an adversary may want to perturb the input such that the classification changes. Therefore, the verification intends to prove that:

$$ \arg\max_{i} N(x^{\prime})*i = t, \quad \forall x^{\prime} \in {x^{\prime} \in \mathbb{R}^{d*{\text{in}}}} \mid |x - x^{\prime}|_{\infty} \leq \epsilon} $$

where:

- t is the target class
- x is the original image
- ϵ is the maximal permissible perturbation magnitude

Thus, the **pre-condition ϕ** describes the inputs an attacker can choose from (an ℓ_{∞}-ball of radius ϵ):

$$ \phi = {x^{\prime} \in \mathbb{R}^{d_{\text{in}}} \mid |x - x^{\prime}|_{\infty} \leq \epsilon} $$

The **post-condition ψ** describes the output space corresponding to classification to the target class t:

$$ \psi = {y \in \mathbb{R}^{d_{\text{out}}} \mid y_t > y_i, \ \forall i \neq t} $$

3. Terminology

Instance: An instance is defined by a property specification (pre- and post-condition), a network, and a timeout. For example, one instance might consist of an MNIST classifier with one input image, a given local robustness threshold ϵ, and a specific timeout.

Benchmark: A benchmark is defined as a set of related instances. For example, one benchmark might consist of a specific MNIST classifier with 100 input images, potentially different robustness thresholds ε, and one timeout per input.

Format: ONNX for neural networks and VNN-LIB for specifications.

4. Methods and Algorithms (Verifiers)

We have chosen the following verifiers for performance evaluation based on their strong overall results in previous VNN COMP and their open-source availability:

Verifiers	2021	2022	2023	2024	2025	Hardware and Licenses	Participated Benchmarks	Link—Latest Version	Description

Verifiers	2021	2022	2023	2024	2025	Hardware and Licenses	Participated Benchmarks	Link—Latest Version	Description
α-β-CROWN	1st	1st	1st	1st	1st	CPU and GPU with 32-bit or 64-bit floating point; Gurobi license required for certain benchmarks	All benchmarks	GitHub	α,β-CROWN is an efficient neural network verifier based on the linear bound propagation framework and built on a series of works on boundpropagation-based neural network verifiers: CROWN, auto LiRPA, α-CROWN, β-CROWN, GCP-CROWN, GenBaB, BICCOS. The core techniques in α,βCROWN combine the efficient and GPU-accelerated linear bound propagation method with branch-and-bound methods specialized for neural network verification.
nnenum	8th	4th	5th	5th	5th	CPU, Gurobi license (optional)	acasxu, cgan, collins-rul-cnn, cora, linearizenn, metaroom, safeNLP, nn4sys, tlverifybench, vggnet16	GitHub	The nnenum tool uses multiple levels of abstraction to achieve high performance verification of ReLU networks without sacrificing completeness. The core verification method is based on reachability analysis using star sets, combined with the ImageStar method to propagate stes through all linear layers supported by the ONNX runtime, such as convolutional layers with arbitrary parameters.
Marabou	5th	7th	2nd	4th	4th	CPU, no license required. Can also be accelerated with Gurobi (which requires a license)	acasxu, cgan, collins rul cnn, list shift, linearizenn, metaroom, nn4sys, safenlp, verifybench, cifar100, tinyimagenet	GitHub	Marabou can answer queries about a network's properties by encoding and solving these queries as constraint satisfaction problems. Marabou supports many different linear, piecewise-linear, and non-linear operations and architectures (e.g., FFNNs, CNNs, residual connections, Graph Neural Networks). Marabou performs complete analysis that employs a specialized convex optimization procedure and abstract interpretation. It also uses the Split-and-Conquer algorithm for parallelization.
NNV	-	-	6th	6th	6th	CPU, MATLAB license	Regular track except for LinearizeNN	GitHub	NNV uses a star-set state-space representation and reachability algorithm that allows for a layer-by-layer computation of exact or overapproximate reachable sets for feed-forward, convolutional, semantic segmentation (SSNN), and recurrent (RNN) neural networks, as well as neural network control systems (NNCS) and neural ordinary differential equations (Neural ODEs).

Verifiers	2021	2022	2023	2024	2025	Hardware and Licenses	Participated Benchmarks	Link—Latest Version	Description
NeuralSAT	-	-	4th	2nd	2nd	GPU, Gurobi License	acasxu, cgan, collins-rul-cnn, dist-shift, nn4sys, vggnet16, tlverifybench, trafic-signs-recognition, reach-prob-density, metaroom	GitHub	NeuralSAT integrates conflict-driven clause learning (CDCL) in SAT/SMT-solving with an DNN abstraction based theory solver for infeasibility checking. The design of NeuralSAT is inspired by the core algorithms used in SMT solvers such as CDCL components (light shades) and theory solving (dark shade). NeuralSAT does not require parameter tuning and works out of the box, e.g., the rool runs on the wide-range of benchmarks in VNN-COMPs without any tuning.
CORA	-	-	-	7th	4th	GPU, MATLAB license	acasxu, cifar100, collins-rul-cnn, cora, dist-shift, nn4sys, safenlp, tinyimagenet, tlverifybench	GitHub	CORA enables the formal verification of neural networks, both in open-loop as well as in closed-loop scenarios. Open-loop verification refers to the task where properties of the output set of a neural network are verified, e.g. correctly classified images given noisy input, as also considered at VNN-COMP. In closed-loop scenarios, the neural network is used as a controller of a dynamic system, e.g., controlling a car while keeping a safe distance over some time horizon. This is realized using reachability analysis, mainly using polynomial zonotopes, allowing a non-convex enclosure of the output set of a neural network.

5. Datasets (Benchmarks)

We have selected the following benchmarks from VNN COMP, all focused on image verification. They are listed in descending order of complexity, as indicated by their parameter counts:

Benchmark	Number of Instances	Application	Network Types	Parameters	Effective Input Dim	Track	Available Verification Tools	Links	Description

Benchmark	Number of Instances	Application	Network Types	Parameters	Effective Input Dim	Track	Available Verification Tools	Links	Description
cGAN	20	Image Generation	Conv. + Vision Transformer	500k - 68M	5	regular	α-β-CROWN, nnenum, NeuralSAT, NNV, Marabou	GitHub	The cGAN benchmark, proposed by Stony Brook University, addresses the verification of conditional generative adversarial networks (cGANs) for robust image classification. The network uses a 100×100×100 image space camera images based on a 1D distance condition and a 4D noise vector. The network under verification combines the generator and discriminator. Specifications check if the discriminator's predicted distance for the generated image aligns with the input condition, given constrained input ranges. Models vary in architecture (CNN and Vision Transformer) and image size (32x32, 64x64) to offer different difficulty levels.

Benchmark	Number of Instances	Application	Network Types	Parameters	Effective Input Dim	Track	Available Verification Tools	Links	Description
ViT	200	Vision	Conv. + Residual + Softmax + BatchNorm	68k - 76k	3072	extended	α-β-CROWN, NeuralSAT, Marabou	GitHub	The ViT benchmark, proposed by the α-β-CROWN team, introduces verification tasks for Vision Transformers (ViTs) to address their complex architecture and diverse nonlinearities. It includes two modified, smaller ViT models (with 2-3 layers, 3 attention heads, and batch normalization replacing layer normalization) trained on the CIFAR-10 dataset. The model evaluates robustness verification under ℓ_∞ perturbations ($\epsilon=1/255$) on 100 carefully selected test instances per model, excluding easily verifiable or attackable examples. A 100-second timeout is set for each instance.

Benchmark	Number of Instances	Application	Network Types	Parameters	Effective Input Dim	Track	Available Verification Tools	Links	Description
cifar100	200	Image Classification	FC + Conv. + Residual, ReLU + BatchNorm	2.5M - 3.8M	3072	regular	α-β-CROWN, NeuralSAT, NNV, CORA	GitHub	The CIFAR100 benchmark, proposed by the α-β-CROWN team, is a reused benchmark from VNN-COMP 2022 featuring two ResNet models of medium and large sizes trained on CIFAR-100. The models have 17-19 convolutional layers plus 10-10 convolutional layers. The model is trained on CIFAR-100 with different training methods. The model has 100-second timeout. Instances easily verified by vanilla CROWN were filtered out to increase difficulty, while about 18% of instances with adversarial examples were retained to help identify unsound verification results.

Benchmark	Number of Instances	Application	Network Types	Parameters	Effective Input Dim	Track	Available Verification Tools	Links	Description
CORA	180	Image Classification	FC + ReLU	575k, 1.1M	784, 3072	regular	α-β-CROWN, nnenum, NeuralSAT, NNV, CORA	GitHub	The CORA benchmark, proposed by the CORA team, focuses on promoting fast neural network verification by using small timeout. It features a uniform ResNet architecture with three different training methods: standard, interval-bound propagation, and set-based training. The verification task is to confirm correct classification for all images within specified input sets.

6. Verifiers available for Benchmarks

Since not every verifier is capable of being evaluated on all benchmarks, we have compiled the following table detailing their respective availabilities:

	cGAN	ViT	cifar100	CORA
α-β-CROWN	√	√	√	√
nnenum	√	×	×	√
Marabou	√	√	×	×
NNV	√	×	√	√
NeuralSAT	√	√	√	√
CORA	×	×	×	√

7. Evaluation Metrics

For each benchmark, we provide the following evaluation matrix to assess the performance of verifiers:

Evaluation Dimension	Calculation Method	Evaluation Purpose
Verified	Number of successfully verified instances	Measure the ability of verifiers to prove safety properties
Falsified	Number of successfully falsified instances	Evaluate the ability of verifiers to find counterexamples
Incorrect result	Number of instances with incorrect verification	Assess the situation of wrong answers from verifiers
Solved	Total number of solved instances	Core metric for comprehensive performance
Time	Total time to verify a benchmark	Measure the verification efficiency of verifiers
Rank	Rank verifiers based on total solved instances	Provide intuitive performance comparison