# Stress-Testing Neural Network Verifiers with Provably Robust Instances

**David Troxell** [1] **Yulia Alexandr** [2] **Sofia Hunt** [1] **Stephanie Lei** [1] **Guido Montúfar** [1 2 3]

## Abstract

Neural network verifiers aim to provide formal guarantees on model behavior, but existing verification benchmarks are fundamentally limited by their lack of ground-truth labels. As a result, verifier evaluation relies on indirect heuristics, which prevents exact scoring and systematic study of verifier failure modes. We address this gap by introducing a reusable framework for generating verification instances whose ground-truth robustness labels are known a priori through analytic construction. Our framework led to the discovery of multiple numeric tolerance concerns and an implementation bug in popular verifiers, highlighting the need for ground-truth labels. Additionally, to systematically study verifier failure modes, we introduce the verification Difficulty Profile, a collection of estimable quantities capturing distinct sources of instance hardness. Using our framework and these profiles, we evaluate five state-of-the-art verifiers and show that different instances stress distinct aspects of the verification pipeline. We show that these results can aid the future development of verifiers as they provide actionable targets for improving numerical reliability, relaxation quality, and search behavior. Our code is publicly available: https://github.com/ctbanonymous12345/VeriStressGT.git.

## 1. Introduction

Deep learning systems are rapidly being integrated into safety-critical applications in society, including AI-enabled medical devices, self-driving vehicles, power grid balancing for energy systems, and quality control for chemical processes (Stanford Institute for Human-Centered Artificial Intelligence, 2025; Ozcanli et al., 2020; Yu and Zhang, 2023). However, these systems often exhibit fragile and unpredictable behavior as small input perturbations can induce large and unintended changes in outputs (Drenkow et al., 2022; Meng et al., 2024; Liu and Jin, 2023). In applications where unexpected fluctuations in model behavior have dire consequences, such instability raises a fundamental question of how to guarantee that a trained model behaves safely under all admissible inputs.

Neural network verification aims to answer this question by providing formal guarantees on model behavior. Given a trained model, an admissible input set, and a desired property, verifier systems attempt to certify whether the property holds for all inputs in the set. This work focuses on robustness specifications, where the goal is to certify stable model behavior under bounded input perturbations. A wide array of verification techniques have been developed in recent years, including Satisfiability Modulo Theory (SMT) and Mixed Integer Programming (MIP) approaches (Katz et al., 2017; Tjeng et al., 2019), convex relaxation methods (Wong et al., 2018), and scalable branch-and-bound frameworks such as $\alpha, \beta$-CROWN (Wang et al., 2021; Xu et al., 2020b). Modern verifiers can now handle networks with millions of parameters and increasingly complex architectures.

However, despite algorithmic progress, the evaluation of neural network verifiers themselves remains fundamentally limited. Specifically, existing benchmarks generally lack ground-truth labels indicating whether a verification instance is truly robust or non-robust, and they offer little characterization of instance difficulty beyond verifier runtime. These limitations have three primary, direct consequences:

First, without ground-truth labels, verifier evaluation currently relies on indirect heuristics. For example, VNN-COMP, the annual competition for neural network verification, uses majority voting when verifiers disagree on a model instance due to numerical tolerance issues (Kaulen et al., 2025). Additionally, when no verifier finds a valid robustness counterexample and instead claims robustness or times out, the instance is treated as robust under an assumption of verifier soundness (Kaulen et al., 2025). However,

[1]Department of Statistics and Data Science, University of California, Los Angeles, USA [2]Department of Mathematics, University of California, Los Angeles, USA [3]Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany. Correspondence to: David Troxell <davidtroxell@ucla.edu>.

*Accepted to the 1st Workshop on Combining Theory and Benchmarks*, CTB@ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

this assumption may be more fragile than anticipated as recent works have found bugs and imprecisions in various verifier implementations (Zhou et al., 2025; Jia and Rinard, 2020; Zombori et al., 2021).

Second, without ground-truth labels, benchmarks cannot systematically stress-test the ability of verifiers to certify increasingly difficult robust instances. Non-robust instances are often quickly exposed by adversarial search methods such as projected gradient descent, making them relatively easier to classify (Madry et al., 2019). The harder and more informative regime is robust-but-difficult instances, where a verifier must prove that no counterexample exists. Recent work makes existing benchmarks harder while preserving their original ground truth by destabilizing ReLU activations (Li and Nguyen, 2025). However, this approach inherits labels from the input benchmark and only varies one difficulty mechanism. Benchmarks therefore still lack a general way to generate robust instances with verifier-independent labels and varied, targeted knobs for different bottlenecks.

Third, current verifier benchmark outcomes alone often provide limited diagnostic information about why a verifier succeeds, fails, or times out. Prior work has shown that performance is related to coarse factors such as network architecture and input dimension (Xu et al., 2020a). However, these high-level factors provide less insight into whether a verifier is limited by loose relaxations, complex local geometry, or other bottlenecks. Without targeted instances that isolate such bottlenecks, it is also difficult to evaluate verifiers designed to address a specific failure mode.

In this work, we address these gaps by introducing VeriStress-GT (Verifier Stress-Testing via Ground Truth), a modular and re-usable framework to generate benchmark instances for neural network verification with analytically proven ground-truth robustness labels. Moreover, we design these constructions to have controllable difficulty, enabling gradual systematic stress-testing of verification methods for the first time. To guide this process and better analyze verifier failure modes, we introduce the Difficulty Profile, a collection of estimable quantities that characterize verification instance hardness. More broadly, this work reframes verification benchmarking from heuristic evaluation to controlled, ground-truth experimentation. Our primary contributions are as follows:

- We introduce VeriStress-GT, to our knowledge the first reusable framework for generating neural network verification instances with verifier-independent ground-truth robustness labels and controllable difficulty for systematic verifier stress-testing.
- We propose the verification instance Difficulty Profile, a unified framework for characterizing verification instance hardness. We show that the profile components enable the systematic evaluation of verifier failure modes.
- We evaluate five state-of-the-art verifiers using VeriStress-GT and Difficulty Profiles, revealing distinct limitations across verification paradigms and identifying previously unreported numerical and implementation issues.

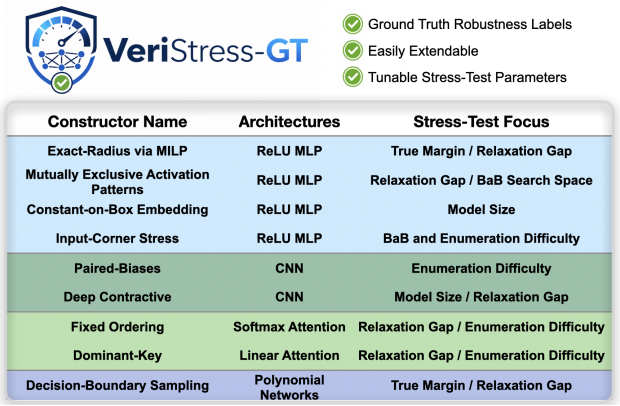



*Figure 1.* High-level overview of VeriStress-GT, a benchmark for stress-testing neural network verifiers. Benchmark instances are generated via various constructors, or methods to generate provably robust instances. New constructors can easily be added to the framework, and each constructor admits difficulty parameters that can gradually increase verification difficulty while retaining robustness.

## 2. Robust Constructions for Neural Network Verification Benchmarking

We first define the robustness verification instance as the tuple $(f, x_0, y, \mathcal{B}_\epsilon)$ consisting of a trained neural network $f : \mathbb{R}^d \to \mathbb{R}^c$, a nominal input $x_0 \in \mathbb{R}^d$ with correct class $y \in [c]$, and a perturbation set $\mathcal{B}_\epsilon(x_0) = \{x : \|x - x_0\|_p \leq \epsilon\}$. In this work we specifically study verification for network robustness; in other words, does $f$ classify every input in $\mathcal{B}_\epsilon(x_0)$ identically to $x_0$? Define the per-class margin functions $\mu_k(x) = f_y(x) - f_k(x)$ for $k \in [c], k \neq y$ and the overall margin $\mu(x) = \min_{k \neq y} \mu_k(x)$. Then, we state that $f$ is provably robust at $x_0$ if and only if $\min_{x \in \mathcal{B}_\epsilon(x_0)} \mu(x) > 0$.

Rather than defining a fixed set of benchmark instances, VeriStress-GT defines a novel collection of constructors, each of which produces verification instances with analytically proven robustness labels. Each constructor targets a particular architectural class or verifier bottleneck, allowing the benchmark to probe different sources of verification difficulty. Figure 1 provides a high-level overview of the constructors and framework design principles. We briefly describe each of the constructors within the framework below and include constructor details in Appendix B.

### 2.1. ReLU Network Constructors

Throughout this section, let $\sigma(t) = \max\{t, 0\}$ denote the ReLU activation function applied elementwise. A ReLU MLP is a composition of affine maps and ReLU, producing logits $f_1(x), \ldots, f_c(x)$.

**Exact-Radius via Mixed-Integer Linear Programming (MILP).** We generate near-boundary robust instances by computing the network's exact robustness radius using a standard mixed-integer linear programming (MILP) encoding of ReLU networks (Tjeng and Tedrake, 2017). Specifically, for each target class $k \neq y$, we solve a mixed-integer linear program that minimizes the $\ell_\infty$ perturbation such that $f_k(x) \geq f_y(x)$. Let $t_k^\star$ be the optimal perturbation for class $k$, and define $r^\star = \min_{k \neq y} t_k^\star$. Then any perturbation radius $\epsilon < r^\star$ gives a robust instance, while choosing $\epsilon$ close to $r^\star$ produces an instance with little robustness slack. The full MILP formulation is provided in Appendix B.1.

**Constant-on-Box Embedding.** Given an intermediate latent dimension $d_z$ and two neural networks $\Pi : \mathbb{R}^d \mapsto \mathbb{R}^{d_z}$, $\psi : \mathbb{R}^{d_z} \mapsto \mathbb{R}^c$, we consider the composition $f = \psi(\Pi(x))$. If $\Pi(x) = t \ \forall x \in \mathcal{B}_\epsilon(x_0)$ for some constant vector $t \in \mathbb{R}^{d_z}$, then any arbitrarily large, complex downstream $\psi$ is also constant on the perturbation set. Furthermore, if $(\psi(t))_y - (\psi(t))_k \geq \gamma > 0$ for all $k \neq y$, then $f$ is robust with margin at least $\gamma$. For $\ell_\infty$ perturbations, the set $\mathcal{B}_\epsilon(x_0)$ is an axis-aligned box, so the condition $x \in \mathcal{B}_\epsilon(x_0)$ decomposes into independent interval constraints on each input coordinate. This allows $\Pi$ to be built from one-dimensional ReLU hinge functions applied separately to each coordinate. To see this in one dimension, choose two scalars $\alpha$ and $\beta$ such that $[x_0 - \epsilon, x_0 + \epsilon] \subseteq [\alpha, \beta]$. Then the scalar map $x \mapsto t + \sigma(\alpha - x) + \sigma(x - \beta)$ is equal to $t$ throughout the perturbation interval. Applying this coordinatewise gives a ReLU network that collapses the entire input box to a single point in the latent dimension. The primary parameter used to stress-test verifiers is therefore the depth and width of the downstream network $\psi$, which can be made arbitrarily large while preserving the same ground-truth robustness label.

**Mutually Exclusive Activation Patterns (MEAP).** Most relaxation-based verifier implementations initially treat network neurons independently before performing additional computationally intensive refinement. Therefore, to stress-test verifiers, MEAP constructs a ReLU network whose hidden units are individually unstable (i.e., switch between active and inactive as $x$ varies within $\mathcal{B}_\epsilon(x_0)$) but are coupled by geometric constraints. Specifically, neurons are organized into pairs such that, for every input in the perturbation set, each pair contains at least one active neuron. To embed this geometric constraint, the network first forms $P$ pairs of affine preactivations. For each $p \in [P]$, let $z_{p,1}(x) = w_p^\top x + b_{p,1}$ and $z_{p,2}(x) = -w_p^\top x + b_{p,2}$. Each pair is then aggregated by $r_p(x) = \max\{\sigma(z_{p,1}(x)), \sigma(z_{p,2}(x))\}$, and the target logit is defined as the minimum over pair scores:

$$f_y(x) = \min_{p \in [P]} r_p(x), \qquad f_k(x) = 0 \quad \text{for all } k \neq y. \quad (1)$$

This function is piecewise linear and can be implemented as a ReLU network using standard identities for pairwise maxima and minima. The robustness follows from choosing biases $b_{p,1}$ and $b_{p,2}$ such that, for every $x \in \mathcal{B}_\epsilon(x_0)$, at least one preactivation in each pair is positive. Equivalently, each pair satisfies $r_p(x) > 0$ throughout the perturbation set. Hence $f_y(x) > 0 = f_k(x)$ for all $k \neq y$, certifying robustness. Appendix B.2 gives the full construction and proof.

**Input-Corner Stress.** The input-corner constructor creates ReLU networks whose robustness certificate is given by finite corner evaluation. Let $\mathcal{B}_\epsilon(x_0)$ be an $\ell_\infty$ box, suppose the logits depend only on an active set $A \subseteq [d]$ of $d_{act}$ input coordinates. We set the correct-class logit to be constant, $f_y(x) = 0$, and define each competitor logit as $f_k(x) = h_k(x) - \beta_k$, where $h_k$ is convex on the active-coordinate box. Then the per-class margin $\mu_k(x) = f_y(x) - f_k(x) = \beta_k - h_k(x)$ is concave, so its minimum over the box is attained at an extreme point. Let $\mathcal{V}_\epsilon(x_0; A) = \{x_0 + \epsilon s : s_i \in \{-1, +1\} \text{ for } i \in A, s_i = 0 \text{ for } i \notin A\}$ denote the active-coordinate corners. Choosing $\beta_k = \max_{v \in \mathcal{V}_\epsilon(x_0;A)} h_k(v) + \gamma$ implies $\mu_k(x) \geq \gamma$ for all $x \in \mathcal{B}_\epsilon(x_0)$, so the instance is robust with margin at least $\gamma$. A ReLU implementation takes $h_k(x) = \sum_{j=1}^{J} c_{k,j} \sigma(a_j^\top x + b_j)$ with $c_{k,j} \geq 0$, since nonnegative sums of ReLU hinges are convex. The construction can nevertheless be difficult for standard verifiers as many hinges can be placed near instability at $x_0$, making local relaxations loose, while the true proof relies on the fact that convex functions attain their maximum over a box at a corner. The main difficulty parameters are the number of hinges $J$, active dimension $d_{act}$, hinge scale $\|a_j\|_1$, and margin $\gamma$. Appendix B.3 gives the formal corner certificate and implementation details.

### 2.2. CNN Constructors

For CNN constructors, inputs are tensors $x \in \mathbb{R}^{C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}}$. We write $Z$ for an intermediate feature map and use $\star$ to denote convolution with padding and stride fixed by the architecture. A convolutional layer with filters $\{\mathcal{W}_i\}_{i=1}^K$ and biases $\{b_i\}_{i=1}^K$ produces preactivations $u_{hw}^{(i)} = (\mathcal{W}_i \star Z)_{hw} + b_i$ at output channel $i$ and spatial location $(h, w)$, followed by a coordinatewise activation such as ReLU.

**Deep Contractive CNN.** This constructor attains robustness by forcing perturbations to contract through depth. We write the feature extractor as $\Phi = C_D \circ C_{D-1} \circ \cdots \circ C_1 \circ P$, where $P$ is an optional front-end map and $C_1, \ldots, C_D$ are contractive convolutional blocks. Let $L_{\text{front}}$ denote the $\ell_\infty$

Lipschitz constant of $P$, with $L_{\text{front}} = 1$ if no front-end map is used. Each block $C_\ell$ consists of a convolutional layer followed by a 1-Lipschitz pointwise activation function and is normalized to have induced $\ell_\infty$ Lipschitz constant of at most $\lambda < 1$. Intuitively, then, each layer in $P$ may expand perturbations by at most $L_{\text{front}}$ while each subsequent block shrinks the worst-case perturbations by a factor of at most $\lambda$. The formal Lipschitz certificate in Appendix B.4 shows that:

$$\|\Phi(x) - \Phi(x')\|_\infty \le L_{\text{front}} \lambda^D \|x - x'\|_\infty. \tag{2}$$

We set the correct-class logit to be a large constant plus a centered perturbation term $f_y(x) = \Gamma + w_y^\top(\Phi(x) - \Phi(x_0))$ and set each incorrect class logit equal to 0. Therefore, over the perturbation set $\mathcal{B}_\epsilon(x_0)$, $f_y$ can decrease by at most $\|w_y\|_1 L_{\text{front}} \lambda^D \epsilon$. Choosing $\Gamma > \|w_y\|_1 L_{\text{front}} \lambda^D \epsilon$ therefore certifies robustness on $\mathcal{B}_\epsilon(x_0)$. The main difficulty parameters are the depth $D$, contraction rate $\lambda$, spatial dimension, front-end Lipschitz constant $L_{\text{front}}$, and margin slack $\Gamma - \|w_y\|_1 L_{\text{front}} \lambda^D \epsilon$.

**Paired-Bias CNN.** The paired-bias CNN construction is a convolutional analogue of MEAP since it involves creating pairs of convolutional channels whose ReLU activations may be unstable, but whose difference is globally nonnegative. The construction may be applied after arbitrary upstream convolutional layers since the certificate holds for every possible value of the shared convolutional response. Let $Z = \Psi(x)$ denote the feature map produced by this upstream network. For each filter pair $i \in [P]$, two channels share the same convolutional filter $\mathcal{W}_i$ but have ordered biases $b_i > c_i$. At the layer where the construction is applied, let $s^{(i)} = \mathcal{W}_i \star Z \in \mathbb{R}^{H_{\text{sp}} \times W_{\text{sp}}}$ denote the shared convolutional response for pair $i$, where $H_{\text{sp}}$ and $W_{\text{sp}}$ are the height and width of the feature map. Therefore, at spatial location $(h, w)$, the paired ReLU inputs are $s^{(i)}_{hw} + b_i$ and $s^{(i)}_{hw} + c_i$. Since the ReLU function is monotone and $b_i > c_i$, we have that $\sigma(s + b_i) - \sigma(s + c_i) \ge 0 \ \forall s \in \mathbb{R}$. The target logit averages these nonnegative differences over all pairs and spatial locations:

$$\begin{aligned} f_y(x) &= \Gamma + T(x), \\ T(x) &:= \frac{1}{P H_{\text{sp}} W_{\text{sp}}} \sum_{i=1}^{P} \sum_{h,w} \left[ \sigma(s^{(i)}_{hw} + b_i) - \sigma(s^{(i)}_{hw} + c_i) \right], \\ f_k(x) &= 0 \quad \text{for all } k \ne y. \end{aligned} \tag{3}$$

Thus $f_y(x) \ge \Gamma > 0$ for all inputs, giving a global robustness certificate. The construction becomes difficult when both channels in many pairs are unstable over the perturbation set. A relaxation that treats $\sigma(s + b_i)$ and $\sigma(s + c_i)$ independently may produce a negative lower bound on their difference, even though the true difference is always nonnegative because the two ReLUs share the same scalar $s$. The number of pairs $P$, size of feature map $H_{\text{sp}} \times W_{\text{sp}}$, bias gap $b_i - c_i$, and margin constant $\Gamma$ control the accumulated relaxation error. Appendix B.5 gives the formal robustness proof.

### 2.3. Attention Module Constructors

For attention module-based constructors, inputs are sequences $X \in \mathbb{R}^{n \times d_{\text{tok}}}$, with rows $x^{(i)} \in \mathbb{R}^{d_{\text{tok}}}$ as tokens. Perturbations are measured in the entrywise norm $\|X - X_0\|_\infty = \max_{i,j} |X_{ij} - (X_0)_{ij}|$. We write $Q(X) = XW_Q$, $K(X) = XW_K$, and $V(X) = XW_V$ for the query, key, and value projections, where $W_Q \in \mathbb{R}^{d_{\text{tok}} \times d_q}$, $W_K \in \mathbb{R}^{d_{\text{tok}} \times d_k}$, and $W_V \in \mathbb{R}^{d_{\text{tok}} \times d_v}$ are learned matrices.

**Fixed-Ordering Softmax Attention.** For single-head softmax attention, robustness can be certified by showing that the score ordering is stable and the resulting attention output cannot drift enough to change the class. Define the score kernel $M = W_Q W_K^\top / \sqrt{d_k}$, and let $S(X) = XMX^\top$ with $S_{ij}(X) = (x^{(i)})^\top M x^{(j)}$. The attention output is $\text{Attn}(X) = A(X)XW_V$, where $A(X)$ is the row-wise softmax of $S(X)$, and the full classifier is $f = h \circ \text{Attn}$ for some downstream head $h$. Let $\pi_i^\star = \arg\max_j S_{ij}(X_0)$ denote the nominal row-wise score maximizer in row $i$, assuming it is unique, and define the score gaps $\Delta_{ij} = S_{i,\pi_i^\star}(X_0) - S_{ij}(X_0) > 0$ for $j \ne \pi_i^\star$. A bilinear expansion of the score differences, derived in Appendix B.6, gives:

$$\left| \left[ S_{i,\pi_i^\star}(X) - S_{ij}(X) \right] - \Delta_{ij} \right| \le \epsilon C_{ij}, \tag{4}$$

for all $\|X - X_0\|_\infty \le \epsilon$, where $C_{ij}$ depends on $\|M\|_{\text{op}}$, the nominal token norms, $d_{\text{tok}}$, and $\epsilon$. If $\Delta_{ij} > \epsilon C_{ij}$ for every $i$ and $j \ne \pi_i^\star$, then row-wise score maximizers remain fixed throughout $\mathcal{B}_\epsilon(x_0)$.

Appendix B.6 also provides a perturbation bound for the attention output: there exists a constant $L_{\text{attn}}$ such that $\|\text{Attn}(X) - \text{Attn}(X_0)\|_F \le \sqrt{n} L_{\text{attn}} \epsilon \ \forall X \in \mathcal{B}_\epsilon(X_0)$. Letting $\mu(X_0) = \min_{k \ne y} (f_y(X_0) - f_k(X_0))$ denote the nominal classification margin, if each logit of the downstream head $h$ is $L_h$-Lipschitz with respect to the Frobenius norm, then each class-wise margin can decrease at most by $2L_h \sqrt{n} L_{\text{attn}} \epsilon$. Hence, if $\mu(X_0) > 2L_h \sqrt{n} L_{\text{attn}} \epsilon$, then $f$ is robust on $\mathcal{B}_\epsilon(X_0)$.

**Dominant-Key Linear Attention.** This constructor uses linear attention, where the softmax score kernel is replaced by a nonnegative feature-map inner product. Let $q^{(i)}(X) = Q(X)[i, :]$ and $k^{(j)}(X) = K(X)[j, :]$. For a nonnegative feature map $\phi$ with positive row sums, define:

$$w_{ij}(X) = \langle \phi(q^{(i)}(X)), \phi(k^{(j)}(X)) \rangle \ge 0. \tag{5}$$

The normalized attention weights are then $\alpha_{ij}(X) =$

$\frac{w_{ij}(X)}{\sum_{\ell=1}^{n} w_{i\ell}(X)}$, and the module's output is:

$$\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] = \sum_{j=1}^{n} \alpha_{ij}(X) V(X)[j,:]. \tag{6}$$

Instead of certifying softmax score-order stability, this constructor enforces a dominant key per query row. Fix a dominant key $j_i^\star$ for each row and suppose that throughout the perturbation set we have:

$$w_{ij_i^\star}(X) \geq \rho_i \sum_{j \neq j_i^\star} w_{ij}(X). \tag{7}$$

This condition says that, for query row $i$, the unnormalized attention weight assigned to key $j_i^\star$ is at least $\rho_i$ times the total weight assigned to all other keys. Letting $\rho = \min_i \rho_i$, we therefore have that row $i$ assigns at least $\frac{\rho_i}{(1+\rho_i)}$ normalized attention mass to $j_i^\star$. Since the attention output is a weighted average of value vectors, the row output remains close to the value vector $V(X)[j_i^\star,:]$ associated with the dominant key. The perturbation proof in Appendix B.7 shows:

$$\|\mathrm{Attn}_{\mathrm{lin}}(X) - \mathrm{Attn}_{\mathrm{lin}}(X_0)\|_{2,\infty} \leq \frac{2}{1+\rho} D_V(X_0) + \left(1 + \frac{2}{1+\rho}\right) \epsilon \sqrt{d_{\mathrm{tok}}} \|W_V\|_{\mathrm{op}}. \tag{8}$$

where:

$$D_V(X_0) = \max_i \max_{j \neq j_i^\star} \|V(X_0)[j,:] - V(X_0)[j_i^\star,:]\|_2 \tag{9}$$

is the largest nominal distance, over query rows, between the value vector associated with the dominant key $j_i^\star$ and any other. Thus, if $f = h \circ \mathrm{Attn}_{\mathrm{lin}}$, each logit of $h$ is $L_h$-Lipschitz in Frobenius norm, and the nominal margin $\mu(X_0)$ is larger than twice the induced logit drift, then the classifier is robust. For this constructor, difficulty can be controlled through the dominance ratio $\rho$, value spread $D_V(X_0)$, and sequence length.

### 2.4. Polynomial Network Constructor

The polynomial network constructor generates candidate robust instances by explicitly controlling the geometry of the decision boundary. We consider polynomial classifiers, such as feedforward networks with affine layers and monomial activations, so that each logit is a polynomial in the input. In the binary classification case, let $\mu(x) = f_1(x) - f_2(x)$ be the margin polynomial and let $D = \{x : \mu(x) = 0\}$ be the decision boundary. Thus $D$ is a real algebraic hypersurface. The constructor samples a smooth boundary point $p \in D$, chooses a norm-adapted normal direction $u(p)$, and places the nominal input at $x_0 = p + (\epsilon + \delta)u(p)$, where $\delta > 0$ is a boundary buffer. The sampled point $p$ is therefore a known boundary point just outside the perturbation set $B_\epsilon(x_0)$. The local placement step creates a near-boundary instance, but the ground-truth robustness label comes from a global separation check certifying that $B_\epsilon(x_0) \cap D = \emptyset$. This rules out another branch or component of the decision boundary entering the perturbation set. When the check succeeds, $\mu$ has constant sign on $B_\epsilon(x_0)$, so the instance is robust. The buffer $\delta$ controls proximity to failure: decreasing $\delta$ makes the perturbation set nearly touch the decision boundary, forcing verifiers to certify a small positive separation from a curved polynomial hypersurface. The main difficulty parameters are the input dimension, polynomial degree, hidden width, and boundary buffer $\delta$. Appendix B.8 gives the full construction, boundary-sampling procedure, and nearest-boundary certificate.

## 3. The Verification Instance Difficulty Profile

In addition to the VeriStress-GT framework, we move towards a quantifiable characterization of verification instance difficulty via Difficulty Profiles. Such quantification allows one to compare instances beyond binary robust and non-robust labels. The components of the Difficulty Profile are chosen to satisfy design principles ensuring computability, applicability, and interpretability (see Appendix F for an explanation of design principles and candidate components tested). Based on the taxonomy of verifier certificate styles (Appendix D) and design principles, we define the profile:

**Definition 3.1.** The *Difficulty Profile* of a verification instance $(f, x_0, y, \mathcal{B}_\epsilon)$ is the tuple:

$$\mathbf{D}(f, x_0, y, \mathcal{B}_\epsilon) = \left(\widehat{M}_{\min},\ G_{\mathrm{IBP}},\ U,\ A_\tau,\ d_{\mathrm{eff}}\right). \tag{10}$$

Each component, defined below, targets a different axis of verification difficulty: $\widehat{M}_{\min}$ measures the amount of empirical margin slack to certify, $G_{\mathrm{IBP}}$ measures how much of this slack is lost under a coarse interval relaxation, $U$ measures the fraction of nonlinear units whose phase remains ambiguous over the input set, $A_\tau$ measures the local regions encountered in the input set, and $d_{\mathrm{eff}}$ measures the effective dimensionality of the input directions along which the margin is sensitive.

**Component 1: Minimum Margin ($\widehat{M}_{\min}$).** Proximity to violation is a basic, intuitive source of verification difficulty as instances with small positive margin may require substantial branching or bound refinement. Since $\min_{x \in \mathcal{B}_\epsilon(x_0)} \mu(x)$ is generally nonconvex and intractable, we estimate it via a sampling procedure. Let $\mathcal{S} = \{x_1, \ldots, x_N\} \subset \mathcal{B}_\epsilon(x_0)$ denote a set of profiling samples, drawn from a mixture of uniform and boundary-biased samples. Then, we define:

$$\widehat{M}_{\min} := \min_{x_i \in \mathcal{S}} \mu(x_i) \tag{11}$$

A small, positive $\widehat{M}_{\min}$ suggests that the instance is close to a counterexample, while $\widehat{M}_{\min} < 0$ indicates that sampling

already found one. We note that $\widehat{M}_{\min}$ is not designed to be a certificate of robustness, but rather a verifier-agnostic proxy for proximity to the decision boundary.

**Component 2: IBP Relative Gap ($G_{\text{IBP}}$).** Relaxation-based verifiers certify robustness via tractable lower bounds on $\mu(x)$. We therefore use IBP as an efficient, architecture-agnostic proxy for relaxation looseness. Let $L_{\text{IBP}}$ denote the IBP lower bound over $\mathcal{B}_\epsilon(x_0)$. Define the relative gap as:

$$G_{\text{IBP}} := \frac{\widehat{M}_{\min} - L_{\text{IBP}}}{|\widehat{M}_{\min}| + \eta}, \tag{12}$$

where $\eta > 0$ is a small numerical stability constant. The numerator estimates the margin slack lost by IBP by the best sampled margin, while the denominator normalizes by the empirical margin scale. Hence larger $G_{\text{IBP}}$ indicates increased instance difficulty, particularly for relaxation-based verifiers.

**Component 3: Unstable Fraction ($U$).** The unstable fraction measures how many nonlinear neurons cannot be treated as fixed or locally affine over $\mathcal{B}_\epsilon(x_0)$. For ReLU networks, this is the fraction of neurons whose IBP pre-activation interval $[\ell_j, u_j]$ crosses zero, i.e., $\ell_j < 0 < u_j$. For smooth nonlinearities, we use the same principle by marking coordinate $j$ as effectively nonlinear when the activation slope varies sufficiently over its IBP interval: $\omega_j := \sup_{s,t\in[\ell_j,u_j]} |\phi_j'(s) - \phi_j'(t)| > \tau$ for a fixed threshold $\tau > 0$. Let $\mathcal{J}$ denote the set of nonlinear coordinates. We define:

$$U := \frac{1}{|\mathcal{J}|} \sum_{j\in\mathcal{J}} \mathbb{1}_{\{\omega_j > \tau\}}. \tag{13}$$

For ReLU neurons, this reduces exactly to the unstable-neuron fraction. Larger $U$ indicates that more units require nontrivial relaxation or case splitting.

**Component 4: Local-Region Complexity ($A_\tau$).** The unstable fraction $U$ measures individually nonlinear coordinates, but not how many distinct local behaviors are realized in $\mathcal{B}_\epsilon(x_0)$. To capture this, we view local-region complexity as an affine-covering problem. For tolerance $\delta > 0$, define:

$$\begin{aligned} N_{\text{aff}}(\delta) := \min\Big\{ m : \mathcal{B}_\epsilon(x_0) \subseteq \textstyle\bigcup_{i=1}^m U_i, \\ \exists (a_i, b_i) \text{ such that, for each } i, \\ \sup_{x\in U_i} \left|\mu(x) - a_i^\top x - b_i\right| \le \delta \Big\}. \end{aligned} \tag{14}$$

where the $U_i$ range over convex subsets of $\mathcal{B}_\epsilon(x_0)$. Let $L_c := \sup_{x\in\mathcal{B}_\epsilon(x_0)} \sup_{\xi\in\partial_c\mu(x)} \|\xi\|_{p,*}$ denote a local Lipschitz constant for the margin over $\mathcal{B}_\epsilon(x_0)$, where $\partial_c\mu(x)$ is the Clarke subdifferential and $\|\cdot\|_{p,*}$ is dual to $\|\cdot\|_p$. Set $A_\tau^\star := \log N_{\text{aff}}(\tau\epsilon L_c)$. Therefore, $A_\tau^\star$ counts how many affine functions are needed to approximate the margin over the perturbation set with resolution $\tau$. For ReLU networks, this corresponds to an effective count of affine regions, while for networks with smooth activation functions it counts the number of local Taylor approximations needed to approximate the curved margin landscape. The following proposition motivates our empirical estimator by showing that affine-cover complexity is governed by how much the normalized gradient can vary over $\mathcal{B}_\epsilon(x_0)$:

**Proposition 3.2** (Smooth affine-cover bound)**.** *Assume that $\mu(x)$ is differentiable on $\mathcal{B}_\epsilon(x_0)$ and that its gradient is $\beta$-Lipschitz. Let $\widetilde{\beta} := \frac{\beta\epsilon}{L_c(\mu)}$ be the normalized gradient-variation scale. Then, up to the standard covering-number constant for the chosen norm, $A_\tau^\star \le d \log\left(1 + \sqrt{2\widetilde{\beta}/\tau}\right)$.*

See Appendix C for the proof. Directly estimating the global gradient-Lipschitz scale $\widetilde{\beta}$ is conservative in practice, so we instead measure the realized variation in local linear behavior by sampling normalized gradients. For each $x_i \in \mathcal{S}$, let $q_\tau(x_i) := \text{Quantize}_\tau(\nabla\mu(x_i)/L_c)$, where $\text{Quantize}_\tau(v)$ denotes coordinatewise quantization of $v$ onto a grid of width $\tau$. We define:

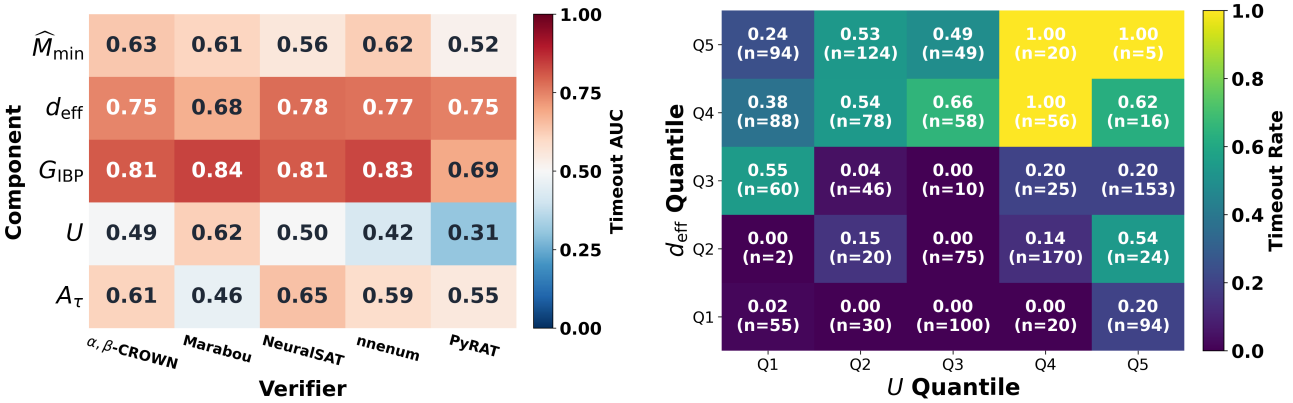
$$A_\tau := \log |\{q_\tau(x_i) : x_i \in \mathcal{S}\}|. \tag{15}$$

$A_\tau$ is a sampled log-count of distinct normalized local affine behaviors, as it estimates the realized local complexity suggested by the affine-cover definition. Larger $A_\tau$ indicates more heterogeneous local behavior, making coarse relaxations, abstractions, or partitions less likely to remain tight.

**Component 5: Effective Gradient dimensionality ($d_{\text{eff}}$).** Even when the margin is sensitive to perturbations, the structure of this sensitivity matters. If sensitivity is concentrated in a few input coordinates, splitting or search may isolate it quickly, but if it is spread across many, the instance is effectively high-dimensional. Let $\nabla\mu(x)$ denote the input gradient of the margin where it exists, using any measurable subgradient at non-smooth points. Define the average effective gradient dimension:

$$d_{\text{eff}} := \frac{1}{|\mathcal{S}|} \sum_{x_i\in\mathcal{S}} \frac{\|\nabla\mu(x_i)\|_1^2}{\|\nabla\mu(x_i)\|_2^2 + \eta}, \tag{16}$$

where $\eta > 0$ is a small constant for numerical stability. This quantity is related to the Participation Ratio (PR) for measuring dimensionality (Gao et al., 2017), and is near 1 when the gradient is concentrated on one coordinate and near $r$ when it is spread evenly across $r$ coordinates. Larger $d_{\text{eff}}$ therefore indicates more distributed margin sensitivity, which is especially relevant for $\ell_\infty$ perturbations since the first-order margin decrease scales with $\epsilon\|\nabla\mu(x)\|_1$.




*(a)* Profile correlations

*(b)* $U \times d_{\text{eff}}$

*Figure 2.* (left) Timeout AUC for each profile component across all benchmarks, treating timed-out instances as positives and solved instances as negatives; values above 0.5 indicate that larger component values are associated with timeouts. (right) Binned timeout rate over unstable fraction $U$ and effective gradient dimension $d_{\text{eff}}$, showing that timeouts can depend on interactions between profile components. For example, we see $U$ have little effect on timeout rate when $d_{\text{eff}}$ is small.

## 4. Numerical Experiments and Verification Study

We evaluate 5 top-scoring verifiers on 225 VeriStress-GT instances spanning all constructors in Section 2, plus two external robustness benchmarks: MNIST_fc, with fully connected ReLU networks trained on MNIST (Bak et al., 2021; Lecun et al., 1998), and oval21, with ReLU CNNs trained on MNIST and CIFAR-related tasks (Bak et al., 2021; Krizhevsky, 2009). The VeriStress-GT instances form a parameter-diverse stress-test suite, though the framework can generate unlimited instances. We compute Difficulty Profile estimates for all instances.

All experiments were run on an Intel Xeon Gold 6152 CPU server with 32 CPU cores, using per-instance timeouts of 600 seconds for VeriStress-GT and 360 seconds for external benchmarks. Appendix G gives framework instantiation details and aggregate verifier outcomes. Figure 3 summarizes verifier runtimes, while Figure 2 shows how Difficulty Profile components and interactions discriminate verifier timeouts. We highlight three main takeaways:

**1. Ground-truth labels reveal incorrect robustness claims.** Despite all instances in this instantiation of the VeriStress-GT framework containing analytic proofs of UNSAT (i.e., robust) labels, multiple verifiers return false SAT (i.e., non-robust) claims. Many of these incorrect claims can primarily be attributed to numerical tolerance limitations. For instance, setting the perturbation radius $\epsilon$ equal to 0.999 times the exact robustness radius computed via the MILP approach led to such SAT claims for a number of verifiers. However, when separately testing verifiers with an Exact-Radius MILP construction containing provably non-robust instances (i.e., a perturbation radius $\epsilon$ larger than the network's exact adversarial radius), we discovered a bug in a popular verifier's handling of disjunctive output constraints, leading to false UNSAT claims which have since been corrected after collaboration with developers (see Appendix E for details). Incorrect claims for verification can be impactful; false UNSAT claims can mislead practitioners into believing a model is certifiably safe when it is not, and false SAT claims distort verifier evaluation. VeriStress-GT helps expose these failures and highlights the need for a careful review of numerical tolerance limits.

**2. Difficulty Profiles help make verifier timeouts diagnosable.** Without Difficulty Profiles, analysis generally ends at measuring the number of timeouts; some instances are certified, others are not, and the reason largely remains opaque. Difficulty Profiles turn these outcomes into diagnosable failure modes. For example, Figures 5 and 6 in Appendix G show that the final 16 MNIST_fc instances have $G_{\text{IBP}} \geq 2 \times 10^5$, several orders of magnitude larger than the remaining MNIST_fc instances and the corresponding values in VeriStress-GT or oval21, suggesting extreme interval-relaxation looseness as the dominant obstruction. In oval21, instances 10–16 are certified by at most two verifiers each and all have $d_{\text{eff}} > 1000$, more than twice any value observed in MNIST_fc, suggesting high-dimensional margin sensitivity. These examples illustrate how Difficulty Profiles convert raw verifier outcomes into more actionable explanations, helping identify whether future progress should target tighter relaxations, better branching, dimension-aware heuristics, or other verifier improvements.

**3. Difficulty Profiles help turn VeriStress-GT into a targeted stress-test suite.** As VeriStress-GT instances scale, different constructors isolate different failure modes, making the benchmark useful not only for measuring performance but also for testing verifier improvements. For MEAP instances, increasing difficulty produces large relaxation

| Veristress-GT Constructor | $\alpha, \beta$-CROWN | Marabou | NeuralSAT | PyRAT | nnenum |
|---|---|---|---|---|---|
| MEAP | 13/14 Avg: 69.5s | 2/14 Avg: 507.7s | 6/14 Avg: 510.3s | 14/14 Avg: 21.1s | 0/14 avg — |
| MILP | 19/31 Avg: 242.4s | 19/31 Avg: 228.9s | 27/31 Avg: 126.9s | 19/31 Avg: 98.9s | 24/31 Avg: 136.1s |
| Input-Corner Stress | 19/22 Avg: 169.9s | 12/22 Avg: 274.0s | 19/22 Avg: 160.8s | 19/22 Avg: 93.6s | 0/22 avg — |
| Paired-Biases | 36/46 Avg: 176.9s | 34/46 Avg: 167.5s | 35/46 Avg: 236.4s | 36/46 Avg: 133.9s | 31/46 Avg: 204.0s |
| Constant-on-Box | 5/5 Avg: 3.6s | 5/5 Avg: 5.6s | 5/5 Avg: 5.3s | 5/5 Avg: 3.9s | 0/5 avg — |
| Deep Contractive | 50/50 Avg: 61.0s | 50/50 Avg: 8.3s | 50/50 Avg: 232.4s | 50/50 Avg: 4.1s | 50/50 Avg: 2.0s |
| Dominant Key Attn | 18/18 Avg: 95.6s | 0/18 avg — | 18/18 Avg: 69.7s | 0/18 avg — | 0/18 avg — |
| Fixed Order Attn | 17/17 Avg: 137.8s | 0/17 avg — | 17/17 Avg: 140.1s | 17/17 Avg: 6.1s | 0/17 avg — |
| Polynomial Net | 16/22 Avg: 111.2s | 0/22 avg — | 11/22 Avg: 140.4s | 0/22 Avg: 360.0s | 0/22 avg — |
| MNIST_fc | 41/90 Avg: 224.8s | 38/90 Avg: 211.5s | 35/90 Avg: 266.8s | 47/90 Avg: 162.7s | 37/90 Avg: 209.1s |
| oval21 | 15/30 Avg: 237.1s | 20/30 Avg: 110.6s | 18/30 Avg: 160.9s | 8/30 Avg: 283.9s | 3/30 Avg: 328.9s |

Verifier

All UNSAT | Mixed | All Timeout/Error

*Figure 3.* Runtime outcomes for 5 verifiers on VeriStress-GT constructors and two external benchmarks, shown in the last two rows. Average runtimes include timeouts of 600 seconds for VeriStress-GT and 360 seconds for external benchmarks. Fractions report correct results over total instances; missing runtimes indicate unsupported computation graphs.

gaps $G_{\text{IBP}}$, high instability $U$, and large effective gradient dimensionality $d_{\text{eff}}$. These instances therefore form a targeted stress test for methods that must reason about activation-pattern ambiguity, branching decisions, and loose intermediate bounds. In contrast, the Corners construction has $A_\tau = 0$ and $d_{\text{eff}} \approx 0$, so it is structurally simple from an activation-pattern perspective. However, verifiers slow or timeout as $G_{\text{IBP}}$ grows, showing that relaxation looseness and geometric bound growth can be isolated from nonlinear-region complexity.

These distinctions make the profiles actionable. A new branching heuristic can be evaluated on MEAP instances, a tighter relaxation can be tested on Corners instances, and depth-aware abstractions can be tested on Paired-Bias CNNs. Rather than treating all timeouts as equivalent failures, Difficulty Profiles reveal which instances are useful for stress-testing specific verifier capabilities.

## 5. Conclusion, Limitations, and Future Work

We introduced VeriStress-GT, a modular framework for constructing neural network verification instances with verifier-independent ground-truth robustness labels. Together with the Difficulty Profile, VeriStress-GT enables targeted stress-testing and makes verifier timeouts more diagnosable. It also exposed numerical tolerance issues and implementation bugs, underscoring the need for ground-truth labels in reliable verifier evaluation.

Several limitations and opportunities for future work remain. Due to computational costs, the verifiers were tested with a limited number of repeated seeds. Moreover, we do not claim that Difficulty Profiles are the sole source of predictive power for verifier runtime; instead, we aim for a collection of complementary, succinct, interpretable, and predictive components.

Potential avenues for future work include extending the framework to additional architectures, such as residual networks and graph neural networks. Difficulty Profiles may also help guide and test verifiers for modern language-model components, where component-specific verifiers and compositional certificates could improve scalability.

## References

Yulia Alexandr, Hao Duan, and Guido Montúfar. Robustness verification of polynomial neural networks. *arXiv preprint arXiv:2602.06105*, 2026.

Stanley Bak. nnenum: Verification of ReLU neural networks with optimized abstraction refinement. In *NASA Formal Methods: 13th International Symposium, NFM 2021, Virtual Event, May 24–28, 2021, Proceedings*, page 19–36, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-76383-1. doi: 10.1007/

978-3-030-76384-8_2. URL https://doi.org/10.1007/978-3-030-76384-8_2.

Stanley Bak, Changliu Liu, and Taylor Johnson. The second international verification of neural networks competition (vnn-comp 2021): Summary and results, 2021. URL https://arxiv.org/abs/2109.00498.

Nathan Drenkow, Numair Sani, Ilya Shpitser, and Mathias Unberath. A systematic review of robustness in deep learning for computer vision: Mind the gap?, 2022. URL https://arxiv.org/abs/2112.00639.

Hai Duong, ThanhVu Nguyen, and Matthew Dwyer. A dpll(t) framework for verifying deep neural networks, 2024. URL https://arxiv.org/abs/2307.10266.

Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. DPLL(T): Fast decision procedures. In *Computer Aided Verification*, pages 175–188, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, 2017. doi: 10.1101/214262. URL https://www.biorxiv.org/content/early/2017/11/12/214262.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv:1810.12715*, 2018. URL http://arxiv.org/abs/1810.12715.

Kai Jia and Martin C. Rinard. Exploiting verified neural networks via floating point numerical error. *CoRR*, abs/2003.03021, 2020. URL https://arxiv.org/abs/2003.03021.

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.

Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification*, pages 443–452, Cham, 2019. Springer International Publishing.

Konstantin Kaulen, Tobias Ladner, Stanley Bak, Christopher Brix, Hai Duong, Thomas Flinkow, Taylor T. Johnson, Lukas Koller, Edoardo Manino, ThanhVu H Nguyen, and Haoze Wu. The 6th international verification of neural networks competition (vnn-comp 2025): Summary and results, 2025. URL https://arxiv.org/abs/2512.19007.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Linhan Li and ThanhVu Nguyen. Destabilizing neurons to generate challenging neural network verification benchmarks. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1351–1363, 2025. doi: 10.1109/ASE63991.2025.00115.

Jia Liu and Yaochu Jin. A comprehensive survey of robust deep learning in computer vision. *Journal of Automation and Intelligence*, 2(4):175–195, 2023. URL https://www.sciencedirect.com/science/article/pii/S294985542300045X.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. URL https://arxiv.org/abs/1706.06083.

Mark Huasong Meng, Guangdong Bai, Sin Gee Teo, Zhe Hou, Yan Xiao, Yun Lin, and Jin Song Dong. Adversarial robustness of deep neural networks: A survey from a formal verification perspective. *IEEE Transactions on Dependable and Secure Computing*, page 1–1, 2024. URL http://dx.doi.org/10.1109/TDSC.2022.3179131.

Asiye K Ozcanli, Fatma Yaprakdal, and Mustafa Baysal. Deep learning methods and applications for electrical power systems: A comprehensive review. *International Journal of Energy Research*, 44(9):7136–7157, 2020.

Stanford Institute for Human-Centered Artificial Intelligence. AI index report 2025, 2025. URL https://hai.stanford.edu/assets/files/hai_ai_index_report_2025.pdf.

Vincent Tjeng and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *CoRR*, abs/1711.07356, 2017. URL http://arxiv.org/abs/1711.07356.

Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyGIdiRqtm.

Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems, 2020. URL https://arxiv.org/abs/2004.05519.

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. In *Advances in Neural Information Processing Systems*, volume 34, pages 29909–29921, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/fac7fead96dafceaf80c1daffeae82a4-Paper.pdf.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/358f9e7be09177c17d0d17ff73584307-Paper.pdf.

Dong Xu, David Shriver, Matthew B. Dwyer, and Sebastian Elbaum. Systematic generation of diverse benchmarks for dnn verification. In *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I*, page 97–121, Berlin, Heidelberg, 2020a. Springer-Verlag. ISBN 978-3-030-53287-1. doi: 10.1007/978-3-030-53288-8_5. URL https://doi.org/10.1007/978-3-030-53288-8_5.

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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1129–1141, 2020b. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0cbc5671ae26f67871cb914d81ef8fc1-Paper.pdf.

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, 2021. URL https://arxiv.org/abs/2011.13824.

Jianbo Yu and Yue Zhang. Challenges and opportunities of deep learning-based process fault detection and diagnosis: a review. *Neural Computing and Applications*, 35(1):211–252, 2023.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions, 2018. URL https://arxiv.org/abs/1811.00866.

Xingjian Zhou, Keyi Shen, Andy Xu, Hongji Xu, Cho-Jui Hsieh, Huan Zhang, and Zhouxing Shi. SoundnessBench: A soundness benchmark for neural network verifiers. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=UuYYldVLH3.

Dániel Zombori, Balázs Bánhelyi, Tibor Csendes, István Megyeri, and Márk Jelasity. Fooling a complete neural network verifier. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=4IwieFS44l.

| Feature | GDVB (Xu et al., 2020a) | SoundnessBench (Zhou et al., 2025) | ReluSplitter (Li and Nguyen, 2025) | VeriStress-GT (ours) |
|---|---|---|---|---|
| Automatic Benchmark Generation | ✓ | ✓ | ✓ | ✓ |
| Controllable Difficulty Variation | Architecture Only | SAT Focused | Instability Only | ✓ |
| Number of Difficulty Controls | 4 model factors | Few CEX parameters | Neuron Instability | 5+ Per Constructor |
| Exact Correctness Scoring | × | SAT Only | × | ✓ |
| Analytic Control of Margin/Slack structure | × | Trained Heuristic | × | ✓ |
| Supports ReLU Networks | ✓ | ✓ | ✓ | ✓ |
| Supports CNNs | ✓ | ✓ | ✓ | ✓ |
| Supports Attention Architectures | × | × | × | ✓ |
| Supports Polynomial Networks | × | × | × | ✓ |

*Table 1.* Comparison between VeriStress-GT and existing benchmark-generation or verifier-testing frameworks.

## A. Comparison with Existing Verification Benchmarks

The ReluSplitter benchmark generation approach (Li and Nguyen, 2025) shares with VeriStress-GT the high-level goal of constructing verification instances that systematically induce verifier difficulty. Their method specifically targets neuron instability by modifying or partitioning networks so that many ReLU activations remain unstable over the verification region. We note that most constructors in the VeriStress-GT framework explicitly contain specific methods to gradually increase the fraction of unstable neurons, as seen in our code repository. Additionally, each constructor in VeriStress-GT is not restricted to a single source of hardness and instead possesses multiple knobs to gradually increase difficulty.

Another relevant work is GDVB (Generalized/Diverse Generation Verification Benchmark) (Xu et al., 2020a), which generates families of verification benchmarks by varying characteristics specific to the trained model $f$ believed to influence verifier difficulty, such as number of network layers and input dimension. VeriStress-GT differs fundamentally in that its constructions are designed around verifier-independent ground-truth guarantees, enabling exact scoring of verifier correctness and controlled analysis of false SAT/UNSAT claims independent of verifier consensus. Additionally, whereas GDVB studies verification difficulty factors solely via properties of the model (the four studied factors are input dimension, number of layers, types of layers, and number of neurons), the difficulty profile component study in our work instead evaluates factors inherent to the entire verification tuple $(f, x_0, y, \mathcal{B}_\epsilon)$.

SoundnessBench (Zhou et al., 2025) shares a related motivation to VeriStress-GT in attempting to evaluate verifier soundness. Their approach trains networks with intentionally inserted adversarial examples that are difficult for standard attacks to locate. In contrast, VeriStress-GT constructions directly encode provable robustness or violation structure through analytic mechanisms such as invariant projections, activation-pattern constraints, contraction arguments, or exact optimization certificates. While SoundnessBench focuses on verifiers' abilities to correctly claim SAT (non-robust), VeriStress-GT utilizes a suite of techniques to study verifiers' overall soundness (both robust and non-robust) while simultaneously gradually stress-testing verifiers.

Traditional verification benchmarks included in VNN-COMP collections are valuable for evaluating practical verifier scalability. However, these benchmarks generally lack verifier-independent ground-truth labels. Consequently, correctness evaluation often depends implicitly on verifier agreement or the absence of discovered counterexamples.

Table 1 compares VeriStress-GT with other verification benchmarks.

## B. Robust Constructor Details

This appendix provides the implementation details for VeriStress-GT's robust constructors described in Section 2.

### B.1. Exact-Radius via MILP

Let an $L$-layer ReLU MLP have preactivations $s^{(\ell)} = W^{(\ell)} z^{(\ell-1)} + b^{(\ell)}$ and activations $z^{(\ell)} = \sigma(s^{(\ell)})$, with $z^{(0)} = x_0 + \delta$ and $\sigma(t) = \max\{t, 0\}$. Given valid preactivation bounds $l_j^{(\ell)} \le s_j^{(\ell)} \le u_j^{(\ell)}$ obtained via simple interval arithmetic or Interval Bound Propagation (Gowal et al., 2018), each ReLU node is encoded using a binary variable $a_j^{(\ell)} \in \{0, 1\}$ and standard big-$M$ constraints. The interval bounds are used only to define valid big-$M$ constants and do not relax the network semantics. With binary decision variables, the constraints encode the ReLU graph exactly. For a fixed target class $k \neq y$, the

minimum $\ell_\infty$ perturbation required to make class $k$ match or exceed the true-class logit is obtained by solving the following program:

$$
\begin{aligned}
\min_{\delta,\, t,\, s,\, z,\, a} \quad & t && \text{(smallest adversarial radius)} \\
\text{s.t.} \quad & -t \le \delta_i \le t, \quad \forall i && (\ell_\infty \text{ ball}) \\
& z^{(0)} = x_0 + \delta && \text{(perturbed input)} \\
& s^{(\ell)} = W^{(\ell)} z^{(\ell-1)} + b^{(\ell)},\ \forall \ell && \text{(affine layers)} \\
& z_j^{(\ell)} \ge 0, \quad z_j^{(\ell)} \ge s_j^{(\ell)} && \text{(ReLU lower bounds)} \\
& z_j^{(\ell)} \le u_j^{(\ell)} a_j^{(\ell)} && \text{(inactive case)} \\
& z_j^{(\ell)} \le s_j^{(\ell)} - l_j^{(\ell)} (1 - a_j^{(\ell)}) && \text{(active case)} \\
& a_j^{(\ell)} \in \{0, 1\},\ \forall \ell, j && \text{(ReLU on/off)} \\
& f_k(x_0 + \delta) \ge f_y(x_0 + \delta) && \text{(misclassification).}
\end{aligned}
$$

The certified radius is $r^\star = \min_{k \neq y} t_k^\star$. Thus, for any chosen $\epsilon < r^\star$, the instance is robust by exact optimization.

### B.2. MEAP Constructor Details

We give the explicit parameterization used in the mutually exclusive activation pattern constructor. For each pair $p \in [P]$, choose a direction $w_p \in \mathbb{R}^d$ and a margin parameter $\gamma_p > 0$. Define

$$b_{p,1} = \gamma_p - w_p^\top x_0, \qquad b_{p,2} = \gamma_p + w_p^\top x_0. \tag{17}$$

Then, the paired preactivations satisfy

$$z_{p,1}(x_0) = z_{p,2}(x_0) = \gamma_p, \qquad b_{p,1} + b_{p,2} = 2\gamma_p > 0. \tag{18}$$

Thus both neurons are active at the nominal point, but the pair is constructed so that they cannot both be inactive at any input. To see this, observe that for any $x$,

$$z_{p,1}(x) + z_{p,2}(x) = b_{p,1} + b_{p,2} = 2\gamma_p. \tag{19}$$

Therefore, the two preactivations cannot both be negative. Moreover,

$$
\begin{aligned}
r_p(x) &= \max\{\sigma(z_{p,1}(x)), \sigma(z_{p,2}(x))\} \\
&= \max\{0, z_{p,1}(x), z_{p,2}(x)\} \\
&\ge \frac{z_{p,1}(x) + z_{p,2}(x)}{2} = \gamma_p.
\end{aligned} \tag{20}
$$

Hence each pair contributes a global lower bound $\gamma_p$. Letting $\gamma = \min_{p \in [P]} \gamma_p$, we have $r_p(x) \ge \gamma$ for all $p$ and all $x$. Since the target logit is $f_y(x) = \min_{p \in [P]} r_p(x)$ and all competing logits are fixed to zero, it follows that

$$f_y(x) - f_k(x) \ge \gamma \qquad \forall x, \quad \forall k \neq y. \tag{21}$$

Thus the MEAP network is robust with margin at least $\gamma$ on $\mathcal{B}_\epsilon(x_0)$.

The construction is made difficult by choosing parameters so that both ReLU neurons in each pair are unstable over the perturbation box. For the $\ell_\infty$ ball $\mathcal{B}_\epsilon(x_0)$, the minimum values of the two affine preactivations satisfy

$$\min_{x \in \mathcal{B}_\epsilon(x_0)} z_{p,1}(x) = \gamma_p - \epsilon \|w_p\|_1, \qquad \min_{x \in \mathcal{B}_\epsilon(x_0)} z_{p,2}(x) = \gamma_p - \epsilon \|w_p\|_1. \tag{22}$$

Similarly, both preactivations have maximum value $\gamma_p + \epsilon\|w_p\|_1$ over the box. Therefore, if

$$0 < \gamma_p < \epsilon\|w_p\|_1, \tag{23}$$

then the interval bounds for both $z_{p,1}$ and $z_{p,2}$ cross zero. Each ReLU is individually unstable, even though the pair satisfies the exact lower bound (20). This creates the intended verifier stress, as relaxations that treat the two ReLUs independently may admit the infeasible relaxed state $\sigma(z_{p,1}) = \sigma(z_{p,2}) = 0$, while the true network prevents it. Finally, the aggregation operations can be implemented using ReLU identities. For scalars $a, b$, note:

$$\max\{a, b\} = b + \sigma(a - b), \qquad \min\{a, b\} = a - \sigma(a - b). \tag{24}$$

Thus the pairwise maxima $r_p(x)$ and the final minimum over pairs can be represented as a standard piecewise-linear ReLU network.

### B.3. Input-Corner Stress Constructor Details

We give the formal certificate for the input-corner construction. Let $A \subseteq [d]$ be an active coordinate set with $|A| = d_{act}$, and let $\mathcal{Q}_\epsilon(x_0; A)$ denote the projection of $\mathcal{B}_\epsilon(x_0)$ onto these active coordinates. Its vertices are:

$$\mathcal{V}_\epsilon(x_0; A) = \{(x_0)_A + \epsilon s : s_i \in \{-1, +1\} \text{ for } i \in A, s_i = 0 \text{ for } i \notin A\}. \tag{25}$$

The constructed logits depend only on $x_A$. We set $f_y(x) = 0$ and $f_k(x) = h_k(x_A) - \beta_k$ for $k \neq y$, where each $h_k : \mathbb{R}^{d_{act}} \to \mathbb{R}$ is convex on $\mathcal{Q}_\epsilon(x_0; A)$. For a desired margin $\gamma > 0$, define

$$\beta_k = \max_{v \in \mathcal{V}_\epsilon(x_0;A)} h_k(v) + \gamma. \tag{26}$$

**Proposition B.1** (Input-corner certificate)**.** *If each $h_k$ is convex on $\mathcal{Q}_\epsilon(x_0; A)$ and $\beta_k$ is chosen as in (26), then the constructed classifier is robust on $\mathcal{B}_\epsilon(x_0)$ with margin at least $\gamma$.*

*Proof.* For any $x \in \mathcal{B}_\epsilon(x_0)$, we have $x_A \in \mathcal{Q}_\epsilon(x_0; A)$. Since $\mathcal{Q}_\epsilon(x_0; A)$ is a box, $x_A$ can be written as a convex combination of its vertices:

$$x_A = \sum_{v \in \mathcal{V}_\epsilon(x_0;A)} \lambda_v v_A, \qquad \lambda_v \geq 0, \qquad \sum_{v \in \mathcal{V}_\epsilon(x_0;A)} \lambda_v = 1 \tag{27}$$

And by the convexity of $h_k$. we have:

$$h_k(x_A) \leq \sum_{v \in \mathcal{V}_\epsilon(x_0;A)} \lambda_v h_k(v_A) \leq \max_{v \in \mathcal{V}_\epsilon(x_0;A)} h_k(v_A). \tag{28}$$

Therefore, for any $x \in \mathcal{B}_\epsilon(x_0)$ and $k \neq y$:

$$\mu_k(x) = f_y(x) - f_k(x) = \beta_k - h_k(x_A) \geq \beta_k - \max_{v \in \mathcal{V}_\epsilon(x_0;A)} h_k(v) = \gamma. \tag{29}$$

Since this holds for every $k \neq y$, we have $\mu(x) = \min_{k \neq y} \mu_k(x) \geq \gamma$ throughout $\mathcal{B}_\epsilon(x_0)$. □

A ReLU implementation is obtained by choosing

$$h_k(u) = \sum_{j=1}^{J} c_{k,j} \sigma(a_j^\top u + b_j), \qquad c_{k,j} \geq 0 \tag{30}$$

Because $u \mapsto \sigma(a_j^\top u + b_j)$ is convex and nonnegative linear combinations preserve convexity, each $h_k$ is convex. To make the instance difficult, the constructor may place hinges near the nominal input by setting, for example, $b_j = -a_j^\top (x_0)_A$. Then the $j$-th hinge preactivation is zero at $x_0$, and its interval over the active-coordinate box satisfies the following:

$$a_j^\top (u - (x_0)_A) \in [-\epsilon\|a_j\|_1, \epsilon\|a_j\|_1], \tag{31}$$

so the corresponding ReLU is unstable under IBP bounds. Increasing the number of hinges $J$, the active dimension $d_{\text{act}}$, or the hinge scales $\|a_j\|_1$ increases the number and magnitude of unstable hinge contributions while preserving the robustness certificate.

### B.4. Contractive CNN Constructor Details

We give the formal certificate for the deep contractive CNN constructor. Write the convolutional feature extractor as $\Phi = C_D \circ C_{D-1} \circ \cdots \circ C_1 \circ P$ where $P$ is an optional front-end map and $C_1, \ldots, C_D$ are contractive convolutional blocks. Let $L_{\text{front}}$ denote the induced $\ell_\infty$ Lipschitz constant of $P$, with $L_{\text{front}} = 1$ if no front-end map is used. Assume each block $C_\ell$ is $\ell_\infty$-Lipschitz with constant at most $\lambda < 1$. Then, for all $x, x'$,

$$\|\Phi(x) - \Phi(x')\|_\infty \leq L_{\text{front}} \lambda^D \|x - x'\|_\infty. \tag{32}$$

This follows directly by composing the Lipschitz constants of $P, C_1, \ldots, C_D$.

Next, define the centered target logit by $f_y(x) = \Gamma + w_y^\top(\Phi(x) - \Phi(x_0))$ and set $f_k(x) = 0 \ \forall k \neq y$. The following Proposition defines our robustness claim for the constructor:

**Proposition B.2** (Contractive CNN certificate)**.** *If*

$$\Gamma > \|w_y\|_1 L_{\text{front}} \lambda^D \epsilon, \tag{33}$$

*then the classifier $\Phi$ is robust on $\mathcal{B}_\epsilon(x_0)$.*

*Proof.* For any $x = x_0 + \delta$ with $\|\delta\|_\infty \leq \epsilon$, Hölder's inequality and (32) give

$$\begin{aligned} \left| w_y^\top(\Phi(x) - \Phi(x_0)) \right| &\leq \|w_y\|_1 \|\Phi(x) - \Phi(x_0)\|_\infty \\ &\leq \|w_y\|_1 L_{\text{front}} \lambda^D \|x - x_0\|_\infty \\ &\leq \|w_y\|_1 L_{\text{front}} \lambda^D \epsilon. \end{aligned} \tag{34}$$

Therefore,

$$f_y(x) \geq \Gamma - \|w_y\|_1 L_{\text{front}} \lambda^D \epsilon > 0 = f_k(x) \qquad \forall k \neq y. \tag{35}$$

Thus, $\mu(x) = \min_{k \neq y}(f_y(x) - f_k(x)) > 0$ throughout $\mathcal{B}_\epsilon(x_0)$, so the instance is robust. □

The centered logit is convenient because the nominal target logit is exactly $\Gamma$ and all competing logits are fixed to zero. An un-centered linear logit head can also be used. Suppose

$$f_i(x) = b_i + w_i^\top \Phi(x), \qquad i \in [c]. \tag{36}$$

Then for each $k \neq y$, define the nominal class-wise margin $\mu_k(x_0) = f_y(x_0) - f_k(x_0)$. Then, $\mu_k(x) = \mu_k(x_0) + (w_y - w_k)^\top(\Phi(x) - \Phi(x_0))$.

Therefore, by the same Lipschitz argument:

$$\mu_k(x) \geq \mu_k(x_0) - \|w_y - w_k\|_1 L_{\text{front}} \lambda^D \epsilon. \tag{37}$$

Thus, the uncentered head is robust whenever

$$\mu_k(x_0) > \|w_y - w_k\|_1 L_{\text{front}} \lambda^D \epsilon \qquad \text{for all } k \neq y. \tag{38}$$

### B.5. Paired-Bias CNN Constructor Details

We give the formal certificate and instability parameterization for the paired-bias CNN constructor. Let $Z = \Psi(x)$ denote the feature map produced by any upstream convolutional layers. For each pair $i \in [P]$, the two channels share the same convolutional filter $\mathcal{W}_i$ and differ only in their biases. Write $s^{(i)} = \mathcal{W}_i \star Z \in \mathbb{R}^{H_{\text{sp}} \times W_{\text{sp}}}$ so that at spatial location $(h, w)$, the paired activations are $\sigma(s^{(i)}_{h,w} + b_i)$ and $\sigma(s^{(i)}_{h,w} + c_i)$, where $b_i > c_i$. The certificate follows from monotonicity of the ReLU function. For any $s \in \mathbb{R}$ and any $b_i > c_i$, we have $s + b_i > s + c_i$, and since $\sigma$ is monotone and nondecreasing, then:

$$\sigma(s + b_i) \geq \sigma(s + c_i). \tag{39}$$

Therefore, each paired difference is nonnegative:

$$\sigma(s + b_i) - \sigma(s + c_i) \geq 0 \qquad \forall s \in \mathbb{R}. \tag{40}$$

Next, define the logit outputs as the following:

$$f_y(x) = \Gamma + \frac{1}{PH_{\mathrm{sp}}W_{\mathrm{sp}}} \sum_{i=1}^{P} \sum_{h=1}^{H_{\mathrm{sp}}} \sum_{w=1}^{W_{\mathrm{sp}}} \left[ \sigma(s_{h,w}^{(i)} + b_i) - \sigma(s_{h,w}^{(i)} + c_i) \right], \qquad f_k(x) = 0 \quad \text{for all } k \neq y. \tag{41}$$

**Proposition B.3** (Paired-bias CNN certificate)**.** *If $\Gamma > 0$ and $b_i > c_i$ for every pair $i \in [P]$, then the classifier defined in* (41) *is robust with margin at least $\Gamma$ for every input $x$.*

*Proof.* By (40), every summand in (41) is nonnegative. Hence $f_y(x) \geq \Gamma$ for every input $x$. Since $f_k(x) = 0$ for all $k \neq y$, we have

$$f_y(x) - f_k(x) \geq \Gamma > 0$$

for every $x$ and every $k \neq y$. Therefore

$$\mu(x) = \min_{k \neq y} (f_y(x) - f_k(x)) \geq \Gamma,$$

so the classifier is robust on any perturbation set $\mathcal{B}_\epsilon(x_0)$. □

To make the instance difficult for relaxation-based verifiers, the constructor chooses biases so that many paired ReLUs are unstable over the perturbation set. Suppose interval propagation gives bounds

$$s_{h,w}^{(i)} \in [s_{\mathrm{lo},h,w}^{(i)}, s_{\mathrm{hi},h,w}^{(i)}]$$

for the shared convolutional response over $\mathcal{B}_\epsilon(x_0)$. In practice, we choose a channelwise center $t_i$, such as a spatial average of interval midpoints, and a bias half-gap $\Delta_i > 0$, and set

$$b_i = -t_i + \Delta_i, \qquad c_i = -t_i - \Delta_i. \tag{42}$$

Then $b_i - c_i = 2\Delta_i > 0$, so the monotonicity certificate remains valid. For a given spatial location $(h, w)$, the shifted intervals are

$$s_{h,w}^{(i)} + b_i \in [s_{\mathrm{lo},h,w}^{(i)} - t_i + \Delta_i,\ s_{\mathrm{hi},h,w}^{(i)} - t_i + \Delta_i], \tag{43}$$

and

$$s_{h,w}^{(i)} + c_i \in [s_{\mathrm{lo},h,w}^{(i)} - t_i - \Delta_i,\ s_{\mathrm{hi},h,w}^{(i)} - t_i - \Delta_i]. \tag{44}$$

When these intervals cross zero, the corresponding ReLUs are unstable under interval bounds. Therefore, the constructor can create many unstable paired activations while preserving the exact nonnegativity guarantee. Independent relaxations may lose the fact that the two ReLUs are functions of the same scalar $s_{h,w}^{(i)}$, and can therefore produce a negative lower bound on a quantity that is globally nonnegative.

### B.6. Fixed-Ordering Softmax Attention Constructor Details

The robustness certificate for the fixed-ordering attention constructor has two parts. First, we show that the nominal row-wise score maximizers remain unchanged throughout the perturbation set. This ensures that the attention score pattern is stable. Second, we bound the total change in the attention module (i.e. score matrix times the value matrix $XW_V$) output. This bound, combined with a Lipschitz downstream head and a sufficiently large nominal classification margin, gives the final robustness certificate. Thus, the first proposition certifies score-pattern stability, while the second proposition certifies classification robustness. Let $X = X_0 + \Delta X$, and write $\delta^{(i)}$ for the $i$-th row of $\Delta X$. We assume $\|\Delta X\|_\infty \leq \epsilon$, so $\|\delta^{(i)}\|_2 \leq \epsilon\sqrt{d_{\mathrm{tok}}}$ for every $i$.

**Part 1: Score-gap stability.** For any $j \neq \pi_i^\star$, since $X = X_0 + \Delta X$, we have

$$\begin{aligned} S_{i,\pi_i^\star}(X) - S_{ij}(X) &= (x_0^{(i)} + \delta^{(i)})^\top M \left[ (x_0^{(\pi_i^\star)} + \delta^{(\pi_i^\star)}) - (x_0^{(j)} + \delta^{(j)}) \right] \\ &= {x_0^{(i)}}^\top M \left( x_0^{(\pi_i^\star)} - x_0^{(j)} \right) + {\delta^{(i)}}^\top M \left( x_0^{(\pi_i^\star)} - x_0^{(j)} \right) \\ &\quad + {x_0^{(i)}}^\top M \left( \delta^{(\pi_i^\star)} - \delta^{(j)} \right) + {\delta^{(i)}}^\top M \left( \delta^{(\pi_i^\star)} - \delta^{(j)} \right). \end{aligned} \tag{45}$$

The first term is the nominal score gap,

$$x_0^{(i)\top} M \left( x_0^{(\pi_i^\star)} - x_0^{(j)} \right) = S_{i,\pi_i^\star}(X_0) - S_{ij}(X_0) = \Delta_{ij}.$$

Therefore,

$$\begin{aligned} &\left[ S_{i,\pi_i^\star}(X) - S_{ij}(X) \right] - \Delta_{ij} \\ &\quad = \delta^{(i)\top} M \left( x_0^{(\pi_i^\star)} - x_0^{(j)} \right) + x_0^{(i)\top} M \left( \delta^{(\pi_i^\star)} - \delta^{(j)} \right) + \delta^{(i)\top} M \left( \delta^{(\pi_i^\star)} - \delta^{(j)} \right). \end{aligned} \tag{46}$$

Using $|u^\top M v| \le \|M\|_{\mathrm{op}} \|u\|_2 \|v\|_2$, $\|\delta^{(i)}\|_2 \le \epsilon\sqrt{d_{\mathrm{tok}}}$, and $\|\delta^{(\pi_i^\star)} - \delta^{(j)}\|_2 \le 2\epsilon\sqrt{d_{\mathrm{tok}}}$, we obtain

$$\left| \left[ S_{i,\pi_i^\star}(X) - S_{ij}(X) \right] - \Delta_{ij} \right| \le \epsilon C_{ij}(\epsilon), \tag{47}$$

where

$$C_{ij}(\epsilon) = \sqrt{d_{\mathrm{tok}}}\, \|M\|_{\mathrm{op}} \left( \|x_0^{(\pi_i^\star)} - x_0^{(j)}\|_2 + 2\|x_0^{(i)}\|_2 \right) + 2\epsilon d_{\mathrm{tok}}\, \|M\|_{\mathrm{op}}. \tag{48}$$

**Proposition B.4** (Attention score-pattern stability)**.** *If $\Delta_{ij} > \epsilon C_{ij}(\epsilon)$ for every $i$ and every $j \neq \pi_i^\star$, then $\pi(X) = \pi^\star$ for every $X$ satisfying $\|X - X_0\|_\infty \le \epsilon$.*

*Proof.* By (47),

$$S_{i,\pi_i^\star}(X) - S_{ij}(X) \ge \Delta_{ij} - \epsilon C_{ij}(\epsilon) > 0 \tag{49}$$

for every $j \neq \pi_i^\star$. Hence the row-wise maximizer remains $\pi_i^\star$ for every row $i$. □

**Part 2: Attention-output perturbation.** We next bound the attention output. Let $A_0 = A(X_0)$, $\widetilde{A} = A(X)$, $V_0 = X_0 W_V$, and $\Delta V = \Delta X W_V$. The exact decomposition is

$$\mathrm{Attn}(X) - \mathrm{Attn}(X_0) = (\widetilde{A} - A_0)V_0 + A_0 \Delta V + (\widetilde{A} - A_0)\Delta V. \tag{50}$$

For every score entry, bilinearity gives:

$$|S_{ij}(X) - S_{ij}(X_0)| \le \epsilon\sqrt{d_{\mathrm{tok}}}\, \|M\|_{\mathrm{op}} \left( \|x_0^{(i)}\|_2 + \|x_0^{(j)}\|_2 \right) + \epsilon^2 d_{\mathrm{tok}}\, \|M\|_{\mathrm{op}}. \tag{51}$$

Define

$$\bar{B}_S(\epsilon) = \max_{i,j} \left[ \sqrt{d_{\mathrm{tok}}}\, \|M\|_{\mathrm{op}} \left( \|x_0^{(i)}\|_2 + \|x_0^{(j)}\|_2 \right) + \epsilon d_{\mathrm{tok}}\, \|M\|_{\mathrm{op}} \right]. \tag{52}$$

Then $\|S(X) - S(X_0)\|_\infty \le \epsilon \bar{B}_S(\epsilon)$.

Using the softmax Jacobian bound $\|\nabla \mathrm{softmax}(z)\|_{\mathrm{op}} \le \frac{1}{2}$, for each attention row $i$ we have

$$\|\widetilde{a}_i - a_i^0\|_1 \le \frac{n}{2} \epsilon \bar{B}_S(\epsilon). \tag{53}$$

Also, $\|\Delta V\|_{2,\infty} \le \epsilon\sqrt{d_{\mathrm{tok}}}\, \|W_V\|_{\mathrm{op}}$, where $\|Z\|_{2,\infty} = \max_i \|Z[i,:]\|_2$. Applying these bounds to the three terms in (50) results in:

$$\|\mathrm{Attn}(X) - \mathrm{Attn}(X_0)\|_{2,\infty} \le \epsilon L_{\mathrm{attn}}(\epsilon), \tag{54}$$

where

$$L_{\mathrm{attn}}(\epsilon) = \frac{n}{2} \bar{B}_S(\epsilon) \|V_0\|_{2,\infty} + \sqrt{d_{\mathrm{tok}}}\, \|W_V\|_{\mathrm{op}} + \frac{n}{2} \epsilon \bar{B}_S(\epsilon) \sqrt{d_{\mathrm{tok}}}\, \|W_V\|_{\mathrm{op}}. \tag{55}$$

Since $\|Z\|_F \le \sqrt{n} \|Z\|_{2,\infty}$, this implies

$$\|\mathrm{Attn}(X) - \mathrm{Attn}(X_0)\|_F \le \sqrt{n} L_{\mathrm{attn}}(\epsilon) \epsilon. \tag{56}$$

**Proposition B.5** (Fixed-pattern attention robustness)**.** *Let $f = h \circ \mathrm{Attn}$, and suppose each logit of $h$ is $L_h$-Lipschitz with respect to the Frobenius norm. If the score-pattern stability condition holds and $\mu(X_0) > 2L_h\sqrt{n} L_{\mathrm{attn}}(\epsilon)\epsilon$, then $f$ is robust on $\mathcal{B}_\epsilon(X_0)$.*

*Proof.* By (56) and the logit-wise $L_h$-Lipschitzness of $h$, for each class $r$,

$$|f_r(X) - f_r(X_0)| \le L_h \sqrt{n} L_{\mathrm{attn}}(\epsilon)\epsilon. \tag{57}$$

Therefore, for any $k \neq y$,

$$\begin{aligned} f_y(X) - f_k(X) &\ge f_y(X_0) - f_k(X_0) - |f_y(X) - f_y(X_0)| - |f_k(X) - f_k(X_0)| \\ &\ge \mu(X_0) - 2L_h\sqrt{n}L_{\mathrm{attn}}(\epsilon)\epsilon > 0. \end{aligned} \tag{58}$$

Thus the predicted class remains $y$ for every $X \in \mathcal{B}_\epsilon(X_0)$. □

### B.7. Dominant-Key Linear Attention Constructor Details

The robustness certificate has three steps, and we use the same notation as in the main text. First, we show that the dominance condition implies that each attention row is close to the value vector associated with its dominant key. Second, we show that this closeness gives a uniform bound on the full attention-output drift. Third, this output-drift bound combines with the nominal classification margin to certify robustness.

**Part 1: Closeness to dominant value.** The dominance condition says that for query row $i$, the unnormalized weight assigned to key $j_i^\star$ is at least $\rho_i$ times the total weight assigned to all other keys. The next lemma makes the consequence precise:

**Lemma B.6** (Closeness to the dominant value)**.** *If row $i$ satisfies the dominance condition in* (7) *throughout $\mathcal{B}_\epsilon(X_0)$, then for every $X \in \mathcal{B}_\epsilon(X_0)$,*

$$\|\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] - V(X)[j_i^\star,:]\|_2 \le \frac{1}{1+\rho_i} \max_{j \neq j_i^\star} \|V(X)[j,:] - V(X)[j_i^\star,:]\|_2 . \tag{59}$$

*Proof.* Let $S_i(X) = \sum_{j \neq j_i^\star} w_{ij}(X)$ and $w_i^\star(X) = w_{ij_i^\star}(X)$. We also have that $w_i^\star(X) \ge \rho_i S_i(X)$ by (7). Therefore:

$$\alpha_{ij_i^\star}(X) = \frac{w_i^\star(X)}{w_i^\star(X) + S_i(X)} \ge \frac{\rho_i}{1+\rho_i}, \qquad 1 - \alpha_{ij_i^\star}(X) \le \frac{1}{1+\rho_i}. \tag{60}$$

Using the definition of linear attention,

$$\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] - V(X)[j_i^\star,:] = \sum_{j \neq j_i^\star} \alpha_{ij}(X) \left(V(X)[j,:] - V(X)[j_i^\star,:]\right). \tag{61}$$

Applying the triangle inequality and (60) gives

$$\begin{aligned} \|\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] - V(X)[j_i^\star,:]\|_2 &\le \sum_{j \neq j_i^\star} \alpha_{ij}(X) \|V(X)[j,:] - V(X)[j_i^\star,:]\|_2 \\ &\le \left(1 - \alpha_{ij_i^\star}(X)\right) \max_{j \neq j_i^\star} \|V(X)[j,:] - V(X)[j_i^\star,:]\|_2 \\ &\le \frac{1}{1+\rho_i} \max_{j \neq j_i^\star} \|V(X)[j,:] - V(X)[j_i^\star,:]\|_2 . \end{aligned} \tag{62}$$

□

**Part 2: Attention-output perturbation.** The previous lemma controls each attention row relative to its dominant value vector. We now compare the full attention outputs at $X$ and $X_0$ by inserting the dominant value vector at both inputs. For compactness, let $V_0 = V(X_0)$ and set $L_V = \sqrt{d_{\mathrm{tok}}}\|W_V\|_{\mathrm{op}}$. If $X = X_0 + \Delta X$ and $\|\Delta X\|_\infty \le \epsilon$, then each token perturbation satisfies $\|\delta^{(i)}\|_2 \le \epsilon\sqrt{d_{\mathrm{tok}}}$, so

$$\|V(X)[j,:] - V_0[j,:]\|_2 = \|\delta^{(j)} W_V\|_2 \le \epsilon L_V. \tag{63}$$

**Proposition B.7** (Dominant-key attention-output perturbation)**.** *Suppose every row satisfies the dominance condition in* (7) *throughout* $\mathcal{B}_\epsilon(X_0)$*, and let* $\rho = \min_i \rho_i$*. Then for every* $X \in \mathcal{B}_\epsilon(X_0)$*,*

$$\|\mathrm{Attn}_{\mathrm{lin}}(X) - \mathrm{Attn}_{\mathrm{lin}}(X_0)\|_{2,\infty} \le \frac{2}{1+\rho} D_V(X_0) + \left(1 + \frac{2}{1+\rho}\right) \epsilon L_V. \tag{64}$$

*Proof.* Fix a row $i$. Insert the value vector associated with the dominant key at both $X$ and $X_0$:

$$\begin{aligned} &\|\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] - \mathrm{Attn}_{\mathrm{lin}}(X_0)[i,:]\|_2 \\ &\quad \le \|\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] - V(X)[j_i^\star,:]\|_2 + \|V(X)[j_i^\star,:] - V_0[j_i^\star,:]\|_2 \\ &\qquad + \|V_0[j_i^\star,:] - \mathrm{Attn}_{\mathrm{lin}}(X_0)[i,:]\|_2 . \end{aligned} \tag{65}$$

By Lemma B.6 applied at $X$,

$$\|\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] - V(X)[j_i^\star,:]\|_2 \le \frac{1}{1+\rho} \max_{j \ne j_i^\star} \|V(X)[j,:] - V(X)[j_i^\star,:]\|_2. \tag{66}$$

For every $j \ne j_i^\star$, adding and subtracting nominal values gives

$$\begin{aligned} \|V(X)[j,:] - V(X)[j_i^\star,:]\|_2 &\le \|V_0[j,:] - V_0[j_i^\star,:]\|_2 \\ &\quad + \|V(X)[j,:] - V_0[j,:]\|_2 + \|V(X)[j_i^\star,:] - V_0[j_i^\star,:]\|_2 \\ &\le D_V(X_0) + 2\epsilon L_V. \end{aligned} \tag{67}$$

Thus the first term in (65) is bounded by $\frac{D_V(X_0)+2\epsilon L_V}{(1+\rho)}$. The middle term is bounded by $\epsilon L_V$ using (63). The last term is bounded by $\frac{D_V(X_0)}{(1+\rho)}$ by Lemma B.6 applied at $X_0$. Summing the three bounds gives

$$\|\mathrm{Attn}_{\mathrm{lin}}(X)[i,:] - \mathrm{Attn}_{\mathrm{lin}}(X_0)[i,:]\|_2 \le \frac{2}{1+\rho} D_V(X_0) + \left(1 + \frac{2}{1+\rho}\right) \epsilon L_V. \tag{68}$$

Taking the maximum over rows gives the $\|\cdot\|_{2,\infty}$ bound. □

**Part 3: Robustness from output drift.** The previous proposition bounds the drift of the linear-attention output. The final step is the same argument used in the fixed-ordering softmax constructor proof, i.e. if the downstream head cannot move the logits enough to close the nominal margin, then the predicted class is fixed.

**Proposition B.8** (Dominant-key linear attention robustness certificate)**.** *Let* $f = h \circ \mathrm{Attn}_{\mathrm{lin}}$*, and suppose each logit of* $h$ *is* $L_h$*-Lipschitz with respect to the Frobenius norm. Define*

$$\Delta_{\mathrm{lin}} = \frac{2}{1+\rho} D_V(X_0) + \left(1 + \frac{2}{1+\rho}\right) \epsilon L_V. \tag{69}$$

*If* $\mu(X_0) > 2L_h\sqrt{n}\,\Delta_{\mathrm{lin}}$*, then* $f$ *is robust on* $\mathcal{B}_\epsilon(X_0)$*.*

*Proof.* By Proposition B.7, $\|\mathrm{Attn}_{\mathrm{lin}}(X) - \mathrm{Attn}_{\mathrm{lin}}(X_0)\|_{2,\infty} \le \Delta_{\mathrm{lin}}$. Therefore,

$$\|\mathrm{Attn}_{\mathrm{lin}}(X) - \mathrm{Attn}_{\mathrm{lin}}(X_0)\|_F \le \sqrt{n}\,\Delta_{\mathrm{lin}}. \tag{70}$$

Since each logit of $h$ is $L_h$-Lipschitz, for every class $r$,

$$|f_r(X) - f_r(X_0)| \le L_h\sqrt{n}\,\Delta_{\mathrm{lin}}. \tag{71}$$

Therefore, for any $k \ne y$,

$$\begin{aligned} f_y(X) - f_k(X) &\ge f_y(X_0) - f_k(X_0) - |f_y(X) - f_y(X_0)| - |f_k(X) - f_k(X_0)| \\ &\ge \mu(X_0) - 2L_h\sqrt{n}\,\Delta_{\mathrm{lin}} > 0. \end{aligned} \tag{72}$$

Thus the predicted class remains $y$ for every $X \in \mathcal{B}_\epsilon(X_0)$. □

### B.8. Polynomial Network Constructor

**Construction and robustness certificate** We describe the polynomial network constructor in more detail. The construction applies to binary classifiers whose margin is a polynomial function of the input. Let $f : \mathbb{R}^n \to \mathbb{R}^2$ be a binary classifier, and write $\mu(x) = f_1(x) - f_2(x)$ for its margin polynomial. The decision boundary is the real algebraic hypersurface $D = \{x \in \mathbb{R}^n : \mu(x) = 0\}$. A point $x_0$ is robust at radius $\epsilon$ with respect to a norm $\|\cdot\|$ if $\mu$ has constant sign on $B_\epsilon(x_0) = \{x : \|x - x_0\| \le \epsilon\}$. Equivalently, $B_\epsilon(x_0) \cap D = \emptyset$.

The constructor proceeds in three steps. First, it samples a smooth point $p \in D$, so that $\nabla\mu(p) \neq 0$. Second, it chooses a norm-adapted normal direction $u(p)$ pointing to one side of the decision boundary. More precisely, $u(p)$ is normalized in the chosen norm and satisfies $\nabla\mu(p) \cdot u(p) > 0$. Third, for a target verification radius $\epsilon > 0$ and buffer $\delta > 0$, it sets $x_0 = p + (\epsilon + \delta)u(p)$. The sampled point $p$ is then a known boundary point at distance $\epsilon + \delta$ from $x_0$, and hence lies just outside the perturbation set $B_\epsilon(x_0)$.

The preceding step is only a local construction. It guarantees that one known point of the decision boundary lies outside the verification set, but it does not rule out the possibility that another branch or component of $D$ enters $B_\epsilon(x_0)$. The robustness label is therefore obtained from the following global separation certificate.

**Proposition B.9.** *If $B_\epsilon(x_0) \cap D = \emptyset$, then $x_0$ is robust at radius $\epsilon$.*

*Proof.* Since $B_\epsilon(x_0)$ is connected and $\mu$ is continuous, the image $\mu(B_\epsilon(x_0))$ is connected in $\mathbb{R}$. If $B_\epsilon(x_0) \cap D = \emptyset$, then $\mu(x) \neq 0$ for every $x \in B_\epsilon(x_0)$. Hence $\mu$ cannot change sign on $B_\epsilon(x_0)$. Therefore the classifier assigns the same label to every point in $B_\epsilon(x_0)$, so $x_0$ is robust at radius $\epsilon$. □

Thus the constructor separates the task of generating near-boundary candidates from the task of certifying their ground-truth labels. The perturbation step places $x_0$ close to the algebraic decision boundary with a prescribed buffer $\delta$, while the global separation check certifies that the full perturbation set remains on one side of $D$. Decreasing $\delta$ produces instances that are robust but closer to failure, and therefore harder for verifiers to certify.

**Boundary sampling and shallow polynomial networks** We now describe how boundary points are sampled and how the construction is instantiated for the shallow polynomial networks used in the experiments. Since $\mu$ is a polynomial, points on $D = \{\mu = 0\}$ can be obtained by slicing $D$ with affine linear spaces. In our setting $D$ is a hypersurface, so a generic affine line is enough. We choose a random point $a \in \mathbb{R}^n$ and a random direction $v \in \mathbb{R}^n$, and restrict $\mu$ to the line $\ell(t) = a + tv$. This gives the univariate polynomial $\mu_\ell(t) = \mu(a + tv)$. Each real root of $\mu_\ell$ gives a boundary point $p = \ell(t) \in D$. For a generic line, the intersection is transverse away from the singular locus of $D$, so sampled points are smooth unless they also satisfy $\nabla\mu(p) = 0$. We discard such points.

In the experiments, we use one-hidden-layer polynomial networks with architecture $(n, h, 2)$,

$$f(x) = W_2(W_1 x + b_1)^d + b_2,$$

where the power is applied entrywise, $W_1 \in \mathbb{R}^{h\times n}$, $b_1 \in \mathbb{R}^h$, $W_2 \in \mathbb{R}^{2\times h}$, and $b_2 \in \mathbb{R}^2$. Writing $W_{2,1}$ and $W_{2,2}$ for the two rows of $W_2$, the margin polynomial is

$$\mu(x) = (W_{2,1} - W_{2,2}) \cdot (W_1 x + b_1)^d + (b_{2,1} - b_{2,2}).$$

Thus the decision boundary $D = \{\mu = 0\}$ is a real algebraic hypersurface of degree at most $d$.

For the $\ell_\infty$ perturbation sets used in our experiments, we take the norm-adapted normal direction to be $u(p) = \mathrm{sign}(\nabla\mu(p))$. This direction has $\|u(p)\|_\infty = 1$ and satisfies $\nabla\mu(p) \cdot u(p) = \|\nabla\mu(p)\|_1 > 0$ at every smooth boundary point. We fix a push distance $r > 0$, set $x_0 = p + ru(p)$, and verify at radius $\epsilon = r - \delta$. Equivalently, $r = \epsilon + \delta$. The sampled boundary point $p$ is then at $\ell_\infty$ distance $r$ from $x_0$, while the verification box $B_\epsilon(x_0)$ has buffer $\delta$ from this known boundary point.

**Numerical separation check and verifier protocol** The global condition $B_\epsilon(x_0) \cap D = \emptyset$ is the mathematical certificate for robustness. In the present experiments, we implement this check numerically before passing instances to the verifier. For each candidate center $x_0$, we search for an intersection between the verification box and the decision boundary by solving $\min_{z\in B_\epsilon(x_0)} \mu(z)^2$. The minimum is zero if and only if the box intersects $D = \{\mu = 0\}$. We solve this box-constrained problem using L-BFGS-B from 50 random initializations in $B_\epsilon(x_0)$, with a maximum of 500 iterations per restart. If the

smallest value found is below $10^{-6}$, we treat the candidate as numerically intersecting the decision boundary and discard it. Otherwise, we keep the candidate as passing the numerical separation check.

For each retained instance, we export the network to ONNX and generate a VNNLIB specification for the $\ell_\infty$ box $B_\epsilon(x_0)$. The input constraints are $x_{0,i} - \epsilon \leq x_i \leq x_{0,i} + \epsilon$ for $i = 1, \ldots, n$. The output constraint requires the predicted label at $x_0$ to remain unchanged throughout the box. Equivalently, if $\mu(x_0) > 0$, the verifier is asked to prove $\mu(x) > 0$ for all $x \in B_\epsilon(x_0)$, and if $\mu(x_0) < 0$, it is asked to prove $\mu(x) < 0$ for all $x \in B_\epsilon(x_0)$.

We record verifier outcomes as *robust*, *non-robust*, *timeout*, or *unknown*. A *robust* result means that the verifier certified the output constraint over the full box. A *non-robust* result means that it found a violation of the specification. A *timeout* or *unknown* result means that no certificate or counterexample was found within the time limit.

**Ablation Study Results** Table 2 summarizes an ablation study for the shallow polynomial network constructor using $\alpha, \beta$-CROWN. We vary one parameter at a time around a baseline configuration, keeping the push distance fixed at $r = 0.02$ and setting the verification radius to $\epsilon = r - \delta$. Each row reports the number of retained instances submitted to $\alpha, \beta$-CROWN after the numerical separation check in Section B.8. A timeout is recorded when $\alpha, \beta$-CROWN does not return either a certificate or a counterexample within the time limit in the *Limit* column.

The clearest transitions in this ablation occur when increasing the polynomial degree $d$ or decreasing the buffer $\delta$. This matches the geometry of the construction: $d$ controls the degree of the margin polynomial, and hence the algebraic complexity of the decision boundary. This is also consistent with the ED degree formulas in (Alexandr et al., 2026), which count complex critical points of the closest-boundary equations and involve powers of $d - 1$. The buffer $\delta$ controls a complementary geometric difficulty: since $\epsilon = r - \delta$, smaller $\delta$ places the verification box closer to the sampled boundary point, leaving less separation for $\alpha, \beta$-CROWN to certify. In contrast, varying $n$ or $h$ changes the ambient dimension or the number of hidden units, but has a milder effect in this sweep.

*Table 2.* Verification outcomes for retained near-boundary instances generated from shallow polynomial networks and evaluated with $\alpha, \beta$-CROWN. The push distance is fixed at $r = 0.02$, and the verification radius is $\epsilon = r - \delta$. Mean time includes timeout runs; mean time without timeouts averages only certified runs.

| $n$ | $d$ | $h$ | $\delta$ | Limit | Points | Robust | Timeouts | Mean time | Mean time (no T/O) |
|---|---|---|---|---|---|---|---|---|---|
| 50 | 10 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 9.10s | 9.10s |
| 100 | 10 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 7.88s | 7.88s |
| 200 | 10 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 7.86s | 7.86s |
| 300 | 10 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.42s | 8.42s |
| 100 | 2 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.32s | 8.32s |
| 100 | 4 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.63s | 8.63s |
| 100 | 6 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.36s | 8.36s |
| 100 | 8 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.22s | 8.22s |
| 100 | 10 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 7.88s | 7.88s |
| 100 | 12 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.23s | 8.23s |
| 100 | 14 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.22s | 8.22s |
| 100 | 16 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 8.00s | 8.00s |
| 100 | 18 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 4 | 1 | 68.43s | 8.53s |
| 100 | 20 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 3 | 2 | 129.13s | 9.50s |
| 100 | 22 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 2 | 3 | 187.51s | 8.85s |
| 100 | 10 | 50 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 6.28s | 6.28s |
| 100 | 10 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 7.88s | 7.88s |
| 100 | 10 | 200 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 6.22s | 6.22s |
| 100 | 10 | 300 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 6.12s | 6.12s |
| 100 | 10 | 500 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 6.77s | 6.77s |
| 100 | 10 | 100 | $10^{-3}$ | 300s | 5 | 4 | 1 | 66.72s | 6.58s |
| 100 | 10 | 100 | $5 \cdot 10^{-3}$ | 300s | 5 | 5 | 0 | 7.88s | 7.88s |
| 100 | 10 | 100 | $10^{-4}$ | 300s | 5 | 1 | 4 | 248.96s | 7.94s |

Table 3 reports a complementary verifier-comparison experiment on selected polynomial-network instances from the VeriStress study. Unlike Table 2, this table is not an ablation study for $\alpha, \beta$-CROWN over the full parameter sweep. Instead, each row records the outcome of running several verifiers on a retained near-boundary instance generated by the same constructor. The purpose is to show that the construction produces instances that can stress multiple verification back ends. Across these selected instances, $\alpha, \beta$-CROWN certifies many cases but begins to time out as $d$ increases or $\delta$ decreases, while NeuralSAT and PyRAT time out on more of the listed configurations.

*Table 3.* Verifier outcomes for selected retained near-boundary instances used in the VeriStress study described in Section 4. The push distance is fixed at $r = 0.02$, the verification radius is $\epsilon = r - \delta$, and the timeout is 360 seconds. Entries report certification time when the verifier certifies robustness and "Timeout" otherwise. Marabou and nnenum returned errors on these instances and are omitted.

| $n$ | $d$ | $h$ | $\delta$ | $\alpha, \beta$-CROWN outcome | NeuralSAT outcome | PyRAT outcome |
|---|---|---|---|---|---|---|
| 50 | 10 | 100 | $5 \cdot 10^{-3}$ | 9.81s | 39.08s | Timeout |
| 100 | 2 | 100 | $5 \cdot 10^{-3}$ | 8.33s | Timeout | Timeout |
| 100 | 6 | 100 | $5 \cdot 10^{-3}$ | 7.97s | 34.27s | Timeout |
| 100 | 10 | 50 | $5 \cdot 10^{-3}$ | 7.79s | 37.74s | Timeout |
| 100 | 10 | 500 | $5 \cdot 10^{-3}$ | 7.99s | 43.60s | Timeout |
| 100 | 12 | 100 | $5 \cdot 10^{-3}$ | 7.61s | 37.17s | Timeout |
| 100 | 14 | 100 | $5 \cdot 10^{-3}$ | 7.95s | 45.65s | Timeout |
| 100 | 16 | 100 | $5 \cdot 10^{-3}$ | 8.09s | 38.13s | Timeout |
| 100 | 17 | 100 | $5 \cdot 10^{-3}$ | Timeout | Timeout | Timeout |
| 100 | 18 | 100 | $5 \cdot 10^{-3}$ | 9.33s | 41.75s | Timeout |
| 100 | 18 | 100 | $5 \cdot 10^{-3}$ | 9.89s | 42.29s | Timeout |
| 100 | 19 | 100 | $5 \cdot 10^{-3}$ | Timeout | Timeout | Timeout |
| 100 | 19 | 100 | $5 \cdot 10^{-3}$ | Timeout | Timeout | Timeout |
| 100 | 20 | 100 | $5 \cdot 10^{-3}$ | 30.54s | 41.20s | Timeout |
| 100 | 21 | 100 | $5 \cdot 10^{-3}$ | Timeout | Timeout | Timeout |
| 100 | 22 | 100 | $5 \cdot 10^{-3}$ | 34.43s | 44.97s | Timeout |
| 100 | 10 | 100 | $2 \cdot 10^{-3}$ | 34.73s | Timeout | Timeout |
| 100 | 10 | 100 | $1 \cdot 10^{-3}$ | 32.49s | Timeout | Timeout |
| 100 | 10 | 100 | $5 \cdot 10^{-4}$ | 33.92s | Timeout | Timeout |
| 100 | 10 | 100 | $2 \cdot 10^{-4}$ | 34.88s | Timeout | Timeout |
| 100 | 10 | 100 | $1 \cdot 10^{-4}$ | Timeout | Timeout | Timeout |
| 100 | 18 | 100 | $1 \cdot 10^{-3}$ | Timeout | Timeout | Timeout |

## C. Proof of Proposition 3.2

Recall that $L_c := \sup_{x \in \mathcal{B}_\epsilon(x_0)} \sup_{\xi \in \partial_c \mu(x)} \|\xi\|_{p,*}$, where $\|\cdot\|_{p,*}$ is the dual norm of $\|\cdot\|_p$. In the smooth setting of Proposition 3.2, this reduces to $L_c = \sup_{x \in \mathcal{B}_\epsilon(x_0)} \|\nabla \mu(x)\|_{p,*}$. We assume the gradient is $\beta$-Lipschitz with respect to the same primal-dual norm pair, meaning $\|\nabla \mu(x) - \nabla \mu(y)\|_{p,*} \le \beta \|x - y\|_p$ for all $x, y \in \mathcal{B}_\epsilon(x_0)$.

In the following proof, we cover the perturbation ball by smaller $\ell_p$ balls of radius $r$. On each small ball, the first-order Taylor approximation to $\mu$ is accurate up to order $\frac{\beta r^2}{2}$. Choosing $r = \sqrt{2\tau\epsilon L_c/\beta}$ makes this local approximation error at most $\tau\epsilon L_c$, and the number of required balls gives the stated affine-cover bound.

*Proof.* If $L_c = 0$, then $\nabla \mu(x) = 0$ for all $x \in \mathcal{B}_\epsilon(x_0)$, so $\mu$ is constant on the convex set $\mathcal{B}_\epsilon(x_0)$. Hence one affine function represents $\mu$ exactly, giving $N_{\text{aff}}(\delta) = 1$ and $A^\star_\tau = 0$. The result is therefore immediate.

Next, assume $L_c > 0$ and let $\delta := \tau\epsilon L_c$. If $\beta = 0$, then $\nabla \mu$ is constant on $\mathcal{B}_\epsilon(x_0)$, so $\mu$ is affine on $\mathcal{B}_\epsilon(x_0)$. Again, $N_{\text{aff}}(\delta) = 1$ and $A^\star_\tau = 0$. So, assume $\beta > 0$ and set

$$r := \sqrt{\frac{2\delta}{\beta}} = \sqrt{\frac{2\tau\epsilon L_c}{\beta}}. \tag{73}$$

Let $\mathcal{B}_p(z,r) := \{x : \|x-z\|_p \le r\}$ be a closed $\ell_p$ ball. By the standard covering-number bound for an $\ell_p$ ball, there exist points $z_1,\dots,z_M \in \mathcal{B}_\epsilon(x_0)$ such that:

$$\mathcal{B}_\epsilon(x_0) \subseteq \bigcup_{i=1}^{M} \left(\mathcal{B}_p(z_i,r) \cap \mathcal{B}_\epsilon(x_0)\right), \qquad M \le \left(1+\frac{2\epsilon}{r}\right)^d. \tag{74}$$

Define $U_i := \mathcal{B}_p(z_i,r) \cap \mathcal{B}_\epsilon(x_0)$. Each $U_i$ is convex because it is the intersection of two convex sets. For each cell $U_i$, define the first-order affine approximation

$$\ell_i(x) := \mu(z_i) + \nabla\mu(z_i)^\top (x-z_i). \tag{75}$$

Fix any $x \in U_i$. By the fundamental theorem of calculus along the segment from $z_i$ to $x$, we have that:

$$\mu(x) - \mu(z_i) = \int_0^1 \nabla\mu\big(z_i + t(x-z_i)\big)^\top (x-z_i)\,dt, \tag{76}$$

$$\mu(x) - \ell_i(x) = \int_0^1 \left[\nabla\mu\big(z_i + t(x-z_i)\big) - \nabla\mu(z_i)\right]^\top (x-z_i)\,dt. \tag{77}$$

Using Holder duality and the $\beta$-Lipschitz property of the gradient:

$$|\mu(x) - \ell_i(x)| \le \int_0^1 \left\|\nabla\mu\big(z_i + t(x-z_i)\big) - \nabla\mu(z_i)\right\|_{p,*} \|x-z_i\|_p\,dt \tag{78}$$

$$\le \int_0^1 \beta t \|x-z_i\|_p^2\,dt \tag{79}$$

$$= \frac{\beta}{2}\|x-z_i\|_p^2. \tag{80}$$

Since $x \in U_i \subseteq \mathcal{B}_p(z_i,r)$, we have $\|x-z_i\|_p \le r$, and hence $|\mu(x)-\ell_i(x)| \le \frac{\beta}{2}r^2 = \delta$. Thus each $U_i$ admits an affine approximation with error at most $\delta$, so:

$$N_{\text{aff}}(\delta) \le M \le \left(1+\frac{2\epsilon}{r}\right)^d \tag{81}$$

Substituting the definition of $r$ gives:

$$\frac{2\epsilon}{r} = 2\epsilon\sqrt{\frac{\beta}{2\tau\epsilon L_c}} = \sqrt{\frac{2\beta\epsilon}{\tau L_c}} = \sqrt{\frac{2\widetilde{\beta}}{\tau}}, \qquad \widetilde{\beta} := \frac{\beta\epsilon}{L_c}. \tag{82}$$

Therefore:

$$N_{\text{aff}}(\tau\epsilon L_c) \le \left(1+\sqrt{\frac{2\widetilde{\beta}}{\tau}}\right)^d \tag{83}$$

$$\implies A_\tau^\star = \log N_{\text{aff}}(\tau\epsilon L_c) \le d\log\left(1+\sqrt{\frac{2\widetilde{\beta}}{\tau}}\right) \tag{84}$$

□

The proposition shows that in smooth networks, affine-cover complexity is controlled by the normalized gradient-variation scale $\widetilde{\beta} = \frac{\beta\epsilon}{L_c}$. In principle, one could estimate $\widetilde{\beta}$ directly and use the bound as a profile component. In practice, however, estimating a global gradient-Lipschitz constant over $\mathcal{B}_\epsilon(x_0)$ is often conservative, so Section 3 instead uses the empirical statistic $A_\tau$, which samples and counts distinct local linearization classes. For piecewise-linear networks where a global smoothness constant may be infinite, $A_\tau$ is interpreted as an empirical count of distinct local affine behaviors encountered inside the perturbation set.

## D. Verifier Taxonomy by Certificate Type

Our characterization of instance difficulty is guided by the wide array of verification algorithm implementations, as the profile must apply to all verifiers. Verification algorithms differ not only in implementation but in the fundamental mathematical object they construct as evidence that $\min_{x \in \mathcal{B}_\epsilon(x_0)} \mu(x) > 0$. We organize the verifier landscape by certificate type rather than by implementation details such as solver backend or bounding subroutine. We identify four certificate types:

Style D (Dual Relaxation). The verifier produces a dual certificate, i.e., a feasible solution to the dual of a convex relaxation of $\min_{x \in \mathcal{B}} \mu(x)$. Verification terminates in a single pass and no branching or combinatorial search is performed. The quality of the certificate depends entirely on the tightness of the relaxation. Examples: IBP (Gowal et al., 2018), CROWN (Zhang et al., 2018), $\alpha$-CROWN (Xu et al., 2020b; 2021) without branching.

Style P (Partition Search). The verifier produces a finite cover $\{U_i\}$ of $\mathcal{B}_\epsilon(x_0)$ together with a bound on $\mu$ over each $U_i$, jointly witnessing $\mu > 0$ on $\mathcal{B}_\epsilon(x_0)$. The cover is constructed by recursively splitting the domain and bounding $\mu$ on each piece via relaxation, pruning subproblems whose relaxation already certifies the property. The cost depends on the number of subproblems solved via pruning. Examples: $\alpha, \beta$-CROWN, Marabou's divide-and-conquer mode (Katz et al., 2019).

Style S (SAT Refutation). The verifier produces a refutation certificate, i.e. a proof that no assignment of activation states is jointly consistent with the constraints implied by $\mu(x) \leq 0$. The search is often conducted by a DPLL(T)-style SAT engine (Ganzinger et al., 2004) over Boolean activation variables with a solver checking consistency of each partial assignment and learned conflict clauses pruning future candidates. The cost depends on how many partial assignments must be explored before the search space is exhausted. Examples: NeuralSAT (Duong et al., 2024), Marabou's DPLL(T) engine.

Style R (Reachability). The verifier produces a reachability certificate, i.e. an overapproximation of the output range $\{f(x) : x \in \mathcal{B}_\epsilon(x_0)\}$, obtained by propagating an abstract set representation of $\mathcal{B}_\epsilon(x_0)$ forward through $f$ layer by layer. Unlike other verifier styles, the certificate is typically constructed in the forward direction without explicit duality or domain decomposition, and its cost depends on how the abstract representation grows as it passes through the network $f$. Examples: nnenum (zonotopes with refinement) (Bak, 2021), NNV (star sets) (Tran et al., 2020).

Note that several verifiers combine multiple certificate types. For instance, $\alpha, \beta$-CROWN uses a dual relaxation certificate at each node of a partition search (styles D and P). Similarly, nnenum uses abstract propagation with domain splitting (styles P and R).

## E. Bug Discovery in Open-Source Verifier

During development of VeriStress-GT, we reported a bug in the open-source verifier [Verifier][1]. The issue arose in [Verifier]'s MIP presolve routine for specifications with disjunctive output constraints. In the verifier convention used here, SAT means that the adversarial constraints are feasible, i.e., a counterexample exists, while UNSAT means that no counterexample exists for the given specification.

The relevant class of specifications can be written as a disjunction$\Phi = \bigvee_{j=1}^{m} \Phi_j$. Correct handling of such a specification requires OR semantics, i.e. the verifier should return SAT if any disjunct $\Phi_j$ is feasible, should return UNSAT only if all disjuncts are infeasible, and should return UNKNOWN if no disjunct is shown feasible but at least one disjunct cannot be resolved. However, the observed behavior in the MIP presolve routine appeared to return the status of the final disjunct checked. As a result, if an earlier disjunct was feasible but the final disjunct was infeasible, the routine could incorrectly return UNSAT on a satisfiable adversarial specification.

This failure mode occurred only under a specific combination of conditions. The specification contained multiple disjuncts, at least one non-final disjunct was feasible, the final disjunct in iteration order was infeasible, and MIP presolve was enabled. The reported fix was to aggregate disjunct statuses explicitly. In other words, return SAT immediately upon finding a feasible disjunct and otherwise return UNKNOWN if any disjunct was unresolved, and return UNSAT only after all disjuncts have been proved infeasible.

This discovery illustrates one of the motivations for VeriStress-GT. Benchmarks whose labels are inferred from verifier consensus or from the absence of counterexamples can obscure implementation-level soundness failures. By contrast, VeriStress-GT instances have verifier-independent ground-truth robustness labels, allowing incorrect verifier outcomes to be

[1] Identity of codebase has been redacted to preserve anonymity in the double blind review process.

detected directly rather than treated as ambiguous disagreements among tools. We appreciate the responsiveness of the developers of the [Verifier] package in quickly reviewing and approving our proposed fix for this issue.

## F. Difficulty Profile Study Details

### F.1. Design Principles

Below we state the three major design principles used to guide the process of Difficulty Profile component selection:

(D1, Instance-level computability): Each component can be computed without access to the internals of a particular verifier. Thus the difficulty profile component characterizes the problem instance rather than implementation and efficiency details of a verifier.

(D2, Certificate Coverage): The components are selected to cover varying bottlenecks in the certification problem. These correspond to fundamentally different reasons a verifier may require branching or refinement and help ensure the Difficulty Profile is applicable to all verifier types (See Appendix D for a description of different verifier styles).

(D3, Non-Redundancy and Interpretability): No two components intend to capture the same phenomenon. Otherwise, one may saturate the pool of possible components with many statistics recorded during experimentation, reducing interpretability and leading to a bloated set of components.

### F.2. Component Search

Lastly, we note that many difficulty profile components were tested before decided on the canonical set enumerated in Section 3. Table 4 below details many of the component ideas that were tested but ultimately not included, typically due to limited correlations with runtime or timeout predictions.

## G. Numerical Study Details

Table 5 displays the aggregated VeriStress-GT stress-test study results, while Table 6 details what hyperparameter ranges were used in each construction for the instantiation of the VeriStress-GT framework. To see the specific configuration for all 225 instances for the stress-test, see "veristress.yaml" and "polynomial_stress_22.yaml" in the configs folder in our public code repository: https://github.com/ctbanonymous12345/VeriStressGT.git

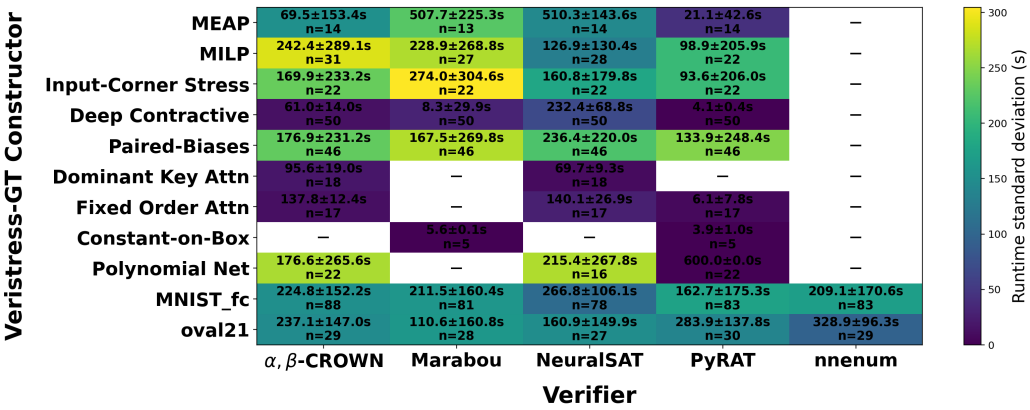



*Figure 4.* Runtime variability across instances for each constructor/benchmark and verifier. Each cell reports empirical mean ± standard deviation of runtime in seconds, computed over verified instances and timeouts using the same timing convention as Figure 3. Verified instances use measured wall-clock time, while timeouts are assigned the benchmark timeout value. Error, unsupported, and unknown outcomes are excluded from the runtime-spread estimate. A cell is left blank if the standard deviation for the runtime was not applicable or 0.

Additionally, we also include trajectory plots showing Difficulty Profile component estimates and corresponding verifier outcomes for each instance in the external MNIST_fc and oval21 benchmarks in Figures 5 and 6. To access the full csv files containing the difficulty profile estimates and corresponding outcomes (including those for the 225 tested VeriStress-GT framework) please see the Results folder in our public code repository: https://github.com/

*Table 4.* Difficulty Profile component ideas tested and corresponding explanations as to why they did not appear in the canonical profile defined in Section 3.

| Component Idea | Intended Signal | Reason Discarded |
|---|---|---|
| Empirical mean margin | Average robustness slack over sampled perturbations. | Discarded in favor of $\widehat{M}_{\min}$, which better matches verification failure: a single low-margin region can dominate robustness difficulty even when the average margin is large. |
| IBP absolute gap | Raw looseness of interval propagation. | Folded into $G_{\mathrm{IBP}}$. The relative version was more comparable across constructions because raw gaps are strongly affected by logit scale. |
| IBP lower margin | Whether plain IBP certifies the instance. | Used as a diagnostic, but discarded as a profile component because $G_{\mathrm{IBP}}$ captures relaxation looseness more directly and is measured via a continuous scale |
| Unstable count | Number of ambiguous nonlinear units. | Folded into $U$. The fraction normalizes for network size |
| Total nonlinear units | Raw nonlinear network size. | Discarded because it is mostly a size statistic. $U$ and $A_\tau$ better capture whether nonlinearities are actually difficult inside the perturbation set. |
| Raw local pattern count | Number of observed local activation patterns. | Folded into $A_\tau$. The tolerance-based/log-scaled version was more stable |
| Gradient dimension maximum | Worst sampled gradient-dimensionality estimate. | Discarded in favor of $d_{\mathrm{eff}}$ mean/aggregate, since the maximum was noisier and more sensitive to outlier samples. |
| Gradient norm | Local first-order sensitivity scale. | Discarded due to scale dependence and limited standalone intuition. Margin scale and relaxation looseness were better captured by $\widehat{M}_{\min}$ and $G_{\mathrm{IBP}}$. |
| Gradient sensitivity | Variation of gradients across the perturbation set. | Discarded because it was noisier than the canonical components. Its useful signal was mostly absorbed by $A_\tau$ and $d_{\mathrm{eff}}$. |
| Gradient concentration | Whether sensitivity lies in few coordinates. | Folded into $d_{\mathrm{eff}}$, which gives a more direct and normalized measure of effective sensitivity dimension. |
| Relative linearization error | Deviation from first-order local behavior. | Discarded due to weaker and less consistent correlation. $A_\tau$ gave a cleaner proxy for local piecewise-linear complexity. |
| Absolute linearization error | Raw first-order approximation error. | Discarded because it is scale-sensitive. The relative version was preferable, but ultimately $A_\tau$ had clearer intuition. |
| First-order margin ratio | Margin compared to local linear drift. | Discarded because it mixed multiple effects. $\widehat{M}_{\min}$ and $d_{\mathrm{eff}}$ separated margin tightness from directional complexity more cleanly. |
| Lipschitz-margin ratio | Worst-case sensitivity relative to margin. | Discarded because global/local Lipschitz estimates were loose and scale-dependent. $\widehat{M}_{\min}$ and $G_{\mathrm{IBP}}$ were more directly tied to verification outcomes. |
| Estimated local Lipschitz | Sampled local sensitivity bound. | Discarded due to estimator noise and limited added value beyond gradient- and margin-based canonical components. |
| Curvature proxy | Smooth nonlinear bending of the margin. | Discarded because estimates were noisy and less natural for ReLU networks. $A_\tau$ was a better architecture-agnostic nonlinearity proxy. |
| Parameter count | Raw model size. | Does not distinguish easy large networks from hard small ones. |
| Layer count | Raw network depth. | Discarded as too architecture-dependent. Depth can create difficulty, but only through effects better captured by $U$, $A_\tau$, $G_{\mathrm{IBP}}$, or $d_{\mathrm{eff}}$. |
| Instability-region interaction | Many unstable units across many local regions. | Discarded because it was less interpretable than reporting $U$ and $A_\tau$ separately, with limited evidence of consistent added correlation. |
| Looseness-instability interaction | Loose relaxations plus many ambiguous units. | Discarded because it was mostly redundant with the two canonical components and harder to explain cleanly. |
| Looseness-region interaction | Loose relaxations across many regions. | Discarded because it added complexity without enough interpretability gain over $G_{\mathrm{IBP}}$ and $A_\tau$. |
| Margin-looseness interaction | Small margin plus loose bounds. | Discarded because it was highly sensitive when $\widehat{M}_{\min}$ was small. Reporting $\widehat{M}_{\min}$ and $G_{\mathrm{IBP}}$ separately was more stable. |

`ctbanonymous12345/VeriStressGT.git`

Finally, we also report the distribution of runtimes in Figure 4, corresponding to the reporting of averages in Figure 3.

| Verifier | UNSAT | SAT | Timeout/Unknown | Error/Unsupported | Total |
|---|---|---|---|---|---|
| $\alpha, \beta$-CROWN | 193 | 0 | 32 | 0 | 225 |
| Marabou | 122 | 5 | 41 | 57 | 225 |
| NeuralSAT | 188 | 2 | 23 | 12 | 225 |
| PyRAT | 160 | 0 | 25 | 40 | 225 |
| nnenum | 102 | 1 | 45 | 77 | 225 |

*Table 5.* Verification outcomes from Section 4, aggregated across constructors.

*Table 6.* Constructor hyperparameter ranges for the VeriStress-GT instantiation described in Section 4. The fixed column lists compact constructor-level constants, while the varied column reports the hyperparameters swept across instances. Note that $d$ represents input dimension and $C$ represents number of output classes.

| Constructor | # | Fixed | Varied Hyperparameters | Hyperparameter Descriptions |
|---|---|---|---|---|
| MEAP | 14 | $d = 100$, $C = 10$ | $P \in \{2, 16, 32, 128\}$; $\varepsilon \in \{0.05, 0.1, 0.25, 0.5\}$; $\gamma \in \{10^{-5}, 10^{-4}, 10^{-3}, 1, 100\}$ | $P$: ReLU pair count; $\varepsilon$: perturbation radius; $\gamma$: certified margin buffer. |
| MILP Exact-Radius | 31 | $d = 50$, $C = 5$ | $h_1, h_2 \in \{5, 10, 20, 50, 100\}$; $\varepsilon_{\text{frac}} \in \{0.5, 0.9, 0.99, 0.999, 0.9999, 0.99999\}$ | $h_1, h_2$: hidden-layer widths; $\varepsilon_{\text{frac}}$: Exact radius multiplier. |
| Input-Corner Stress | 22 | $d = 100$, $C = 10$, $\varepsilon = 0.2$ | `hinge_l1` $\in \{10^{-4}, 10^{-2}, 1, 10^2, 10^4, 10^6\}$; `num_hinges` $\in \{16, 256, 1024, 4096\}$ | `hinge_l1`: hinge slope scale; `num_hinges`: convex hinge count. |
| Constant-on-Box Embedding | 5 | $d = 100$, $C = 10$, $\varepsilon = 0.1$ | `wide-layers` $\in \{1, 5, 10, 20, 50\}$ | `wide-layers`: downstream network depth. |
| Deep Contractive CNN | 50 | $\varepsilon = 0.02$, $C = 10$, $k = 3$ | `depth` $\in \{2, 4, 6, 8, 10\}$; `channels` $\in \{4, 8, 16, 32, 64\}$; $\lambda \in \{0.70, 0.80, 0.90, 0.95, 0.98\}$; `margin` $\in \{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$; `bias_instability_frac` $\in \{0.10, 0.25, 0.40, 0.55, 0.70\}$ | `depth`: contractive block count; `channels`: feature-map width; $\lambda$: per-layer contraction; `margin`: certified slack floor; `bias_instability_frac`: instability placement target. |
| Paired-Biases CNN | 46 | $\varepsilon = 0.05$, $C = 10$, $k = 3$ | $P \in \{2, 4, 8, 16, 32\}$; `margin` $\in \{10^{-3}, 0.05, 0.1, 0.5, 1.0\}$; $\delta \in \{0.005, 0.01, 0.025, 0.05, 0.1\}$; `num_backbone_layers` $\in \{1, 2, 3, 4\}$; $H, W \in \{4, 8, 12, 16\}$ | $P$: paired-channel count; `margin`: global logit offset; $\delta$: bias gap half-width; `num_backbone_layers`: upstream CNN depth; $H, W$: spatial feature size. |
| Dominant-Key (Linear) Attention | 18 | $C = 10$, $d_k = n$ | $n \in \{2, 4, 6\}$; $d \in \{2, 4, 6, 8, 12\}$; $d_v \in \{2, 4, 8\}$; $\varepsilon \in \{0.02, 0.05, 0.08\}$; `margin_factor` $\in \{1.1, 1.5, 2.0, 3.0\}$; (`gate_scale`, `noise_scale`) $\in \{(0.30, 0.10), (0.50, 0.20), (0.80, 0.30)\}$ | $n$: sequence length; $d$: token dimension; $d_v$: value dimension; $\varepsilon$: perturbation radius; `margin_factor`: certificate headroom; `gate_scale`: dominant-key strength; `noise_scale`: off-key suppression. |
| Fixed-Ordering Attention | 17 | $d_v = 8$, $C = 50$ | $n \in \{2, 8, 16, 32\}$; $d \in \{4, 8, 16\}$; $\alpha \in \{2.0, 5.0, 10.0\}$; $\varepsilon \in \{5 \cdot 10^{-4}, 10^{-3}, 10^{-2}\}$; `margin_slack` $\in \{1.00001, 1.001, 1.05, 1.1, 2.0, 5.0\}$ | $n$: sequence length; $d$: token dimension; $\alpha$: attention score scale; $\varepsilon$: perturbation radius; `margin_slack`: certificate headroom. |
| Polynomial Network | 22 | C = 2 | $d \in \{50, 100\}$; `degree` $\in [2, 22]$; $h \in \{50, 100, 500\}$; $\delta \in [1 \times 10^{-4}, 5 \times 10^{-3}]$ | $d$: input dimension; `degree`: polynomial degree; $h$: hidden width; $\delta$: boundary buffer. |

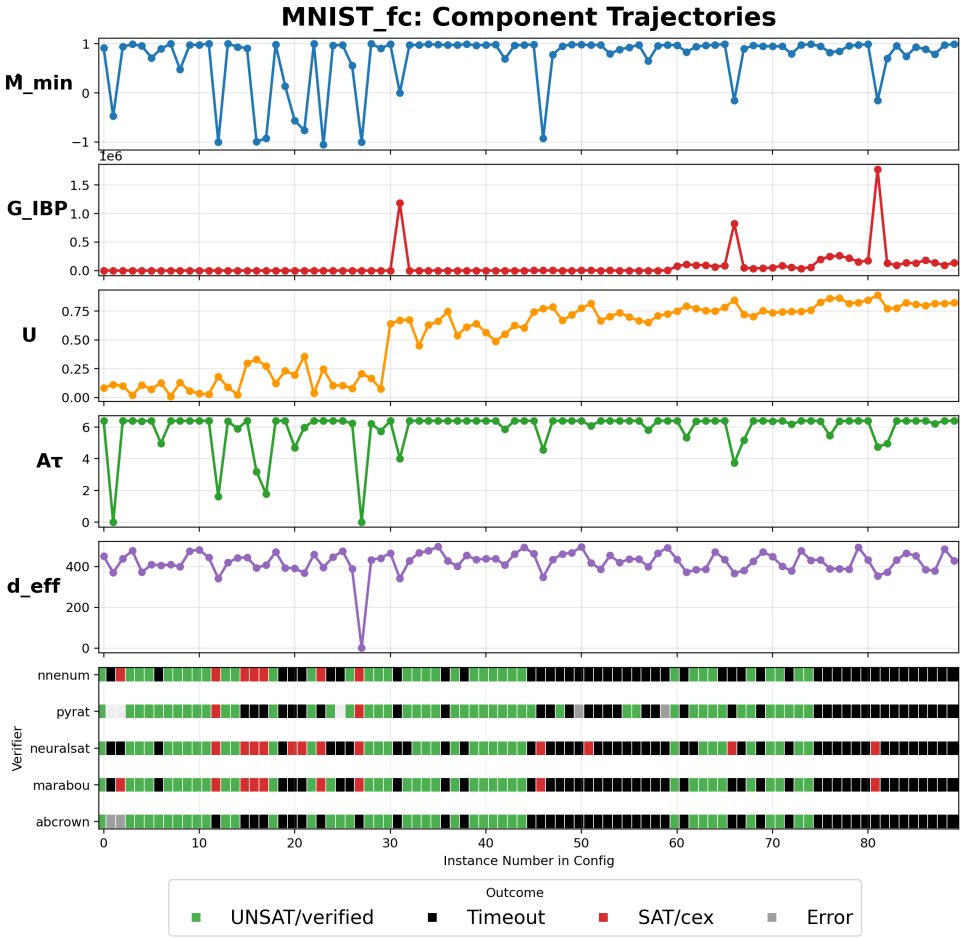



*Figure 5.* Difficulty profile component estimates and verifier outcomes for each of the 90 instances in the MNIST_fc VNNCOMP benchmark. Each instance had a timeout limit of 360 seconds per verifier.

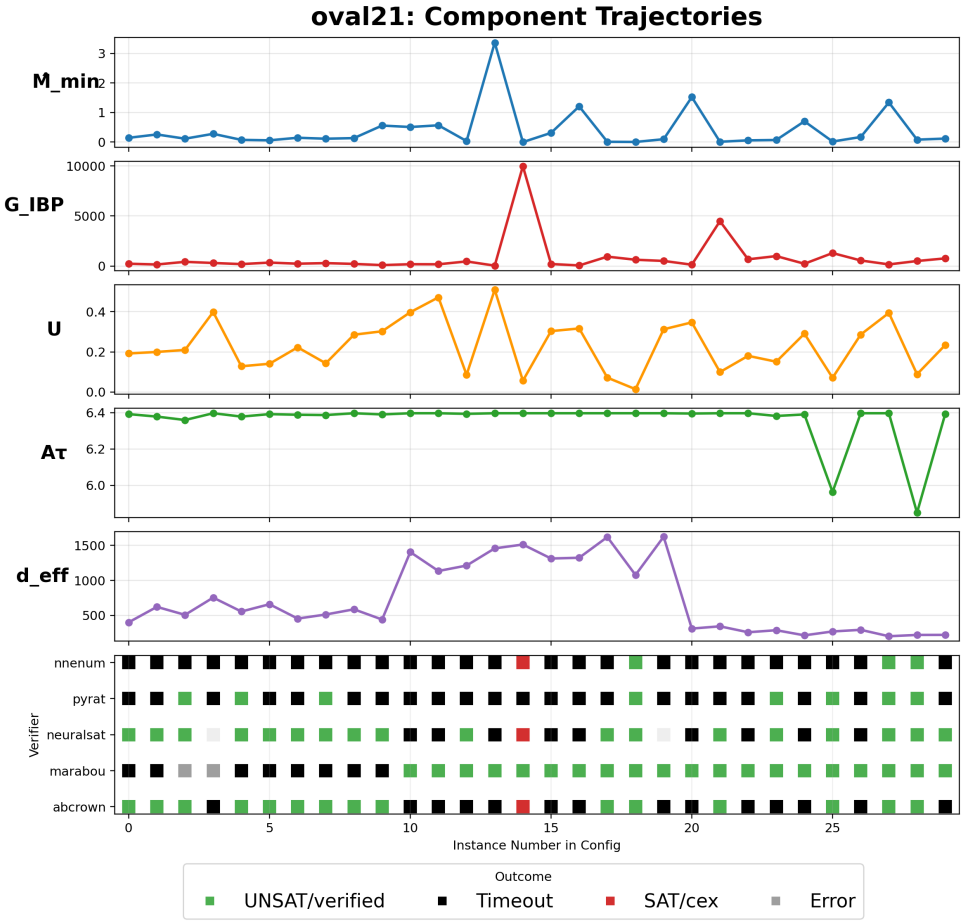



*Figure 6.* Difficulty profile component estimates and verifier outcomes for each of the 90 instances in the Oval21 VNNCOMP benchmark. Each instance had a timeout limit of 360 seconds per verifier.