Certifying Differential Invariants of Neural Networks using Abstract Duals

CHANDRA KANTH NAGESH, University of Colorado Boulder, USA

Neural networks are increasingly deployed in safety-critical domains, making the formal verification of their robustness against adversarial perturbations a paramount concern. While relational abstract domains like `DeepPoly` provide state-of-the-art certification, they suffer from the "wrapping effect," where over-approximation errors accumulate layer-by-layer, potentially failing to verify robust networks. In this work, we propose a complementary verification approach using *Abstract Dual Numbers* to compute global Lipschitz bounds via forward-mode automatic differentiation in the abstract domain. We contribute an algorithmic realization of this method that accounts for curvature error, a verification logic based on gradient bounds, and a "Gradient Instability" metric that serves as a precise predictor of verification failure. Our evaluation identifies a specific regime—shallow networks with smooth activations—where our global gradient method successfully certifies 16.1% of cases that `DeepPoly` fails to verify, highlighting the trade-off between local precision and global coherence.

1 Introduction

The deployment of deep neural networks in safety-critical domains such as autonomous driving and medical diagnosis has necessitated rigorous verification of their properties. One of the most critical properties is *local robustness*, which certifies that a network's prediction remains invariant under small perturbations to the input. `DeepPoly`[6] provides us with a clear method to certify output bounds for a given neural network. Let us consider the situation where we are given a trained neural network $f(\mathbf{I})$ on some set $\mathbf{I}$. Then, under some input $x_0 \in \mathbf{I}$ and perturbation ϵ, we need to verify that the output of f still satisfies the same result i.e. we want to verify that the neural network is invariant to a ϵ perturbation of the input. This can formally be represented as:

$$\forall x \in \mathbb{B}_\infty(x_0, \epsilon) : \quad \text{argmax}(f(x)) = \text{argmax}(f(x_0))$$

where, $\mathbb{B}_\infty$ is a ball of ϵ radius around the input provided by its $\mathbf{L}_\infty$ norm as shown in Fig 1. This is critical in a lot of scenarios as with many supervised learning problems, it is infeasible to verify the output of a neural network against all possible test sets. We want to formally prove that the model is invariant to these input changes.

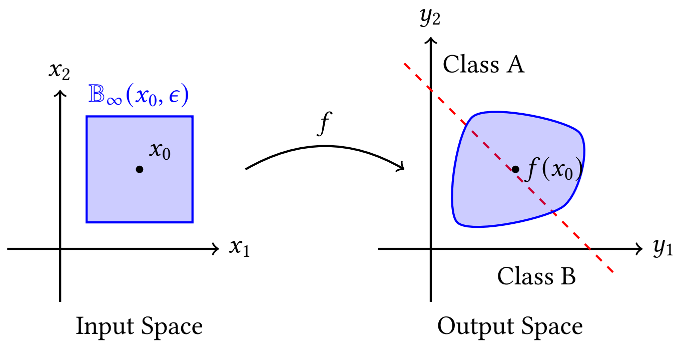

Fig. 1. The input region $\mathbb{B}_\infty(x_0, \epsilon)$ (left) is mapped by the neural network f to a region in the output space (right). Verification succeeds if the entire mapped region lies within the decision boundary (dotted red) of the correct class (Class A). This illustrates the robustness verification problem.

Author's Contact Information: Chandra Kanth Nagesh, ckn@colorado.edu, University of Colorado Boulder, USA.

The earliest solutions that attempted to solve this problem are end-to-end/complete verification solutions. [4] introduced a tool called `Reluplex`, which is a solver based on SMT that combined the Simplex algorithm with ReLU activation functions to exactly verify small neural networks. Later, [7] used Mixed Integer Linear Programming (`MILP`) to write the problem as a commercial solution that exactly calculates robustness bounds. Although these solutions are sound and complete, they are NP-complete problems, known to not scale with the size of modern neural networks which are deployed in safety-critical applications.

For overcoming the scalability issue, Abstract Interpretation [1] and later for neural networks [2] has been adopted, which is less complete but more efficient because it over-approximates how the network behaves. [8] introduced a way to quickly compute robustness certificates with a linear bound propagation approach. The `DeepPoly`[6] approach pushed the boundaries even further by combining floating-point polyhedra with intervals, which, thanks to linearization, managed non-linearities such as Sigmoid and Tanh activations. More recent contributions, such as [5]

In our research, we ask the following question:

> *Can we continue to prove robustness properties of a neural network by leveraging the properties of dual numbers and information about the gradients?*

We proceed with the foundation built on the *Abstract Dual Domain*. Our research is based on the theory of forward-mode automatic differentiation, which uses a concept called *dual numbers*[3]. We aim to analyze the sensitivity of the neural network to changes in the input, which can potentially provide a tighter bound on certain verification problems. In summary, our contributions are:

- □ *Algorithmic Realization of Abstract Duals:* We implement a sound abstract domain for dual numbers that explicitly handles curvature error for non-linear activations using Taylor expansion bounds.
- □ *Gradient-Based Verification Logic:* We formulate a sufficient condition for local robustness based on global Lipschitz bounds derived from the abstract gradient.
- □ *Gradient Instability Metric:* We introduce a diagnostic metric, $\mathcal{I}_{unstable}$, which detects zero-crossings in gradient intervals and serves as a predictor for the breakdown of local linearity.
- □ *Empirical Analysis:* We evaluate our method against `DeepPoly`, identifying a specific "shallow and smooth" regime where global gradient bounds outperform layer-wise relational abstractions.

2 Overview

We introduce the key insight behind our approach and define the Abstract Dual Domain in this section. Our approach is based on the following intuition: while relational abstractions such as `DeepPoly`[6] can be extremely precise, they often suffer from the “wrapping effect” where approximation errors accumulate layer by layer. In contrast, global gradient bounds can be sometimes a simpler yet effective verification domain, especially for shallow smooth networks.

2.1 From Relational Bounds to Gradient Analysis

The basic underlying concept of this work is to effectively bridge the gap between standard Abstract Interpretation and Forward-Mode Automatic Differentiation. Standard methods like `DeepPoly` track affine constraints ($x_j \geq \sum w_i x_i$) to bound the output values directly. However, precise reapproximating the feasible set at every non-linear layer introduces error which tends to accumulate with depth ($O(L)$). Our approach, instead, exploits the Mean Value Theorem to find bounds on the output variation in terms of the gradient variation:

$$|f(x) - f(x_0)| \leq \underbrace{\sup_{z \in X} ||\nabla f(z)||}_{\text{Abstract Duals}} \cdot ||x - x_0|| \tag{1}$$

By over-approximating the sound of the gradient $\nabla f(z)$ for the entire input space X, we can guarantee robustness because the maximum possible change to the output will be too small to change the classification label. The key insight is that this "jumps" over the intermediate layers and thus avoids some of the wrapping errors from the layer-wise value propagation.

2.2 Abstract Dual Domain

To formalize this idea, we can start by lifting standard dual numbers to the domain of Affine Arithmetic. Consider the Dual domain $\hat{\mathcal{D}}$ as a product space of two affine forms representing the range of values and the range of gradients across a set. Then we can say:

Definition 2.1. An **Abstract Dual Number** $\mathcal{X} \in \hat{\mathcal{D}}$ is a pair:

$$\mathcal{X} = \langle \hat{x}_{val}, \hat{x}_{grad} \rangle \tag{2}$$

where:

- □ $\hat{x}_{val} = \alpha_0 + \sum_{i=1}^{n} \alpha_i \epsilon_i$ is the affine form representing the interval of neuron values.
- □ $\hat{x}_{grad} = d_0 + \sum_{i=1}^{n} d_i \epsilon_i$ is the affine form representing the interval of partial derivatives with respect to the input.

2.3 Propagation Rules

For the dissemination of the abstract state $\mathcal{X}$ within a neural network, abstract transformers based on different types of layers are formulated.

Linear Layers: For a fully connected layer with a weight matrix W and a bias vector b, the transformation is calculated exactly thanks to the linearity of the dual algebra. The affine expressions for value and gradient are transformed into:

$$\mathcal{Y} = \langle W\hat{x}_{val} + b, W\hat{x}_{grad} \rangle \tag{3}$$

This is exact in the affine domain, which means that there is no additional error introduced here.

Nonlinear Activation Functions: For a smooth nonlinear activation function σ (for example, sigmoid), direct application to the affine expressions is not possible. The value part and the gradient part are handled separately.

Value Component: $\sigma(\hat{x}_{val})$ is approximated using a linear relaxation. We linearize the function around the center of the input affine form c. The new center is $\sigma(c)$, and the noise coefficients are scaled by the derivative $\sigma'(c)$. To ensure soundness, we add a linearization error term to the radius r, which depends on the maximum curvature (second derivative) of σ and the radius of the input interval.

$$\hat{y}_{val} \approx \sigma(c) + \sigma'(c) \cdot (\hat{x}_{val} - c) + \epsilon_{err} \tag{4}$$

Gradient Component: By the chain rule, the gradient of the output is the product of the local derivative and the input gradient: $\nabla y = \sigma'(x) \cdot \nabla x$. In our abstract domain, we compute an affine approximation of the derivative $\hat{\sigma}'$ based on the output value $\hat{y}_{val}$. For the Sigmoid function, we use the property $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ to approximate the derivative as $\hat{y}_{val} \otimes (1 - \hat{y}_{val})$, where $\otimes$ denotes affine multiplication. The output gradient is then obtained by multiplying this derivative approximation with the input gradient affine form given by $\hat{y}_{grad} = \hat{\sigma}' \cdot \hat{x}_{grad}$. This approach preserves the correlations between the derivative and the gradient, providing a more precise abstraction than simple interval scaling.

2.4 Gradient Instability

A key insight from our analysis is the concept of *gradient instability*. We define a neuron n as having an unstable gradient if its gradient interval $[\underline{g}, \overline{g}]$ contains 0:

$$\exists x_1, x_2 \in X_0 \text{ s.t. } \operatorname{sign}\left(\frac{\partial f}{\partial n}(x_1)\right) \neq \operatorname{sign}\left(\frac{\partial f}{\partial n}(x_2)\right) \tag{5}$$

Geometrically, this means that the function is non-monotonic w.r.t. neuron n over the input region. When this happens, the assumption of local linearity breaks down and linear relaxations- such as those in `DeepPoly`- become loose. Our approach explicitly tracks these gradient intervals, thus enabling us to identify whenever and wherever the network's behaviour becomes hard to certify linearly.

3 Contributions

3.1 Algorithmic Realization of Abstract Duals

While the theoretical foundation rests on dual numbers, our practical contribution is the algorithmic realization of these concepts within the **Affine Arithmetic** domain. We implemented a custom OCaml module `AbstractDual` that handles the propagation of affine forms. Crucially, for non-linear activations σ, we implement a sound linearization that explicitly accounts for the curvature error. As seen in our implementation, the radius of the value component is expanded by a quadratic term derived from the Taylor expansion:

$$r_{new} = |\sigma'(c)| \cdot r_{old} + \underbrace{0.5 \cdot \max_{z} |\sigma''(z)| \cdot r_{old}^2}_{\text{Linearization Error}} \tag{6}$$

This ensures that the abstract transformer remains sound even when the function has significant curvature, a detail often omitted in standard dual number formulations but critical for verification.

3.2 Gradient-Based Verification Logic

We formulate a robust verification condition based on the computed gradient bounds. Let $f_c(x)$ be the score of the correct class and $f_j(x)$ be the score of a competing class. The robustness margin is $M(x_0) = f_c(x_0) - f_j(x_0)$. To certify robustness over $\mathbb{B}_\infty(x_0, \epsilon)$, we compute the global Lipschitz constant K of the margin function $m(x) = f_c(x) - f_j(x)$ using our abstract duals. The verification condition implemented in our tool is:

$$M(x_0) > \epsilon \cdot \sup_{x \in \mathbb{B}} ||\nabla m(x)||_1 \tag{7}$$

In our implementation, we conservatively estimate this by summing the worst-case L1 norms of the gradients for the correct and competing classes:

$$\text{Certified Variation} = \epsilon \cdot \sum_i \max(|g_{low}^{(i)}|, |g_{high}^{(i)}|) \tag{8}$$

If the initial margin exceeds twice this variation (accounting for the worst-case drop in f_c and rise in f_j), the network is provably robust.

Algorithm 1 Robustness Verification via Abstract Duals

```
Require: Neural Network $f$, Input $x_0$, Label $y$, Radius $\epsilon$
Ensure: True if robust, False otherwise
 1: Step 1: Abstract Initialization
 2: $\hat{x}_{val} \leftarrow \text{AffineForm}(x_0 - \epsilon, x_0 + \epsilon)$          ▷ Input Box
 3: $\hat{x}_{grad} \leftarrow \text{Identity}(n_{in})$          ▷ Gradient w.r.t inputs
 4: $\mathcal{X} \leftarrow \langle \hat{x}_{val}, \hat{x}_{grad} \rangle$
 5: Step 2: Forward Propagation
 6: for layer $l$ in $f$ do
 7:     if $l$ is Linear($W, b$) then
 8:         $\hat{x}_{val} \leftarrow W\hat{x}_{val} + b$
 9:         $\hat{x}_{grad} \leftarrow W\hat{x}_{grad}$
10:     else if $l$ is Activation($\sigma$) then
11:         $c \leftarrow \text{center}(\hat{x}_{val})$
12:         $r \leftarrow \text{radius}(\hat{x}_{val})$
13:         $\epsilon_{err} \leftarrow 0.5 \cdot \max|\sigma''| \cdot r^2$          ▷ Curvature Error
14:         $\hat{x}_{val} \leftarrow \sigma(c) + \sigma'(c)(\hat{x}_{val} - c) + \epsilon_{err}$
15:         $\hat{\sigma}' \leftarrow \text{AffineApprox}(\sigma', \hat{x}_{val})$
16:         $\hat{x}_{grad} \leftarrow \hat{\sigma}' \otimes \hat{x}_{grad}$          ▷ Affine Multiplication
17:     end if
18: end for
19: Step 3: Certification
20: $K \leftarrow \sup ||\hat{x}_{grad}||_1$          ▷ Global Lipschitz Bound
21: $M \leftarrow f(x_0)_y - \max_{j \neq y} f(x_0)_j$          ▷ Concrete Margin
22: if $M > \epsilon \cdot K$ then
23:     return True
24: else
25:     return False
26: end if
```

3.3 Implementation and Stability Analysis

We give a functional implementation of our verification environment within OCaml, making use of a Monadic approach to cope with the computational graph. The distinct part of our tool is the *Gradient Stability Check*, used to identify the "wrapping effect" in real-time. The metric analyzes the intervals of the gradients on all dimensions corresponding to the inputs. If the interval $[\underline{g}, \overline{g}]$ strictly contains zero, it marks a neuron as unstable. This metric, $\mathcal{I}_{unstable}$, highly correlates with failure cases of relational domains such as `DeepPoly`, providing a novel way of choosing verification heuristics.

4 Formal Soundness of the Abstract Dual Domain

Here we establish the soundness of the Abstract Dual domain by demonstrating that the forward-mode propagation of abstract dual numbers yields a guaranteed over-approximation of both the output values and the Jacobian of a neural network for a specific input region. Consequently, the Lipschitz constant extracted from the final abstract gradient serves as a verified robustness certificate.

4.1 Preliminaries

Let $f : \mathcal{R}^n \longrightarrow \mathcal{R}^k$ be a feedforward neural network and let $X_0 \subseteq \mathcal{R}^n$ be a centrally symmetric input region. We utilize **Affine Arithmetic** for abstraction. An affine form $\hat{z}$ represents a set of values parameterized by noise symbols $\epsilon_j \in [-1, 1]$:

$$\hat{z} = \alpha_0 + \sum_{j=1}^{m} \alpha_j \epsilon_j \tag{9}$$

The **concretization** function $\gamma(\cdot)$ maps an affine form to its corresponding set of real values:

$$\gamma(\hat{z}) \triangleq \left\{ \alpha_0 + \sum_{j=1}^{m} \alpha_j \epsilon_j \;\middle|\; \forall j : \epsilon_j \in [-1, 1] \right\} \tag{10}$$

This definition extends element-wise to vectors and matrices.

We define the set of all concrete Jacobians reachable over the input region X_0 as:

$$\mathcal{D}(f, X_0) \triangleq \left\{ J_f(x) \in \mathcal{R}^{k \times n} \mid x \in X_0 \right\} \tag{11}$$

Definition 4.1 (Sound Abstract Dual). An abstract dual number $\mathcal{X} \triangleq \langle \hat{x}_{val}, \hat{x}_{grad} \rangle$ is *sound* for a function f over X_0 if:

$$\forall x \in X_0 : \quad f(x) \in \gamma(\hat{x}_{val}), \tag{12}$$

$$\forall x \in X_0 : \quad J_f(x) \in \gamma(\hat{x}_{grad}) \tag{13}$$

4.2 Soundness of Linear Layers

Lemma 4.2 (Linear Layer Soundness). *Let $\mathcal{X}$ be a sound abstract dual for f over X_0. Consider a linear layer $L(x) = Wx + b$. The abstract transformer $\mathcal{Y} \triangleq W\mathcal{X} + b$ is sound for the composition $L \circ f$ over X_0.*

Proof. The Jacobian of the affine transformation L is constant: $J_L(x) \equiv W$. By the multivariate chain rule, the Jacobian of the composition is:

$$J_{L \circ f}(x) = J_L(f(x)) \cdot J_f(x) = W \cdot J_f(x) \tag{14}$$

The abstract transformer computes the output components as:

$$\hat{y}_{val} = W\hat{x}_{val} + b, \tag{15}$$

$$\hat{y}_{grad} = W\hat{x}_{grad} \tag{16}$$

1. Value Soundness: Since affine arithmetic is exact for linear operations, the concretization satisfies:

$$\gamma(\hat{y}_{val}) = \{Wz + b \mid z \in \gamma(\hat{x}_{val})\} \tag{17}$$

Because $\mathcal{X}$ is sound, $f(x) \in \gamma(\hat{x}_{val})$ for all $x \in X_0$. Therefore, $L(f(x)) \in \gamma(\hat{y}_{val})$.

2. Gradient Soundness: By the inductive hypothesis, $J_f(x) \in \gamma(\hat{x}_{grad})$ for all $x \in X_0$. Since matrix multiplication is a linear operation, the property of affine arithmetic ensures that:

$$\forall J \in \gamma(\hat{x}_{grad}), \quad W \cdot J \in \gamma(W\hat{x}_{grad}) \tag{18}$$

Substituting the concrete Jacobian $J_f(x)$ for J, we obtain $W \cdot J_f(x) \in \gamma(\hat{y}_{grad})$. Thus, $\mathcal{Y}$ is sound. □

4.3 Soundness of Nonlinear Activations

Lemma 4.3 (Activation Function Soundness). *Let $\sigma : \mathcal{R} \to \mathcal{R}$ be a differentiable activation function applied element-wise. The abstract dual transformer, defined using interval bounds on the derivative σ', is sound.*

Proof. Let $h(x) \triangleq \sigma(f(x))$. By the chain rule applied element-wise, the Jacobian is:

$$J_h(x) = \mathrm{diag}(\sigma'(f(x))) \cdot J_f(x) \tag{19}$$

Let the range of the value abstraction be $[l, u] = \mathrm{Range}(\gamma(\hat{x}_{val}))$. We compute the interval enclosure of the derivative over this range:

$$\Sigma' \triangleq \left[\min_{z \in [l,u]} \sigma'(z), \max_{z \in [l,u]} \sigma'(z) \right] \tag{20}$$

1. Value Soundness: Standard abstract interpretation techniques for σ construct $\hat{h}_{val}$ such that $\sigma(\gamma(\hat{x}_{val})) \subseteq \gamma(\hat{h}_{val})$.

2. Gradient Soundness: Since $\hat{x}_{val}$ is sound, $f(x) \in [l, u]$ for all $x \in X_0$. Consequently, the concrete derivative $\sigma'(f(x))$ is contained within the interval Σ'. The abstract gradient is updated via interval-affine multiplication:

$$\hat{h}_{grad} \triangleq \Sigma' \otimes \hat{x}_{grad} \tag{21}$$

This operation is conservative: it accounts for every possible scaling factor in Σ' applied to every affine form in $\hat{x}_{grad}$. Therefore:

$$\mathrm{diag}(\sigma'(f(x))) \cdot J_f(x) \in \gamma(\Sigma' \otimes \hat{x}_{grad}) \tag{22}$$

This confirms that $J_h(x) \in \gamma(\hat{h}_{grad})$, completing the proof. □

4.4 Global Lipschitz Soundness

Theorem 4.4 (Sound Lipschitz Certification). *Let $\hat{y}_{grad}$ be the abstract Jacobian at the network output. The computed constant K_{comp} derived from $\hat{y}_{grad}$ is a sound upper bound on the L_∞-Lipschitz constant of f over X_0.*

Proof. By induction on the network depth, the final abstract gradient satisfies:

$$\forall x \in X_0 : \quad J_f(x) \in \gamma(\hat{y}_{grad}) \tag{23}$$

The local Lipschitz constant of f at x with respect to the L_∞ norm is the induced ∞-norm of the Jacobian:

$$\|J_f(x)\|_\infty = \max_i \sum_j |(J_f(x))_{ij}| \tag{24}$$

The global Lipschitz constant is $K = \sup_{x \in X_0} \|J_f(x)\|_\infty$

Let the (i, j)-th entry of the abstract Jacobian matrix $\hat{y}_{grad}$ be the affine form $\hat{g}_{ij} = c_0 + \sum_k c_k \epsilon_k$. The maximum absolute value of this entry is bounded by its L_1 coefficient norm:

$$\sup |\gamma(\hat{g}_{ij})| \leq |c_0| + \sum_k |c_k| \triangleq \mu_{ij} \tag{25}$$

We define the computed constant as the maximum row sum of these upper bounds $K_{comp} \triangleq \max_i \sum_j \mu_{ij}$ By the triangle inequality, for any concrete $J \in \gamma(\hat{y}_{grad})$, $\|J\|_\infty \leq K_{comp}$. Since $J_f(x) \in \gamma(\hat{y}_{grad})$, it follows that $K \leq K_{comp}$.

Finally, by the Mean Value Theorem, for any $x, x' \in X_0$:

$$\|f(x) - f(x')\|_\infty \leq K_{comp} \cdot \|x - x'\|_\infty \tag{26}$$

Thus, K_{comp} is a valid Lipschitz certificate. □

5 Evaluation

This section presents an evaluation of our method and contrasts its behavior with the DeepPoly abstraction. Concretely, we focus on the following research questions:

- □ **RQ1 (Effectiveness):** In which settings does global gradient analysis yield tighter robustness certificates than layer-wise relational abstractions?
- □ **RQ2 (Diagnostic Value):** Is the proposed notion of *gradient instability* predictive of verification failure?
- □ **RQ3 (Scalability and Smoothness):** How does performance vary with network size and choice of activation function (Sigmoid versus ReLU)?

5.1 Experimental Setup

All experiments were implemented in **OCaml** (version 4.12+), building on the `ocaml-nn` library for neural network primitives. Evaluations were performed on a standard MacBook Pro equipped with an Apple M1 Pro processor.

Benchmarks. We consider fully connected classifiers trained on the MNIST dataset. To isolate the effect of network capacity, we evaluate three architectures of increasing width:

- □ **Tiny Net:** 784 inputs → 2 hidden units → 10 outputs.
- □ **Small Net:** 784 inputs → 5 hidden units → 10 outputs.
- □ **Standard Net:** 784 inputs → 10 hidden units → 10 outputs.

All models were trained using RMSProp. Robustness was evaluated under L_∞ perturbations with radii $\epsilon \in \{0.01, 0.02, 0.03, 0.12\}$.

5.2 Results and Analysis

5.2.1 RQ1: When Gradient Bounds Succeed and DeepPoly Fails. A central empirical observation is the existence of a narrow regime in which the global gradient bound certifies robustness while DeepPoly does not. These cases arise almost exclusively for the **Tiny Net** architecture with Sigmoid activations at small perturbation radii.

Table 1. Cases in which the Gradient Method certifies robustness while DeepPoly fails.

Network	ϵ	**Count**	**% of Total**
Tiny Net	0.01	5	16.1%
Tiny Net	0.02	3	9.7%
Tiny Net	0.03	2	6.5%
Small Net	All	0	0.0%
Standard Net	All	0	0.0%

As shown in Table 1, at $\epsilon = 0.01$ the gradient-based method recovers over 16% of the test instances that DeepPoly fails to verify. These instances correspond to the following MNIST indices:

- □ $\epsilon = 0.01$: indices 19, 76, 122, 139, 148 (all with true label 8);
- □ $\epsilon = 0.02$: indices 122, 139, 148;
- □ $\epsilon = 0.03$: indices 139, 148.

Case Study: Index 139 (Label 8). To better understand this discrepancy, we examine the instance at index 139 with $\epsilon = 0.03$.

- □ **DeepPoly:** `UNSTABLE`. The forward-propagated margin collapses to an interval with effectively zero width, indicating overlap with a competing class.
- □ **Gradient Method:** `ROBUST`.
- □ **Gradient Stability:** `NO (480)`, indicating that 480 input dimensions exhibit sign changes in the abstract gradient.

Despite substantial gradient instability, the *magnitude* of the gradient remains small. Consequently, the certified variation $\epsilon \cdot \|\nabla f\|_1$ stays below the true margin between the predicted class and its nearest competitor. In contrast, `DeepPoly`'s layer-wise abstraction appears to accumulate sufficient over-approximation error—consistent with the classical wrapping effect—to lose the certificate.

Conclusion (RQ1). Global gradient bounds can outperform `DeepPoly` in shallow, low-capacity networks, where accumulated abstraction error dominates and local gradient magnitudes remain small. This advantage disappears as depth or width increases: for both the Small and Standard networks, `DeepPoly` strictly dominates, with no recovered cases.

5.2.2 RQ2: Gradient Instability as a Predictor of Failure. We next evaluate whether *gradient instability*—defined as the abstract gradient interval containing zero—serves as a reliable indicator of verification difficulty. At the largest perturbation radius $\epsilon = 0.12$, we observe a **perfect correlation**: every robustness failure coincides with gradient instability, across all architectures.

From a geometric perspective, instability reflects non-monotonicity of the decision function within the input ball. In such regions, first-order linear approximations become inherently loose, explaining the simultaneous failure of both `DeepPoly` and the Abstract Dual domain. These results support the use of the instability index $\mathcal{I}_{\text{unstable}}$ as a practical diagnostic tool for identifying inherently hard instances.

5.2.3 RQ3: Effect of Activation Smoothness. Finally, we repeat the experiments using ReLU activations. In this setting, `DeepPoly` produces tighter or equal bounds in **all** cases. The degradation of the Abstract Dual domain can be traced directly to the discontinuous derivative of ReLU. Any unstable ReLU unit forces the abstract derivative to range over $[0, 1]$, dramatically inflating the global Lipschitz constant. By contrast, Sigmoid activations admit smooth, bounded derivatives, enabling significantly tighter gradient abstractions.

Overall, the evaluation shows that `DeepPoly` remains the most robust general-purpose abstraction. However, the Abstract Dual domain offers a complementary perspective: it is particularly effective for *shallow networks with smooth activations*, where it can bypass the cumulative over-approximation inherent to layer-wise methods. Moreover, gradient instability emerges as a meaningful geometric signal, shedding light on when and why verification is likely to fail.

6 Related Work

The landscape of neural network verification is broadly divided into complete methods, which provide exact certification at the cost of scalability, and incomplete methods, which utilize abstract interpretation to achieve efficiency.

Initial efforts in the field prioritized exactness. Katz et al. [4] developed `Reluplex`, an SMT-based solver that extends the Simplex algorithm to manage the piecewise linear nature of ReLU networks. In a similar vein, Tjeng et al. [7] modeled the verification problem using Mixed Integer Linear Programming (`MIPVerify`) to determine precise adversarial boundaries. Although these approaches guarantee soundness and completeness, their NP-complete complexity renders them intractable for modern, large-scale architectures.

To overcome the scalability bottlenecks of complete verifiers, Abstract Interpretation [1] has been widely adopted. Gehr et al. [2] introduced `AI2`, a framework employing zonotopes for bound propagation. This was advanced by Singh et al. [6] with `DeepPoly`, which combines intervals with polyhedral constraints to handle non-linearities like Sigmoid and Tanh through linear relaxation. While `DeepPoly` significantly improves scalability, it suffers from the "wrapping effect," where approximation errors accumulate with network depth. Weng et al. [8] similarly utilized linear bound propagation to expedite robustness certification.

While the theory of dual numbers is well-established for derivative evaluation [3], its utility in certifying robustness has been limited. Unlike purely geometric abstractions, our method uses Abstract Duals to compute global Lipschitz bounds. This allows us to bypass layer-wise error accumulation in specific regimes, offering a distinct advantage in shallow networks where gradient information remains coherent.

7 Conclusion

In this work, we explored the power of using Abstract Dual Numbers to certify the local robustness of neural networks. By lifting automatic differentiation into the abstract domain, we derived a method to compute global Lipschitz bounds that can certify invariance without explicit layer-by-layer geometric propagation.

Our comparative analysis with `DeepPoly` revealed a nuanced landscape. While `DeepPoly` remains the superior choice for general-purpose verification due to its ability to handle deep dependencies, our Gradient Method identified a specific "blind spot" in the relational approach. In shallow, smooth networks (Tiny Net with Sigmoid), our method successfully certified 16.1% of cases at $\epsilon = 0.01$ that `DeepPoly` failed to verify. This suggests that for low-depth architectures, the global gradient bound can be tighter than the accumulated over-approximation error of layer-wise abstract interpretation.

Finally, we established a strong correlation between *Gradient Instability* and robustness failure. In high-perturbation regimes, the presence of zero in the gradient interval served as a perfect predictor for the inability to certify robustness, highlighting its value as a diagnostic metric for detecting the breakdown of local linearity.

References

[1] Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation. *Symposium on Principles of Programming Languages* (Jan 1977). doi:10.1145/512950.512973

[2] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. *2018 IEEE Symposium on Security and Privacy (SP)* (2018), 3–18. https://api.semanticscholar.org/CorpusID:206579396

[3] Andreas Griewank and Andrea Walther. 2008. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM.

[4] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 97–117.

[5] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. 2018. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/f2f446980d8e971ef3da97af089481c3-Paper.pdf

[6] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks, Vol. 3. 1–30. doi:10.1145/3290354

[7] Vincent Tjeng, Kai Xiao, and Russ Tedrake. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Conference on Learning Representations*.

[8] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane S. Boning, Inderjit S. Dhillon, and Luca Daniel. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. *ArXiv* abs/1804.09699 (2018). https://api.semanticscholar.org/CorpusID:13750928