Certifying Differential Invariants of Neural Networks using Abstract Duals

CHANDRA KANTH NAGESH, University of Colorado Boulder, USA

Neural networks are increasingly used in safety-critical domains necessitating formal guarantees about their properties under perturbations to inputs. Existing robust verification techniques, typified by `DeepPoly` primarily focus on robustness analysis during forward mode where certifying output stability is given by $f(\mathbf{I}) \subseteq [\mathbf{y}_L, \mathbf{y}_R]$ for an $\mathbf{L}_\infty$-constrained input set $\mathbf{I} = \{\mathbf{x} \| \mathbf{x} - \mathbf{x}_0\|_\infty \leq \epsilon\}$. While such methods employ a sophisticated polyhedral abstract domain (combining intervals and affine forms) to generate sound, tight bounds, this entire class of analysis still fails to provide sound guarantees over the behaviour. This oversight creates a critical verification gap for gradient-dependent systems where the Backward Pass Verification Problem which can be described formally as bounding the Jacobian $\mathbf{J}(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x})$ such that $\mathbf{J}(\mathbf{x}) \subseteq [\mathbf{J}_L, \mathbf{J}_R]$ for all $\mathbf{x} \in \mathbf{I}$ is essential.

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.

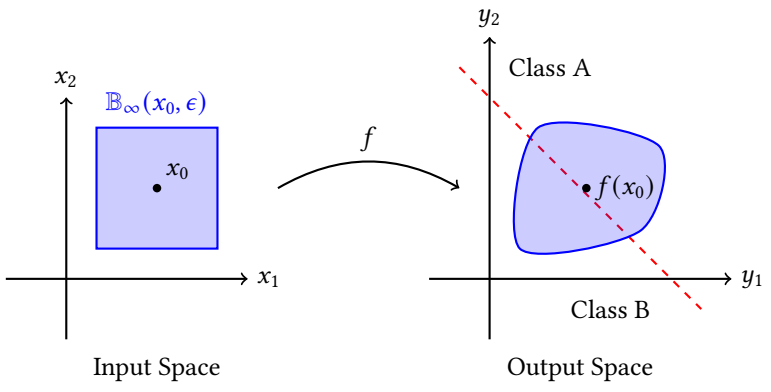

Fig. 1. The input region $\mathbb{B}_\infty(x_0, \epsilon)$ (left) is mapped by the neural network f to a region in the output space (right). Verification succeeds if the entire mapped region lies within the decision boundary (dotted red) of the correct class (Class A). This illustrates the robustness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

2 Overview

In this section, we introduce the key insight behind our approach and define the Abstract Dual Domain. 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 Paradigm Shift: 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

To propagate $\mathcal{X}$ through a neural network, we define abstract transformers for each layer type.

Linear Layers: For a fully connected layer with weight matrix W and bias vector b, the transformation is exactly determined by the linearity of the dual algebra. The affine forms for value and gradient are transformed as follows:

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

This operation is exact in the affine domain and introduces no new approximation errors.

Non-linear Activations: For a smooth non-linear activation function σ (e.g., Sigmoid), we cannot directly apply the function to the affine forms. We handle the value and gradient components separately:

Value Component: We approximate $\sigma(\hat{x}_{val})$ 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:

$$\hat{y}_{grad} = \hat{\sigma}' \cdot \hat{x}_{grad} \tag{5}$$

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:

$$\exists x_1, x_2 \in X_0 \text{ s.t. } \operatorname{sign}\left(\frac{\partial f}{\partial n}(x_1)\right) \neq \operatorname{sign}\left(\frac{\partial f}{\partial n}(x_2)\right) \tag{6}$$

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 Theoretical Results: Soundness and Instability

4 Formal Proof of Soundness

In this section, we establish the mathematical validity of the Abstract Dual domain. We demonstrate that the interval of gradients produced by our forward-mode propagation soundly over-approximates the true range of partial derivatives of the neural network f over the input region X_0.

4.1 Foundational Definitions

Let $f : \mathcal{R}^n \to \mathcal{R}^k$ be a neural network. Let $X_0 \subseteq \mathcal{R}^n$ be a centrally symmetric input region represented by the affine form $\hat{x}_{val}$. We define the **Concrete Derivative Set** as follows:

$$\mathcal{D}(f, X_0) = \{\nabla f(x) \in \mathcal{R}^{k \times n} \mid x \in X_0\} \tag{7}$$

Definition 4.1 (Soundness of Abstract Duals). An Abstract Dual Number $\mathcal{X} = \langle \hat{x}_{val}, \hat{x}_{grad} \rangle$ is considered **sound** with respect to a function f and input region X_0 if and only if:

(1) $\forall x \in X_0 : f(x) \in \gamma(\hat{x}_{val})$
(2) $\forall x \in X_0 : \nabla f(x) \in \gamma(\hat{x}_{grad})$

where $\gamma(\hat{x})$ denotes the concretization function mapping an affine form to its corresponding subset of $\mathcal{R}$.

4.2 Linear Layer Soundness

Lemma 4.2 (Soundness of Linear Transformers). *Given a sound abstract dual $\mathcal{X}$ and a linear transformation $y = Wx + b$, the abstract transformer $\mathcal{Y} = W\mathcal{X} + b$ is sound.*

Proof. By the fundamental rules of multi-variable calculus, if y is a linear function of x defined by $y = Wx + b$, the Jacobian matrix is constant: $\nabla_x y = W$. By the chain rule, for a composite function $y(f(x))$, we have:

$$\nabla y = W \cdot \nabla f(x) \tag{8}$$

In our abstract domain, we define:

$$\hat{y}_{val} = W\hat{x}_{val} + b, \quad \hat{y}_{grad} = W\hat{x}_{grad} \tag{9}$$

Since affine arithmetic is a linear abstraction, it is exact for linear transformations (i.e., $\gamma(W\hat{x} + b) = \{Wx + b \mid x \in \gamma(\hat{x})\}$). Given that $\mathcal{X}$ is sound, $\gamma(\hat{x}_{grad})$ contains all possible values of $\nabla f(x)$. Therefore, $\gamma(W\hat{x}_{grad})$ must contain $W \cdot \nabla f(x)$. This concludes that $\mathcal{Y}$ is sound. □

4.3 Activation Function Soundness

Lemma 4.3 (Soundness of Non-linear Activation Transformers). *Let σ be a differentiable activation function (e.g., Sigmoid, Tanh). The abstract dual transformer $\sigma^{\#}(\mathcal{X})$ defined by interval derivative bounding is sound.*

Proof. Consider the composite function $h(x) = \sigma(f(x))$. By the univariate chain rule applied element-wise:

$$\nabla h(x) = \sigma'(f(x)) \cdot \nabla f(x) \tag{10}$$

Let $[l, u]$ be the interval range of the values $\hat{x}_{val}$. We compute the derivative range Σ' as:

$$\Sigma' = \left[\inf_{z \in [l,u]} \sigma'(z), \sup_{z \in [l,u]} \sigma'(z) \right] \tag{11}$$

The abstract gradient is then computed as the product $\hat{h}_{grad} = \Sigma' \cdot \hat{x}_{grad}$. By the Mean Value Theorem, for any $x \in X_0$, the concrete value $f(x)$ must lie within $[l, u]$. Consequently, the concrete derivative $\sigma'(f(x))$ must be contained within the interval Σ'. Using the soundness property of interval-affine multiplication, the resulting affine form $\hat{h}_{grad}$ is guaranteed to enclose the product of any value in Σ' and any vector in $\gamma(\hat{x}_{grad})$. Thus, $\forall x \in X_0 : \nabla h(x) \in \gamma(\hat{h}_{grad})$. □

4.4 Main Theorem: Lipschitz Certification

Theorem 4.4 (Global Lipschitz Soundness). *The Lipschitz constant K_{comp} derived from the output Abstract Dual $\mathcal{Y}_{out}$ is a sound upper bound for the function f over the region X_0.*

Proof. By induction over the network depth L, using Lemmas 1 and 2, the final gradient affine form $\hat{y}_{grad}$ at the output layer is a sound over-approximation of the set $\mathcal{D}(f, X_0)$. The global Lipschitz constant for f with respect to the L_∞ norm is defined as:

$$K = \sup_{x \in X_0} \|\nabla f(x)\|_1 \tag{12}$$

For any affine form $\hat{g} = \alpha_0 + \sum_{i=1}^{n} \alpha_i \epsilon_i$, the supremum of its absolute value is soundly bounded by:

$$\sup |\gamma(\hat{g})| \leq |\alpha_0| + \sum_{i=1}^{n} |\alpha_i| \tag{13}$$

We define K_{comp} as the sum of these absolute coefficients for the output gradient affine form. It follows that $K_{comp} \geq K$. By the Mean Value Theorem:

$$\forall x \in X_0 : \|f(x) - f(x_0)\|_\infty \leq K_{comp} \cdot \|x - x_0\|_\infty \tag{14}$$

Thus, if the safety condition $f(x_0)_{target} - f(x_0)_{other} > K_{comp} \cdot \epsilon$ is satisfied, the classification is guaranteed to be invariant within the ϵ-ball. □

Acknowledgments

TBD

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