AI^2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation

Timon Gehr Matthew Mirman Dana Drachsler-Cohen Petar Tsankov Swarat Chaudhuri Martin Vechev

I. Introduction

Adversarial examples can be especially problematic when safety-critical systems rely on neural networks. For instance, it has been shown that attacks can be executed physically (e.g., [5], [12]) and against neural networks accessible only as a black box (e.g., [7], [18], [20]). To mitigate these issues, recent research has focused on reasoning about neural network robustness, and in particular on *local robustness*. Local robustness (or robustness, for short) requires that all samples in the neighborhood of a given input are classified with the same label [15]. Many works have focused on designing *defenses* that increase robustness by using modified procedures for training the network (e.g., [7], [8], [14], [15], [19]). Others have developed approaches that can show non-robustness by underapproximating neural network behaviors [1] or methods that decide robustness of small fully connected feedforward networks [10]. However, no existing sound analyzer handles convolutional networks, one of the most popular architectures.

The main challenge facing sound analysis of neural networks is scaling to large classifiers while maintaining a precision that suffices to prove useful properties. The analyzer must consider all possible outputs of the network over a prohibitively large set of inputs, processed by a vast number of intermediate neurons.

To avoid this state space explosion, current methods (e.g., [9], [10], [16]) symbolically encode the network as a logical formula and then check robustness properties with a constraint solver. However, such solutions do not scale to larger (e.g., convolutional) networks, which usually involve many intermediate computations.

II. Contributions

The key insight of our work is to address this challenge by leveraging the classic framework of abstract interpretation (e.g., [2], [3]), a theory which dictates how to obtain sound, computable, and precise finite approximations of potentially infinite sets of behaviors. Concretely, we leverage numerical abstract domains – a particularly good match, as AI systems tend to heavily manipulate numerical quantities. By showing how to apply abstract interpretation to reason about AI safety, we enable one to leverage decades of research and any future advancements in that area (e.g., in numerical domains [17]). With abstract interpretation, a neural network computation is overapproximated using an *abstract domain*. An abstract domain consists of logical formulas that capture certain shapes (e.g., zonotopes, a restricted form of polyhedra).

Based on this insight, we developed a system called AI^2 (*Abstract Interpretation for Artificial Intelligence*). AI^2 is the first scalable analyzer that handles common network layer types, including fully connected and convolutional layers with rectified linear unit activations (ReLU) and max pooling layers.

Given a neural network and an input specification, abstract interpretation computes an abstract output, which is an overapproximation of *all* possible concrete outputs. This enables AI^2 to verify safety properties such as robustness directly on the abstract output.

We evaluated AI^2 on important tasks such as verifying robustness and comparing neural network defenses.

Our main contributions are:

- A sound and scalable method for analysis of deep neural networks based on abstract interpretation.
- AI^2, an end-to-end analyzer, extensively evaluated on feed-forward and convolutional networks (computing with 53 000 neurons), far exceeding capabilities of current systems .
- An application of AI^2 to evaluate *provable robustness* of neural network defenses.

III. Neural Networks

We give a short introduction to neural networks. This introduction covers the type of neural network that can be analyzed by the presented version of the AI^2 system.

a) Layers: Layers are functions. Neural networks are often organized as a sequence of layers, such that the neural network is their composition.

b) Activation Functions: Typically, neural network layers perform a linear transformation followed by a nonlinear activation function applied individually to all components of the input. We focus on the commonly-used ReLU activation: $\mathrm{ReLU}(x) = \max(x, 0)$.

c) Fully-connected Layer: A fully-connected layer $\mathrm{FC}_{W,b} \colon \mathbb{R}^m \to \mathbb{R}^n$ is parameterized by a weight matrix $W \in \mathbb{R}^{n \times m}$ and a bias vector $b \in \mathbb{R}^n$ and is defined as $\mathrm{FC}_{W,b}(x) = \mathrm{ReLU}(W \cdot x + b)$.

d) Convolutional Layer: A convolutional layer $\mathrm{Conv}_{W,b} : \mathbb{R}^{m \times n \times r} \to \mathbb{R}^{m-p+1} \times n-q+1 \times t$ is parameterized by weights $W \in \mathbb{R}^{p \times q \times r \times t}$ and a bias $b \in \mathbb{R}^t$. A convolutional layer computes t convolutions of the input with different filters, before it applies the ReLU activation function.

e) Max Pooling layer: The max pooling layer $\text{MaxPool}_{p,q}\colon \mathbb{R}^{n\times m} \to \mathbb{R}^{n/p\times m/q}$ partitions the input into $p \times q$ subrectangles and replaces each of them by their maximum.

IV. Abstract Interpretation

Given a function $f\colon \mathbb{R}^m \to \mathbb{R}^n$, a set of inputs $X \subseteq \mathbb{R}^m$, and a property $C \subseteq \mathbb{R}^n$, the goal of abstract interpretation is to determine whether the property holds, that is, whether for any input x in X, the output $f(x)$ is in C.

a) Abstract Interpretation: Abstract interpretation uses *abstract transformers* which operate on abstract domains $\mathcal{A}^n$, whose elements be represented and processed on a finite computer. Each element $a \in \mathcal{A}^n$ describes a set $\gamma(a) \subseteq \mathbb{R}^n$ of possible concrete values. An abstract transformer $T_f^{\#}\colon \mathcal{A}^m \to \mathcal{A}^n$ propagates such a set through a function $f\colon \mathbb{R}^m \to \mathbb{R}^n$.

In the end, the property can be verified directly on the abstract output. This abstract output may represent a strict superset of the set of possible concrete outputs.

Accordingly, abstract interpretation is sound, but incomplete: A property proved using abstract interpretation holds, but a particular abstract domain will normally not be able to prove all true properties.

Our approach can be instantiated with different abstract domains. For our purposes, an abstract domain has to support the following operations:

- A bottom element $\bot$, representing the empty set.
- Meet (intersect) an abstract element with a conjunction of linear constraints.
- Join (union) between two abstract elements.
- Apply an affine transformation to an abstract element.

Those operations do not need to be precise, but they need to be *sound*: the resulting abstract element must represent a superset of the set of concrete results obtained when applying the concrete version of the operation to the sets represented by the input abstract elements.

V. Example Abstract Domains

In this section, we provide an incomplete set of example abstract domains.

a) Box Domain: Abstract interpretation with the box domain corresponds to evaluation using interval arithmetic.

b) Zonotope Domain [6]: A zonotope $z \in Z^n$ is either $\bot$, or it can be represented as a matrix $M^{n\times m}$ for some $m \in \mathbb{N}$, and a center $c \in \mathbb{R}^n$. In this case, it in turn represents the set $\gamma(z) = \{M \cdot \epsilon + c \mid \epsilon \in [0,1]^m\}$. The zonotope domain supports an exact affine transformer, while meet and join necessarily produce a sound approximation, because an exact result may not be representable as a zonotope.

c) Polyhedra Domain [4]: The polyhedra domain consists of abstract elements that represent a convex polyhedron and are in turn represented as a set of linear constraints over the input variables. The polyhedra domain supports an exact affine transformer as well as an exact meet, while join produces a convex hull, the best possible sound overapproximation .

d) Bounded Powerset Domains (e.g. Zonotope$N$): The bounded powerset domain is parameterized with a base domain. An element in the bounded powerset domain is a set from the base domain with bounded size. Such abstract elements represent the union of the sets represented by their elements. For example, elemens of Zonotope$N$ consist of N elements of Zonotope.

VI. Abstract Transformers for Neural Networks

As abstract interpretation is composable, it suffices to define abstract transformers for each of the types of layers.

a) Fully-connected Layer, Convolutional Layer: ReLU applied to the i-th component of the abstract element $a \in \mathcal{A}^n$ can be represented as $g_i(a \sqcap (x_i < 0)) \sqcup (a \sqcap (x_i \geq 0))$, where $g_i\colon \mathbb{R}^n \to \mathbb{R}^n$ is an affine transformation that assigns 0 to the i-th component of the input. Both the fully-connected and the convolutional layer are particular affine transformations followed by componentwise ReLU, therefore we can obtain abstract transformers for them by composing ReLU abstract transformers for all components with an affine transformer.

b) Max Pooling Layer: The abstract transformer for the max pooling layer operates on one subrectangle at a time. For one such subrectangle, it uses meets with linear constraints to express a case distinction with one case for each possible location of the maximal element within the subrectangle. For each case, it then uses the affine transformer to extract that particular element. The results from all cases are joined, and the abstract transformers for all subrectangles are composed together.

VII. Results

We have experimentally evaluated AI^2 on neural networks for two well-known classification tasks: MNIST [13] and CIFAR-10 [11].

a) Experiments: We were able to show the following results:

- AI^2 can prove useful robustness properties for convolutional networks with $53\,000$ neurons and large fully connected feedforward networks with $1\,800$ neurons.
- AI^2 benefits from more precise abstract domains: Zonotope enables AI^2 to prove substantially more properties over Box. Further, Zonotope$N$, with $N \geq 2$, can prove stronger robustness properties than Zonotope alone.
- AI^2 scales better than the SMT-based Reluplex [10]: AI^2 is able to verify robustness properties on large networks with ≥ 1200 neurons within few minutes, while Reluplex takes hours to verify the same properties.

b) Defenses: Additionally, we have evaluated state-of-the art neural network defenses [7], [14], [19]). Those neural network defenses modify the way that the neural network is trained, to make it more robust (but without formal guarantees). We showed that different defenses produce neural networks that differ significantly in how amenable they are to verification using our AI^2 system.

REFERENCES

[1] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS)*, pages 2621–2629, 2016.

[2] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.

[3] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM Symposium on Principles of Programming Languages (POPL)*, pages 238–252, 1977.

[4] Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the 5th ACM Symposium on Principles of Programming Languages (POPL)*, pages 84–96, 1978.

[5] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017.

[6] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. The zonotope abstract domain taylor1+. In *Proceedings of the 21st International Conference on Computer Aided Verification (CAV)*, pages 627–633, 2009.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

[8] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, 2014.

[9] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *Computer Aided Verification, 29th International Conference (CAV)*, pages 3–29, 2017.

[10] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification, 29th International Conference (CAV)*, pages 97–117, 2017.

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[12] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[13] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[15] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016.

[16] Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In *Computer Aided Verification, 22nd International Conference (CAV)*, 2010.

[17] Gagandeep Singh, Markus Püschel, and Martin Vechev. Fast polyhedra abstract domain. In *Proceedings of the 44th ACM Symposium on Principles of Programming Languages (POPL)*, pages 46–59, 2017.

[18] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[19] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[20] Florian Tramèr, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. The space of transferable adversarial examples. *CoRR*, abs/1704.03453, 2017.