

AI²: Safety and Robustness Certification of Neural Networks with Abstract Interpretation

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

S&P 2018

Table of Contents

- 1 Introduction
- 2 Representing Neural Networks As Conditional Affine Transformations
- 3 Background
- 4 AI²: AI For Neural Networks
- 5 Evaluation of AI²







Table of Contents

- 1 Introduction
- 2 Representing Neural Networks As Conditional Affine Transformations
- 3 Background
- 4 AI^2 : AI For Neural Networks
- 5 Evaluation of AI^2

Introduction

Recent years have shown a wide adoption of deep neural networks in safety-critical applications, including self-driving cars, malware detection, and aircraft collision avoidance detection.

Despite their success, a fundamental challenge remains: to ensure that machine learning systems, and deep neural networks in particular, behave as intended.

Attack	Original	Perturbed	Diff
FGSM [12], $\epsilon = 0.3$			
Brightening, $\delta = 0.085$			

Introduction

Adversarial examples can be especially problematic when safety-critical systems rely on neural networks.

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.

No existing sound analyzer handles **convolutional networks**, one of the most popular architectures.

Introduction

Key Challenge: Scalability and Precision

The main challenge facing sound analysis of neural networks is scaling to large classifiers while maintaining a precision that suffices to prove useful properties.

Proving the property by running a network exhaustively on all possible input images and checking if all of them are classified as 8 is infeasible.

To avoid this state space explosion, current methods symbolically encode the network as a logical formula and then check robustness properties with a constraint solver.

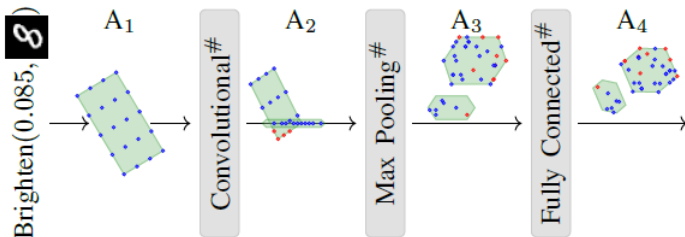
However, such solutions do not scale to larger (e.g., convolutional) networks, which usually involve many intermediate computations.

Introduction

Key Concept: Abstract Interpretation for AI

The key insight of their work is to address the above challenge by leveraging the classic framework of abstract interpretation

By showing how to apply abstract interpretation to reason about AI safety, they enable one to leverage decades of research and any future advancements in that area.



Introduction

The AI² (Abstract Interpretation for Artificial Intelligence) Analyzer

AI² 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.

Defining sound and precise, yet scalable abstract transformers is key to the success of an analysis based on abstract interpretation.

They define abstract transformers for all three layer types.

At the end of the analysis, the abstract output is an overapproximation of all possible concrete outputs.

AI² proved that the FGSM attack is unable to generate adversarial examples from image in Fig 1 for any ϵ between 0 and 0,3.

Introduction

Contributions

- A sound and scalable method for analysis of deep neural networks based on abstract interpretation.
- AI2, 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 AI2 to evaluate provable robustness of neural network defenses.

Table of Contents

- 1 Introduction
- 2 Representing Neural Networks As Conditional Affine Transformations
- 3 Background
- 4 AI²: AI For Neural Networks
- 5 Evaluation of AI²

Representing Neural Networks As Conditional Affine Transformations

They provide background on feedforward and convolutional neural networks and show how to transform them into a representation amenable to abstract interpretation.

Representing Neural Networks As Conditional Affine Transformations

CAT Functions

They express the neural network as a composition of conditional affine transformations (CAT), which are affine transformations guarded by logical constraints.

$$\begin{array}{l} \hline f(\bar{x}) ::= W \cdot \bar{x} + \bar{b} \\ \quad | \quad \text{case } E_1: f_1(\bar{x}), \dots, \text{case } E_k: f_k(\bar{x}) \\ \quad | \quad f(f'(\bar{x})) \\ E ::= E \wedge E \mid x_i \geq x_j \mid x_i \geq 0 \mid x_i < 0 \\ \hline \end{array}$$

Figure: Definition of CAT functions.

Representing Neural Networks As Conditional Affine Transformations

Reshaping of Inputs

Layers often take three-dimensional inputs (e.g colored images).

A three dimensional array $\bar{x} \in \mathbb{R}^{m \times n \times r}$ can be reshaped to $\bar{x}^v \in \mathbb{R}^{m \cdot n \cdot r}$ in a canonical way, first by depth, then by column, finally by row. That is, given \bar{x} :

$$\bar{x}^v = (x_{1,1,1} \dots x_{1,1,r} \ x_{1,2,1} \dots x_{1,2,r} \dots x_{m,n,1} \dots x_{m,n,r})^T.$$

Representing Neural Networks As Conditional Affine Transformations

ReLU to CAT

They express the ReLU activation function as

$$ReLU = ReLU_n \circ \dots \circ ReLU_1$$

where $ReLU_i$ processes the i th entry of the input \bar{x} and is given by:

$$ReLU_i(\bar{x}) = \text{case } (x_i \geq 0): \bar{x}, \\ \text{case } (x_i < 0): I_{i \leftarrow 0} \cdot \bar{x}.$$

$I_{i \leftarrow 0}$ is the identity matrix with the i th row replaced by zeros.

Representing Neural Networks As Conditional Affine Transformations

Fully Connected (FC) Layer

Formally, an FC layer with n neurons is a function $FC_{W,\bar{b}} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ parameterized by a weight matrix $W \in \mathbb{R}^{n \times m}$ and a bias $\bar{b} \in \mathbb{R}^n$. For $\bar{x} \in \mathbb{R}^m$ we have:

$$FC_{W,\bar{b}}(\bar{x}) = \text{ReLU}(W.\bar{x} + \bar{b})$$

Representing Neural Networks As Conditional Affine Transformations

Fully Connected (FC) Layer

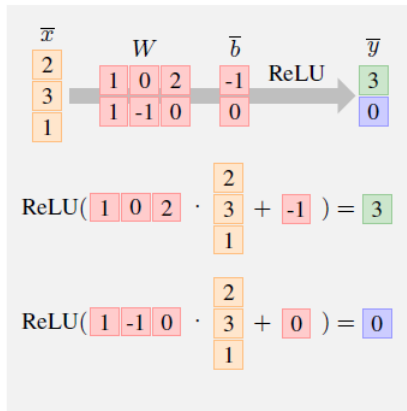


Figure: Fully connected layer FC_W, \bar{b}

Representing Neural Networks As Conditional Affine Transformations

Convolutional Layer

A convolutional layer is defined by a series of t filters

$$F^{p,q} = \{F_1^{p,q}, \dots, F_t^{p,q}\}.$$

A filter takes a three-dimensional array and returns a two-dimensional array:

$$F_i^{p,q} : \mathbb{R}^{m \times n \times r} \rightarrow \mathbb{R}^{(m-p+1) \times (n-q+1)}.$$

The entries of the output y for a given input \bar{x} are given by:

$$y_{i,j} = \text{ReLU}\left(\sum_{i'=1}^p \sum_{j'=1}^q \sum_{k'=1}^r W_{i',j',k'} \cdot x_{(i+i'-1), (j+j'-1), k'} + b\right).$$

Representing Neural Networks As Conditional Affine Transformations

Convolutional Layer

The function ConvF , corresponding to a convolutional layer with t filters, has the following type:

$$\text{ConvF}: \mathbb{R}^{m \times n \times r} \rightarrow \mathbb{R}^{(m-p+1) \times (n-q+1) \times t}.$$

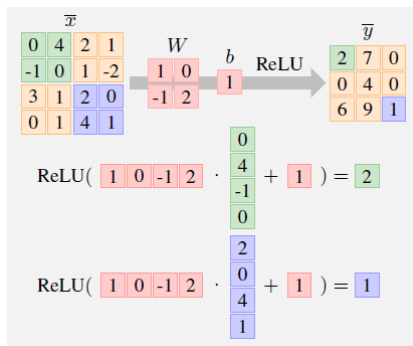


Figure: Convolutional layer $\text{Conv}_{(W,b)}^t$ (one filter)

Representing Neural Networks As Conditional Affine Transformations

Convolutional Layer to CAT

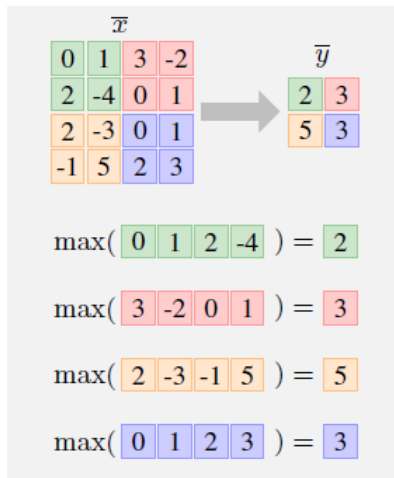
When \bar{x} is reshaped to a vector \bar{x}^v , the four entries $x_{1,1}, x_{1,2}, x_{2,1}$ and $x_{2,2}$ will be found in x_1^v, x_2^v, x_5^v and x_6^v , respectively.

Similarly, when \bar{y} is reshaped to \bar{y}^v , the entry $\bar{y}_{1,1}$ will be found in \bar{y}_1^v .

$$W^F = \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \quad \bar{b}^F = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Representing Neural Networks As Conditional Affine Transformations

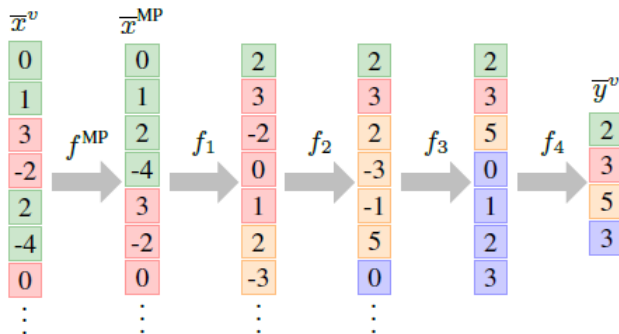
Max Pooling (MP) Layer



Representing Neural Networks As Conditional Affine Transformations

Max Pooling to CAT

They reorder \bar{x}^v using a permutation matrix W^{MP} , yielding \bar{x}^{MP} .



Max Pooling to CAT

[illegible]

Representing Neural Networks As Conditional Affine Transformations

Max Pooling to CAT

$$W^{(1,3)} = \begin{pmatrix} 001000000000000000 \\ 000010000000000000 \\ 000001000000000000 \\ 000000100000000000 \\ 000000010000000000 \\ 000000001000000000 \\ 000000000100000000 \\ 000000000010000000 \\ 000000000001000000 \\ 000000000000100000 \\ 000000000000010000 \\ 000000000000001000 \\ 000000000000000100 \\ 000000000000000010 \\ 000000000000000001 \end{pmatrix}$$

$$f_1(\bar{x}) = \begin{cases} \text{case } (x_1 \geq x_2) \wedge (x_1 \geq x_3) \wedge (x_1 \geq x_4): & W^{(1,1)} \cdot \bar{x}, \\ \text{case } (x_2 \geq x_1) \wedge (x_2 \geq x_3) \wedge (x_2 \geq x_4): & W^{(1,2)} \cdot \bar{x}, \\ \text{case } (x_3 \geq x_1) \wedge (x_3 \geq x_2) \wedge (x_3 \geq x_4): & W^{(1,3)} \cdot \bar{x}, \\ \text{case } (x_4 \geq x_1) \wedge (x_4 \geq x_2) \wedge (x_4 \geq x_3): & W^{(1,4)} \cdot \bar{x}. \end{cases}$$

$$\text{MaxPool}'_{2,2} = f_4 \circ f_3 \circ f_2 \circ f_1 \circ f^{\text{MP}}.$$

$$\text{MaxPool}'_{2,2}(\bar{x}^v) = W^{(4,7)} \cdot W^{(3,6)} \cdot W^{(2,2)} \cdot W^{(1,3)} \cdot W^{\text{MP}} \cdot \bar{x}^v.$$

Table of Contents

- 1 Introduction
- 2 Representing Neural Networks As Conditional Affine Transformations
- 3 Background**
- 4 AI^2 : AI For Neural Networks
- 5 Evaluation of AI^2

Background: Abstract Interpretation

Given a function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$, a set of inputs $X \in \mathbb{R}^m$, and a property $C \in \mathbb{R}^n$, the goal is to determine whether the property holds, that is, whether $\forall \bar{x} \in X, f(\bar{x}) \in C$.

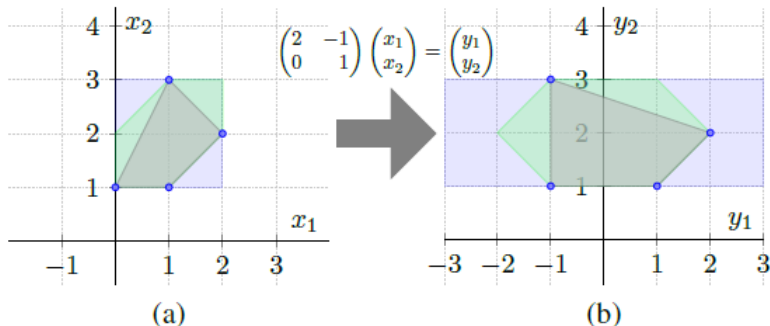


Figure: (a) Abstracting four points with a polyhedron (gray), zonotope (green), and box (blue). (b) The points and abstractions resulting from the affine transformer.

Background: Abstract Interpretation

To reason about all inputs simultaneously, they lift the definition of f to be over a set of inputs X rather than a single input:

$$T_f : P(\mathbb{R}^m) \rightarrow P(\mathbb{R}^n) \quad T_f(X) = \{f(\bar{x}) \mid \bar{x} \in X\}$$

The function T_f is called the *concrete transformer* of f .

Because the set X can be very large (or infinite), we cannot enumerate all points in X to compute $T_f(X)$.

Instead, AI overapproximates sets with abstract elements and then defines *an abstract transformer* of f , which works with these abstract elements and overapproximates the effect of T_f .

Background: Abstract Interpretation

Abstract Domains

Abstract domains consist of shapes expressible as a set of logical constraints: Box (i.e., Interval), Zonotope, Polyhedra.

Box domain consists of boxes, captured by a set of constraints of the form $a \leq x_i < b$, for $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ and $a < b$.

Note that B is not very precise since it includes 9 integer points (along with other points), whereas X has only 4 points.

The **Zonotope** domain consists of *zonotopes*. A zonotope is a center-symmetric convex closed polyhedron. Zonotope is a more precise domain than Box

The **Polyhedra** domain consists of convex closed *polyhedra*, where a polyhedron is captured by a set of linear constraints of the form $A.\bar{x} \leq b$,

Background: Abstract Interpretation

Abstract Transformers

To compute the effect of a function on an abstract element, AI uses the concept of a *abstract transformer*.

Given the (lifted) concrete transformer $T_f : P(\mathbb{R}^m) \rightarrow P(\mathbb{R}^n)$ of a function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$, an abstract transformer of T_f is a function over abstract domains, denoted by $T_f^\# : A^m \rightarrow A^n$.

To define *soundness*, they introduce two functions: the **abstraction** function and the **concretization** function.

Background: Abstract Interpretation

Abstract Transformers

An **abstraction function** $\alpha^m : P(\mathbb{R}^m) \rightarrow A^m$ maps a set of vectors to an abstract element in A^m that overapproximates it. For example, in the Box domain:

$$\alpha^2(\{(0, 1), (1, 1), (1, 3), (2, 2)\}) = \{0 \leq x_1 \leq 2, 1 \leq x_2 \leq 3\}.$$

A **concretization function** $\gamma^m : A^m \rightarrow P(\mathbb{R}^m)$ does the opposite: it maps an abstract element to the set of concrete vectors that it represents. For example, for Box:

$$\gamma^2(\{0 \leq x_1 \leq 2, 1 \leq x_2 \leq 3\}) = \{(0, 1), (0, 2), (0, 3), \\ (1, 1), (1, 2), (1, 3), \\ (2, 1), (2, 2), (2, 3), \dots\}.$$

Background: Abstract Interpretation

Abstract Transformers

An abstract transformer $T_f^\#$ is sound if for all $a \in A^m$, we have $T_f(\gamma^m(a)) \subseteq \gamma^n(T_f^\#(a))$, where T_f is the concrete transformer.

That is, an abstract transformer has to overapproximate the effect of a concrete transformer.

It is also important that abstract transformers are *precise*. That is, the abstract output should include as few points as possible.

Background: Abstract Interpretation

Property Verification

In general, an abstract output $a = T_f^\#(X)$ proves a property $T_f(X) \subseteq C$ if $\gamma^n(a) \subseteq C$.

If the abstract output proves a property, we know that the property holds for all possible concrete values.

However, the property may hold even if it cannot be proven with a given abstract domain. (precision)

To apply AI successfully, we need to:

- (a)** find a suitable abstract domain, and
- (b)** define abstract transformers that are sound and as precise as possible.

Table of Contents

- 1 Introduction
- 2 Representing Neural Networks As Conditional Affine Transformations
- 3 Background
- 4 AI^2 : AI For Neural Networks**
- 5 Evaluation of AI^2

AI²: AI For Neural Networks

Abstract Interpretation for CAT Functions

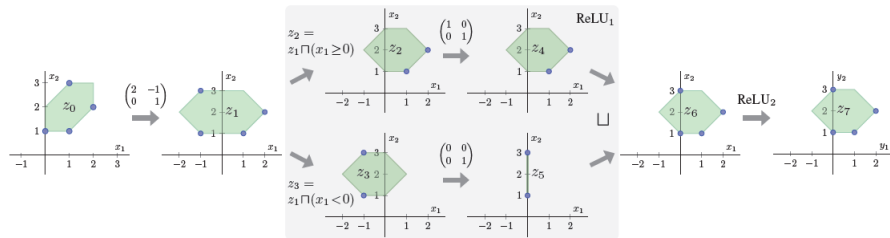


Figure: Illustration of how AI² overapproximates neural network states. Blue circles show the concrete values, while green zonotopes show the abstract elements. The gray box shows the steps in one application of the ReLU transformer.

AI²: AI For Neural Networks

Abstract Interpretation for CAT Functions

When a neural network $N = f'_l \circ \dots \circ f'_1$ is a composition of multiple CAT functions f'_i of the shape

$$f'_i(\bar{x}) = W.\bar{x} + \bar{b} \text{ or}$$

$$f'_i(\bar{x}) = \text{case } E_1 : f_1(\bar{x}), \dots, \text{case } E_k : f_k(\bar{x}),$$

we only have to define abstract transformers for these two kinds of functions.

We then obtain the abstract transformer $T_N^\# = T_{f'_l}^\# \circ \dots \circ T_{f'_1}^\#$

AI²: AI For Neural Networks

Abstract Interpretation for CAT Functions

To split and unify, assume two standard operators for abstract domains:

- (1) *meet* with a conjunction of linear constraints and
- (2) *join*.

Here, the meet operator overapproximates set intersection \cap to get a *sound*, but not perfectly *precise*, result.

Finally, $\gamma(z_7)$ is overapproximation of the network outputs for initial set of points.

The abstract element z_7 is a finite representation of this infinite set.

AI²: AI For Neural Networks

Abstract Interpretation for CAT Functions

For $f(\overline{x}) = W \cdot \overline{x} + \overline{b}$, $T_f^\#(a) = \text{Aff}(a, W, \overline{b})$.

For $f(\overline{x}) = \text{case } E_1: f_1(\overline{x}), \dots, \text{case } E_k: f_k(\overline{y})$,

$$T_f^\#(a) = \bigsqcup_{1 \leq i \leq k} f_i^\#(a \sqcap E_i).$$

For $f(\overline{x}) = f_2(f_1(\overline{x}))$, $T_f^\#(a) = T_{f_2}^\#(T_{f_1}^\#(a))$.

In summary, they define abstract transformers for every kind of CAT function. These definitions supports:

- (1) a meet operator between an abstract element and a conjunction of linear constraints,
- (2) a join operator between two abstract elements, and
- (3) an affine transformer.

AI²: AI For Neural Networks

Abstract Interpretation for CAT Functions

In this section, they explain how to leverage AI with our abstract transformers to prove properties of neural networks (focusing on robustness properties).

Local Robustness. This is a property (X, C_L) where X is a robustness region and C_L contains the outputs that describe the *same* label L :

$$C_L = \left\{ \bar{y} \in \mathbb{R}^n \mid \arg \max_{i \in \{1, \dots, n\}} (y_i) = L \right\}.$$

For example, previous figure shows a neural network and a robustness property (X, C_2) for $X = \{(0, 1), (1, 1), (1, 3), (2, 2)\}$ and $C_2 = \{\bar{y} \mid \arg \max(y_1, y_2) = 2\}$. In this example, (X, C_2) holds.

Table of Contents

- 1 Introduction
- 2 Representing Neural Networks As Conditional Affine Transformations
- 3 Background
- 4 AI²: AI For Neural Networks
- 5 Evaluation of AI²**

Evaluation of AI²

In this section, they present empirical evaluation of AI².

Three most important findings:

- AI² can prove useful robustness properties for convolutional networks with 53 000 neurons and large fully connected feedforward networks with 1 800 neurons.
- AI² benefits from more precise abstract domains: Zonotope enables AI² to prove substantially more properties over Box. Further, ZonotopeN, with $N \geq 2$, can prove stronger robustness properties than Zonotope alone.
- AI² scales better than the SMT-based Reluplex: AI² is able to verify robustness properties on large networks with ≥ 1200 neurons within few minutes, while Reluplex takes hours to verify the same properties.

Evaluation of AI²

They used two popular datasets: MNIST and CIFAR-10.

MNIST consists of 60 000 grayscale images of handwritten digits, whose resolution is 28×28 pixels.

CIFAR consists of 60 000 colored photographs with 3 color channels, whose resolution is 32×32 pixels.

The images are partitioned into 10 different classes (e.g., airplane or bird).

Evaluation of AI²

They trained convolutional and fully connected feedforward networks on both datasets.

The training completed when each network had a test set accuracy of at least 0.9.

For the convolutional networks, they used the LeNet architecture.

They used 7 different architectures of fully connected feedforward networks (FNNs).

Evaluation of AI^2

Robustness Properties

They consider local robustness properties (X, C_L) where the region X captures changes to lighting conditions. (brightness attack)

Formally robustness region is defined as:

$$S_{\bar{x}, \delta} = \{\bar{x}' \in \mathbb{R}^m \mid \forall i \in [1, m]. 1 - \delta \leq x_i \leq x'_i \leq 1 \vee x'_i = x_i\}.$$

For example, the robustness region for $x = (0.6, 0.85, 0.9)$ and bound $\delta = 0.2$ is given by the set: $\{(0.6, x, x') \in \mathbb{R}^3 \mid x \in [0.85, 1], x' \in [0.9, 1]\}$

They used AI^2 to check whether all inputs in a given region $S_{\bar{x}, \delta}$ are classified to the label assigned to \bar{x} .

Evaluation of AI^2

Robustness Properties

They consider 6 different robustness bounds δ , which are drawn from the set $\Delta = \{0.001, 0.005, 0.025, 0.045, 0.065, 0.085\}$.

Given robustness region $S_{x,\delta}$, each pixel of the image x is brightened independently of the other pixels.

They remark that this is useful to capture scenarios where only part of the image is brightened (e.g., due to shadowing).

Evaluation of AI²

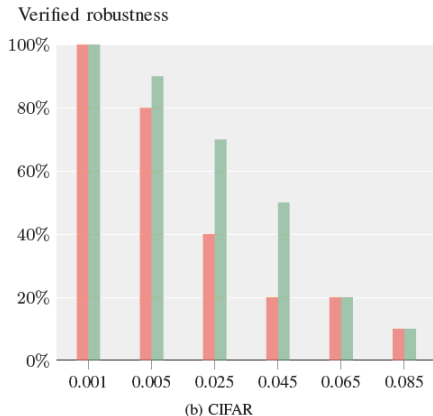
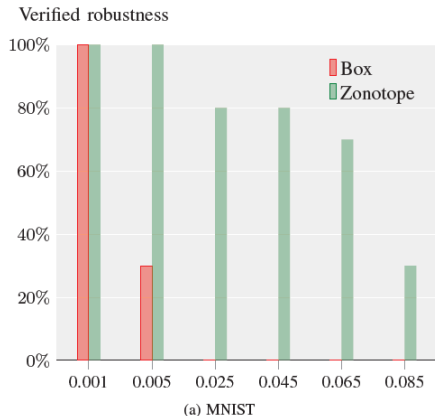
Benchmarks

They selected 10 images from each dataset.

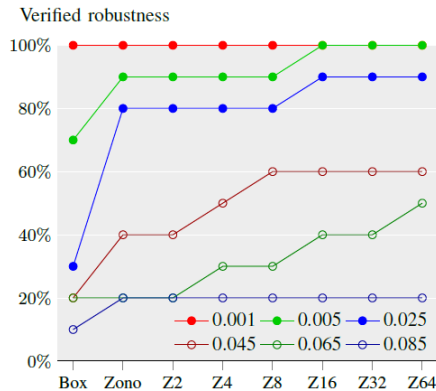
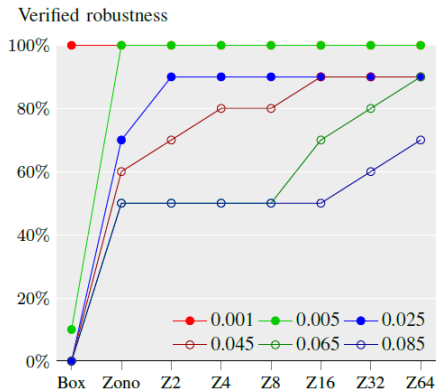
Then, they specified a robustness property for each image and each robustness bound in δ , resulting in 60 properties per dataset.

They ran AI² to check whether each neural network satisfies the robustness properties for the respective dataset.

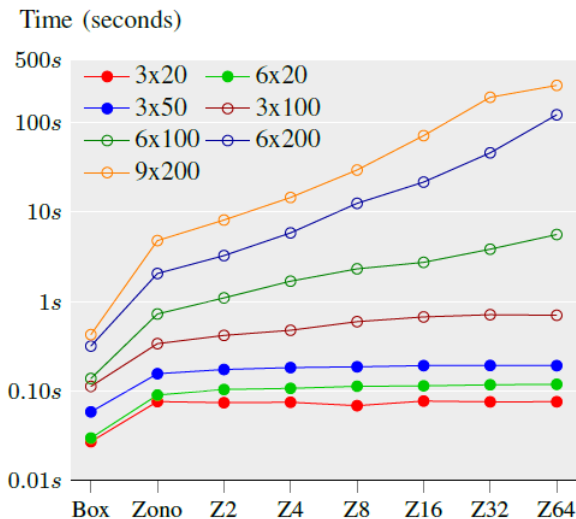
Evaluation of AI²



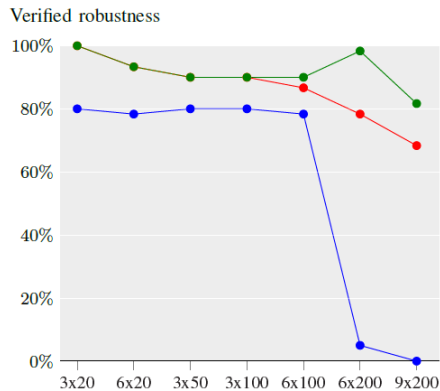
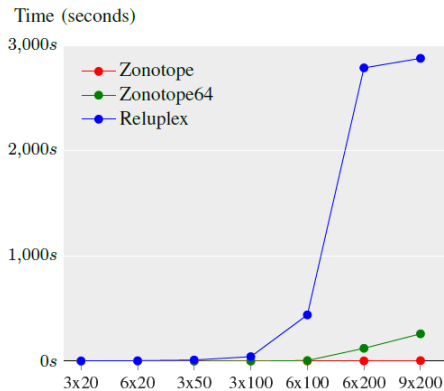
Evaluation of AI²



Evaluation of AI²



Evaluation of AI²



Evaluation of AI²

