AI^2: AI Safety and Robustness with Abstract Interpretation

Martin Vechev
DeepCode.ai and ETH Zurich

safeai.ethz.ch

How good (robust) is your neural net?

Neural networks are ***not*** robust to input perturbations
(e.g., image rotation / change of lighting)

DRV_C1: right   DRV_C2: right   DRV_C3: right

Misclassifications in neural networks deployed in self-driving cars [1]
In each picture one of the 3 networks makes a mistake...

[1] Pei et. al., DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP 2017

Wanted: **Automated** and **scalable** analysis to certify realistic neural nets

Potential Benefits:

- Certify large cyber-physical systems that use the NN
- Prove robustness of NN (beyond just finding adversarial examples)
- Learn interpretable specs of NN
- Compare NNs
- Train NNs

Talk based on

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

IEEE Oakland Security &Privacy, 2018

(Gehr, Mirman, Drachsler-Cohen, Tsankov, Chaudhuri, V)

Differentiable Abstract Interpretation for Provably Robust Neural Networks

ACM ICML 2018

(Mirman, Gehr, V)

Problem Statement and Challenges

Given

- a **neural network** $\boldsymbol{N}$
- a **property over inputs** $\boldsymbol{\varphi}$
- a **property over outputs** $\boldsymbol{\psi}$

check whether $\forall \boldsymbol{i} \in \boldsymbol{I}.\, \boldsymbol{i} \vDash \boldsymbol{\varphi} \Longrightarrow \boldsymbol{N}(\boldsymbol{i}) \vDash \boldsymbol{\psi}$ holds

Challenges:

- The property φ over inputs usually captures an **unbounded set of inputs**
- Existing symbolic solutions **do not scale** to large networks (e.g. conv nets)

To scale:

- Need to under- or over- approximate

High Level Insight: AI for AI

Deep Neural Nets:

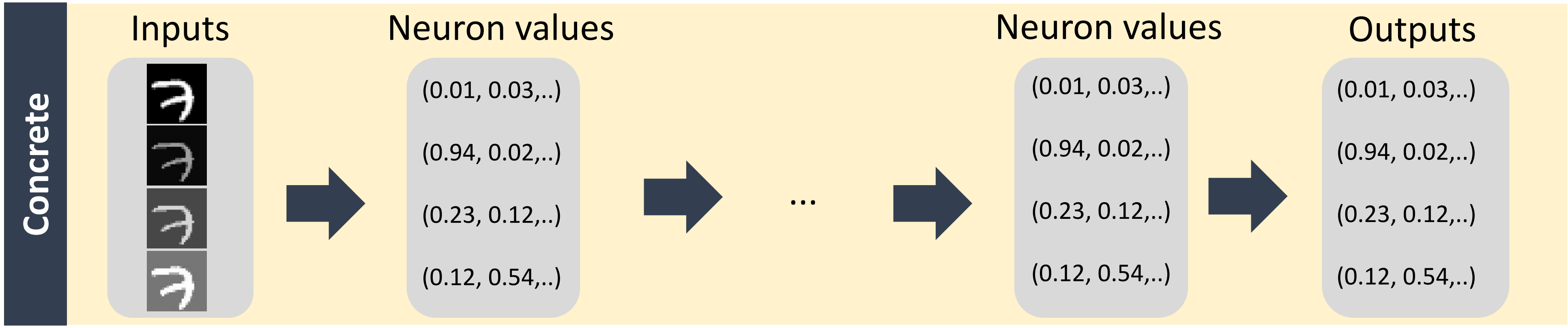
Affine transforms + Restricted non-linearity

+

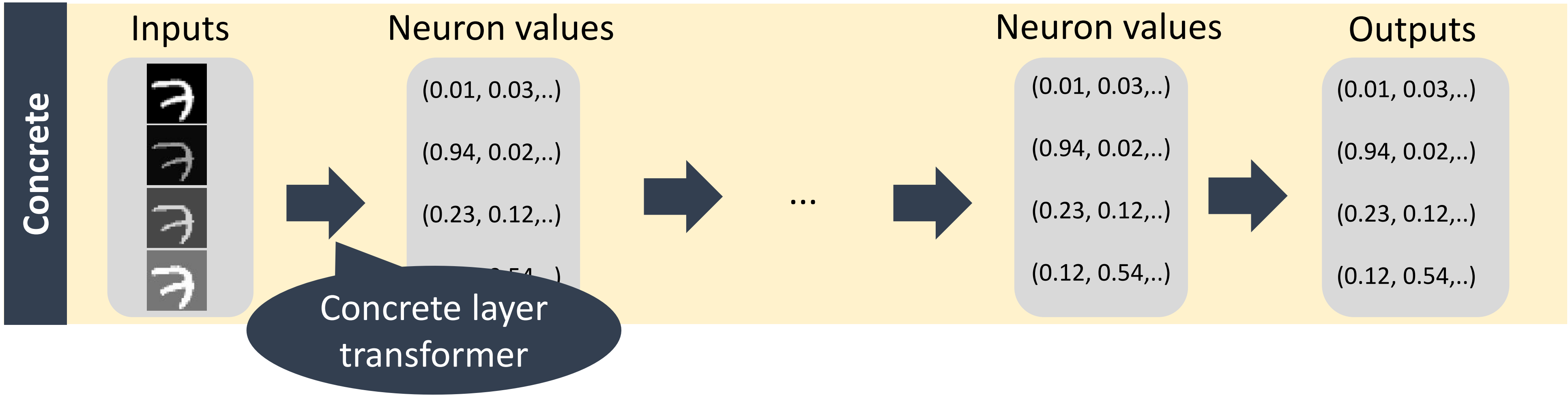
Abstract Interpretation:

Scalable and Precise Numerical Domains

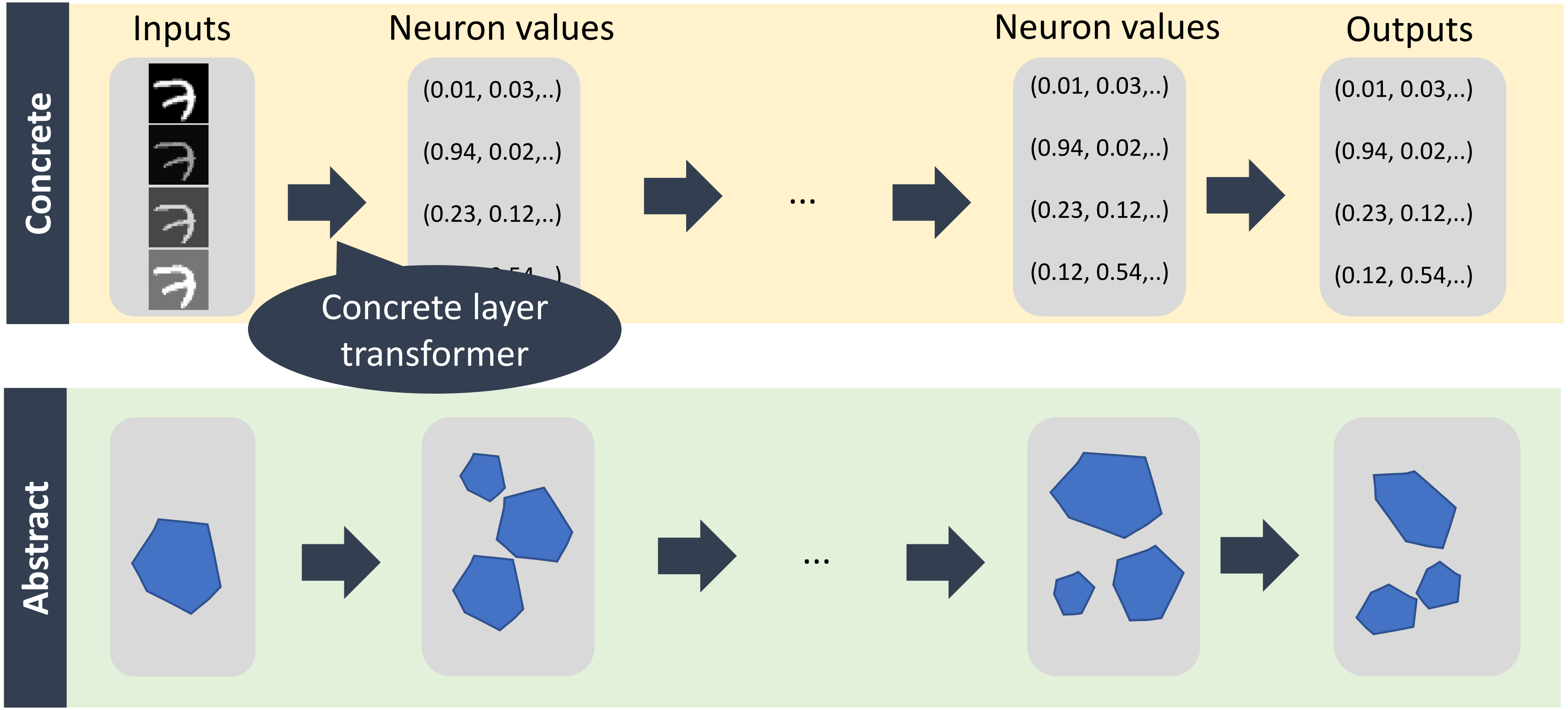
AI²: Abstract Interpretation for NNs



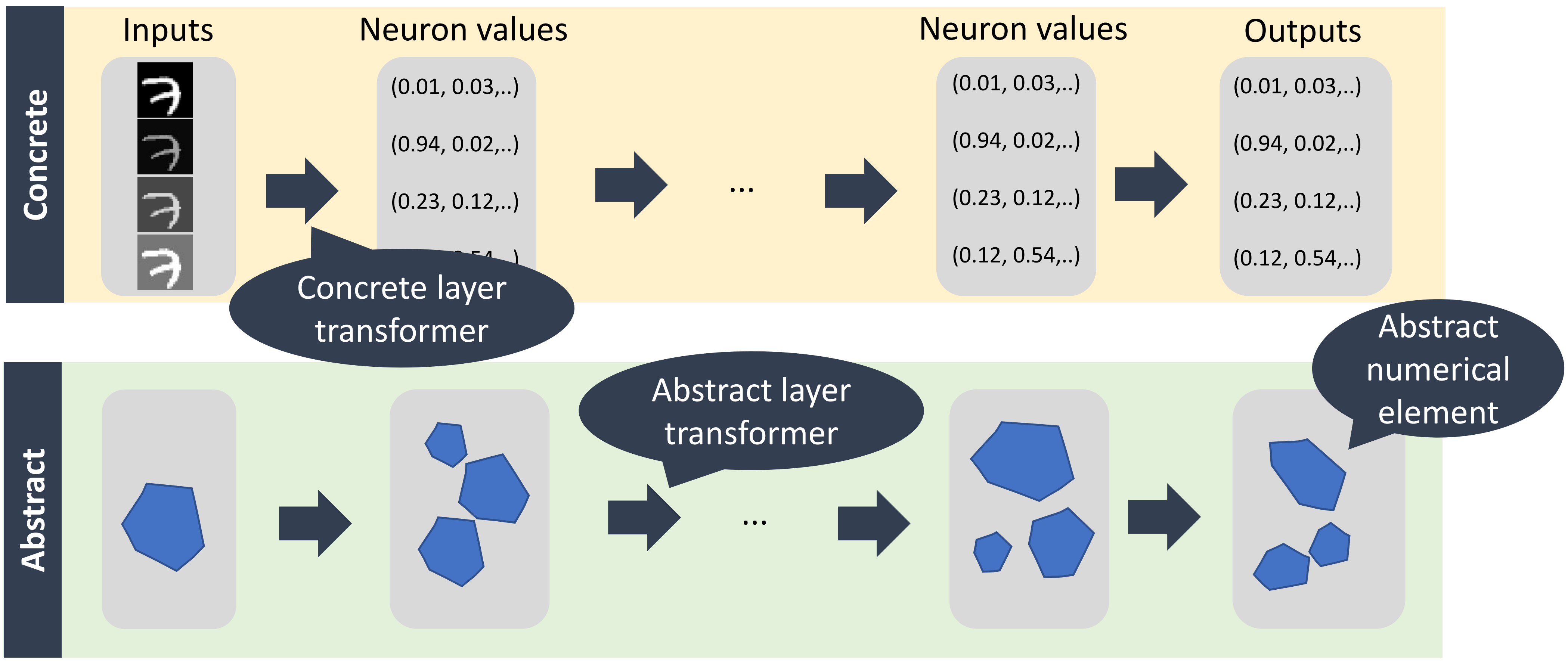
AI²: Abstract Interpretation for NNs



AI^2: Abstract Interpretation for NNs



AI^2: Abstract Interpretation for NNs



Zonotope Abstract Domain

Ghorbal, Goubault, Putot, CAV'09

Exact for linear operations

Each variable (here, abstract neuron) captured in an affine form

Allows relating variables (in limited ways) through parameters (unlike Box)

Zonotope Abstract Domain

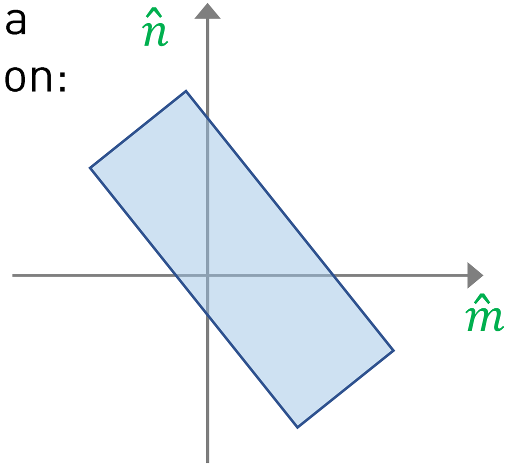
If we have two (concrete) neurons n and m,
then the abstract neurons will look like:

$$\hat{n} = a_0{}^n + \sum_{i=1}^{k} a_i{}^n \, \epsilon_i$$

$$\hat{m} = a_0{}^m + \sum_{i=1}^{k} a_i{}^m \, \epsilon_i$$

The **meaning** (γ) is a polytope centered around $a_0{}^n$ and $a_0{}^m$

Example of a concretization:



Zonotope Abstract Domain

For two (concrete) neurons n and m, the abstract neurons will look like:

$$\hat{n} = {a_0}^n + \sum_{i=1}^{k} {a_i}^n \epsilon_i$$

$$\hat{m} = {a_0}^m + \sum_{i=1}^{k} {a_i}^m \epsilon_i$$

ϵ_i : noise terms ranging [-1,1] shared between abstract neurons

${a_i}^n$: real number that controls magnitude of noise

Closed under affine transforms, e.g., $\hat{n} + \hat{m}$

Not closed under joins and meets, e.g.,: $\hat{n} \sqcup \hat{m}$, $\hat{n} \gtrapprox \hat{m}$

The **meaning** (γ) is a polytope centered around ${a_0}^n$ and ${a_0}^m$

Meaning of a zonotope

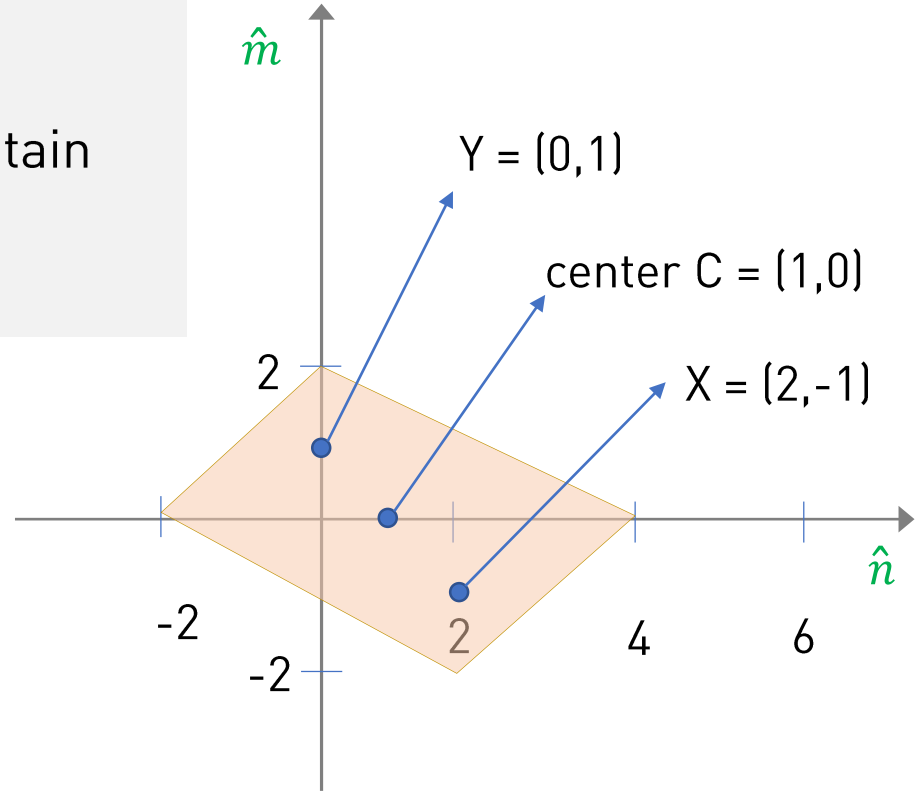
Centered means there is a center point C, where from any point X in the polytope, we can obtain a flipped point Y of X, where Y = 2C-X, and Y is in the polytope and X and Y are equal distance from C.

For instance, ψ below is centered around C = (1,0).

For example, a point X = (2,-1) can be flipped to obtain a point Y = 2C – X = (0,1)

$$\psi = \begin{array}{l} \hat{n} = 1 - 2\epsilon_1 + \epsilon_2 \\ \hat{m} = 0 + \epsilon_1 + \epsilon_2 \end{array}$$

$\gamma\,(\psi)$ is:



Zonotope Operations for Neural Networks

Multiplication by a constant real-valued constant C :

$$\left(a_0^{\ n} + \sum_{i=1}^{k} a_i^{\ n}\, \epsilon_i\right) \ * \ C = \left(C * a_0^{\ n} + \sum_{i=1}^{k} C * a_i^{\ n}\, \epsilon_i\right)$$

Adding two variables is done component-wise (abstract transformer is exact) :

$$\left(a_0^{\ n} + \sum_{i=1}^{k} a_i^{\ n}\, \epsilon_i\right) + \left(a_0^{\ m} + \sum_{i=1}^{k} a_i^{\ m}\, \epsilon_i\right) = (a_0^{\ n} + a_0^{\ m}) + \sum_{i=1}^{k} (a_i^{\ n} + a_i^{\ m}) * \epsilon_i$$

*No need for multiplication of zonotopes

Zonotope Operations for Neural Networks: join $\sqcup$

ψ_1 =
$$\begin{aligned}\hat{n} &= 3 + \epsilon_1 + 2\,\epsilon_2 \\ \hat{m} &= 0 + \epsilon_1 + \epsilon_2\end{aligned}$$

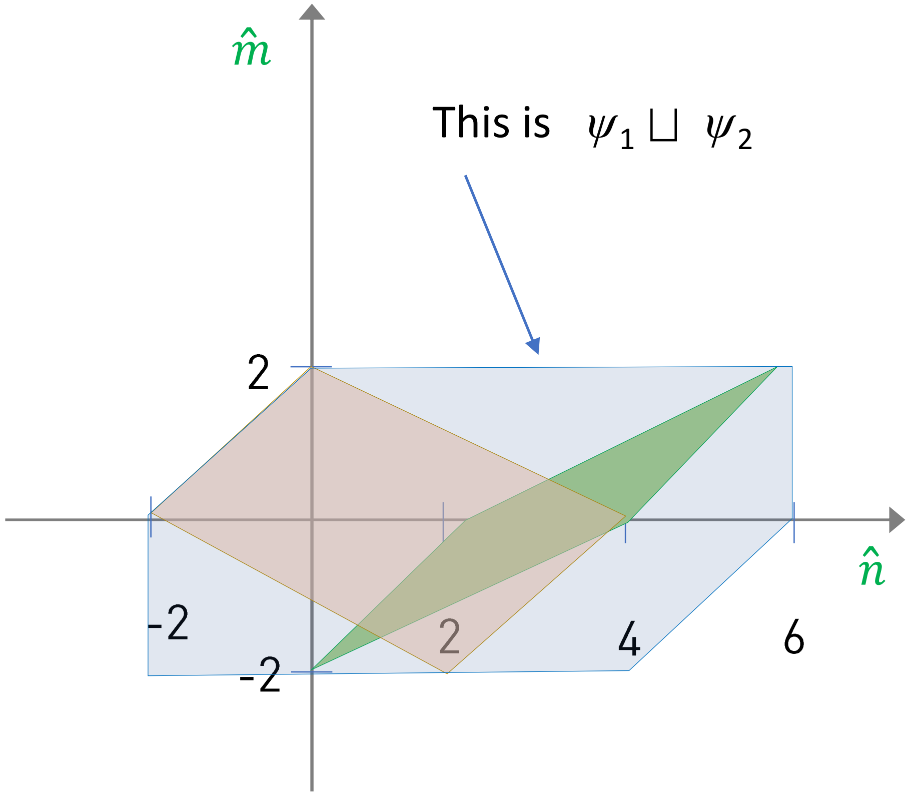
$$\sqcup$$

ψ_2 =
$$\begin{aligned}\hat{n} &= 1 - 2\epsilon_1 + \epsilon_2 \\ \hat{m} &= 0 + \epsilon_1 + \epsilon_2\end{aligned}$$

$$=$$

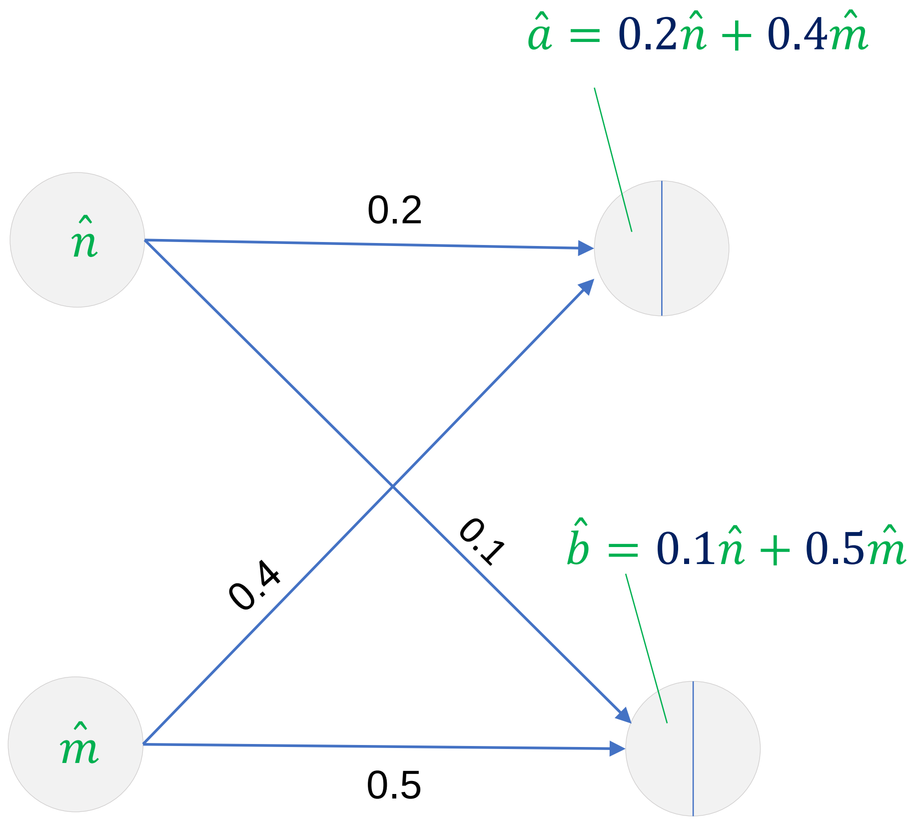
$$\begin{aligned}\hat{n} &= 2 + \epsilon_2 + 3\epsilon_u \\ \hat{m} &= 0 + \epsilon_1 + \epsilon_2\end{aligned}$$

New error term is introduced



Lets see how to apply the operations to analyze networks on a simple 2 layer feed-forward network

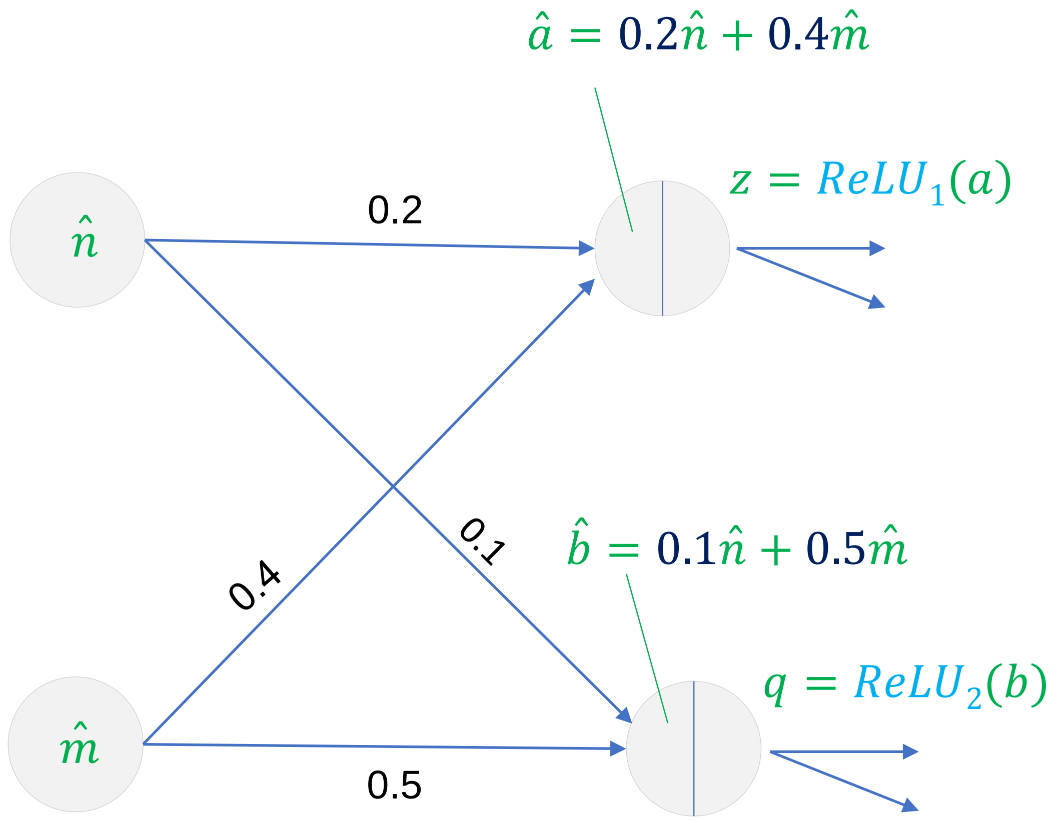
Analysis of a single layer



Step I: compute effect of affine transform:

Affine $\equiv$ $\hat{a} = 0.2\hat{n} + 0.4\hat{m} \quad \wedge \quad \hat{b} = 0.1\hat{n} + 0.5\hat{m}$

Analysis of a single layer

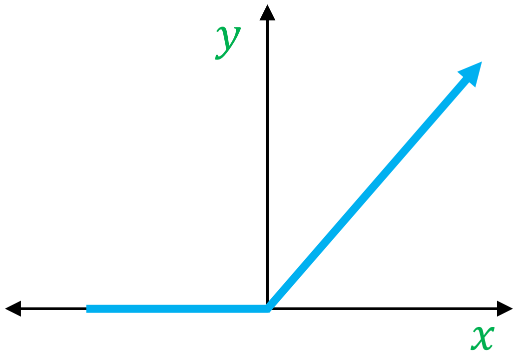


Step I: compute effect of affine transform:

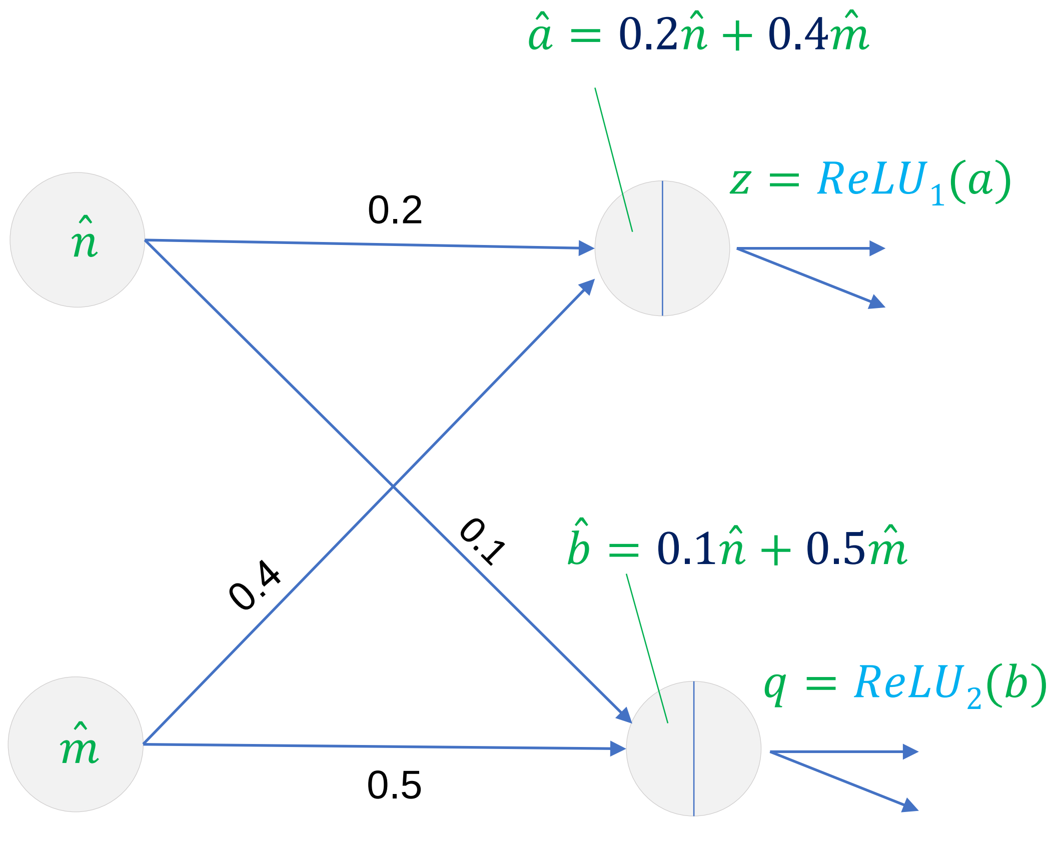
Affine $\equiv$ $\hat{a} = 0.2\hat{n} + 0.4\hat{m} \quad \wedge \quad \hat{b} = 0.1\hat{n} + 0.5\hat{m}$

Step II: compute effect of $ReLU$:

Activation function: $y = ReLU(x) = \max(0, x)$



Analysis of a single layer



Step I: compute effect of affine transform:

$$\text{Affine} \equiv \hat{a} = 0.2\hat{n} + 0.4\hat{m} \quad \wedge \quad \hat{b} = 0.1\hat{n} + 0.5\hat{m}$$

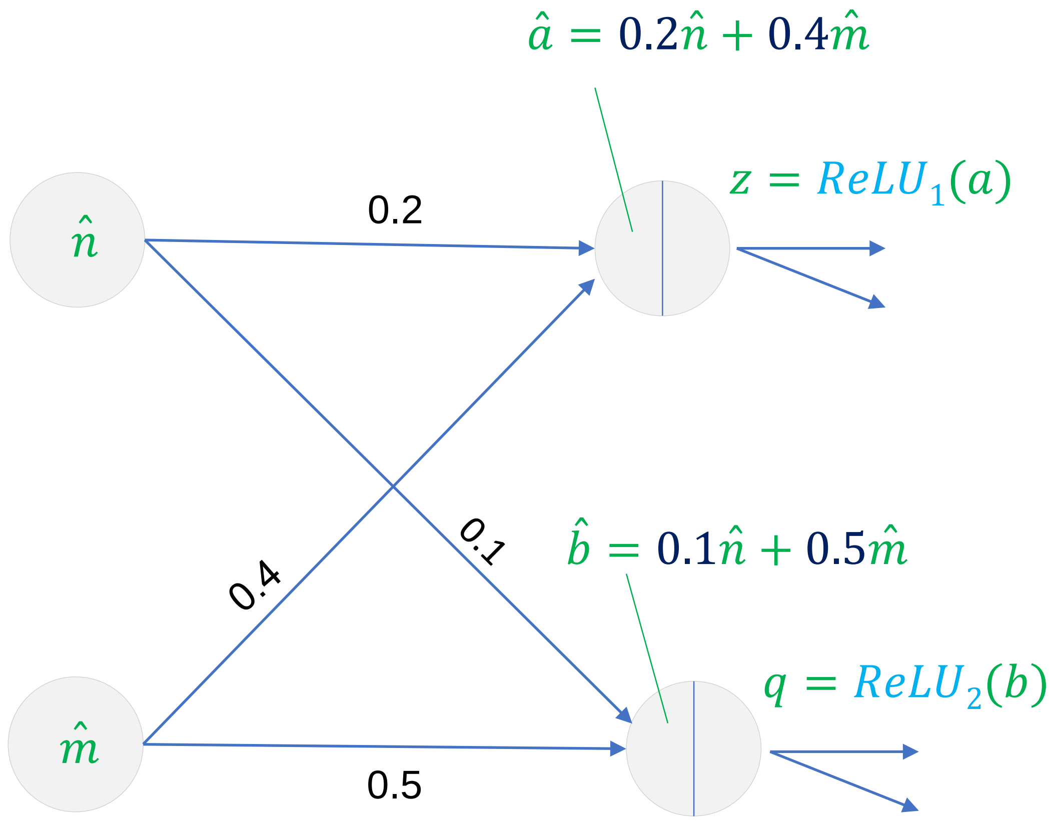
Step II: compute effect of *ReLU* :

$$f^{\#}_{ReLU} = ReLU_2^{\#}(b) \circ ReLU_1^{\#}(a)\ (\text{Affine})$$

$$ReLU_i^{\#}(x_i)(\psi) = (\psi \sqcap \{x_i \geq 0\}) \sqcup \psi_0$$

$$\psi_0 = \begin{cases} [\![x_i = 0]\!]\,(\psi) & \text{if } (\psi \sqcap \{x_i < 0\}) \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

Analysis of a single layer



Step I: compute effect of affine transform:

$$\text{Affine} \equiv \hat{a} = 0.2\hat{n} + 0.4\hat{m} \quad \wedge \quad \hat{b} = 0.1\hat{n} + 0.5\hat{m}$$

Step II: compute effect of $ReLU$:

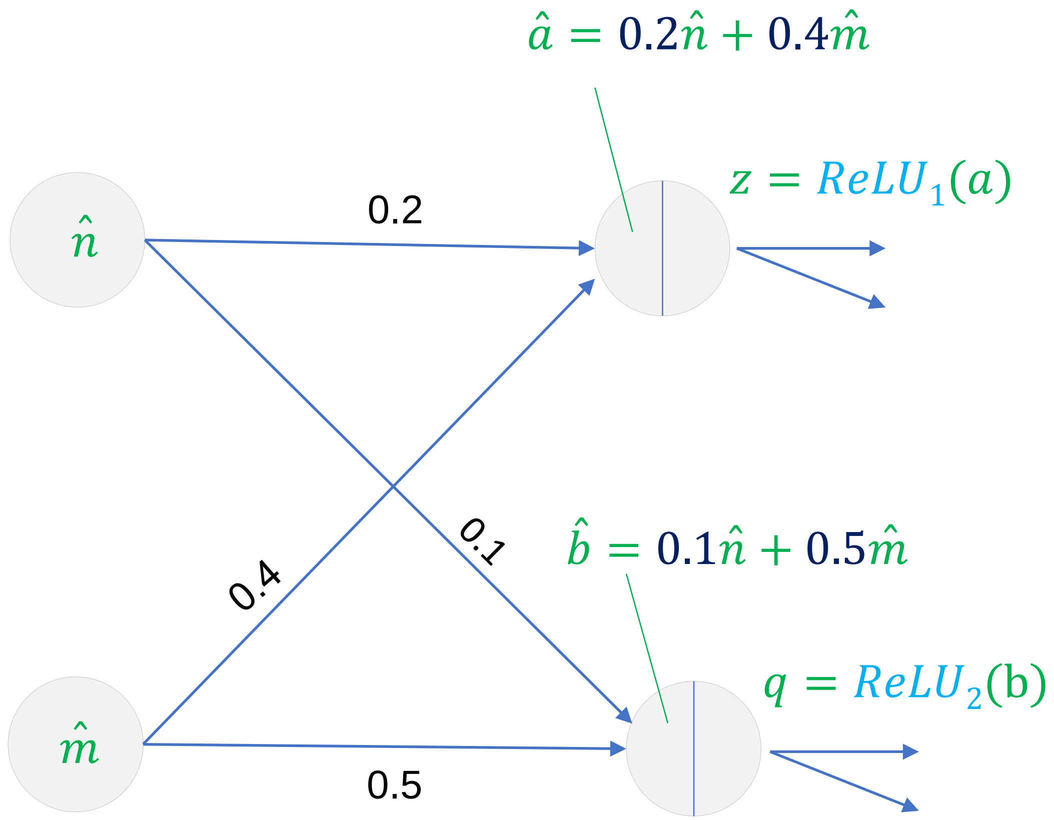
$$f^{\#}_{ReLU} = ReLU^{\#}_2(b) \circ ReLU^{\#}_1(a)\,(\text{Affine})$$

$$ReLU^{\#}_i(x_i)(\psi) = (\psi \sqcap \{x_i \geq 0\}) \sqcup \psi_0$$

$$\psi_0 = \begin{cases} [\![x_i = 0]\!]\,(\psi) & \text{if } (\psi \sqcap \{x_i < 0\}) \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

Problematic: too fine-grained, loses too much precision

Step II: Instead, design custom ReLU transformer



Optimal, precise, scales, can run on GPU:

$$\hat{z} = ReLU_1^{\#}(\hat{a})$$

$$\hat{q} = ReLU_2^{\#}(\hat{b})$$

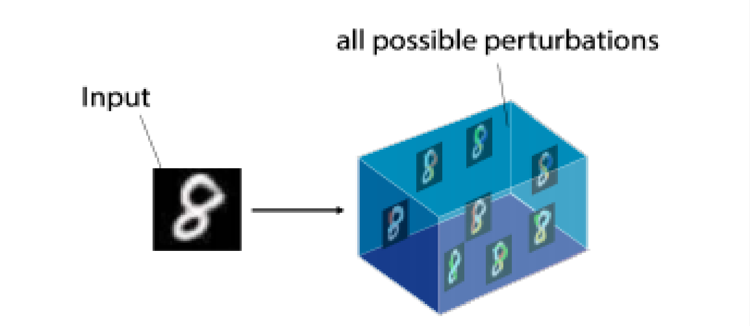
In our follow up works to AI^2 we have designed custom transformers, making it currently the most scalable system for analysis of deep learning

Use case of AI^2 : prove absence of adversarial attacks

Much recent work attacks: Goodfellow et al. (2014); Madry et al. (2018); Evtimov et al., (2017); Athalye & Sutskever (2017); Papernot et al. (2017); Xiao et al. (2018); Carlini & Wagner (2017);

Step 1: Define adversarial region around x based on the perturbation of interest (brightness, L_∞ , rotations, etc). For example:

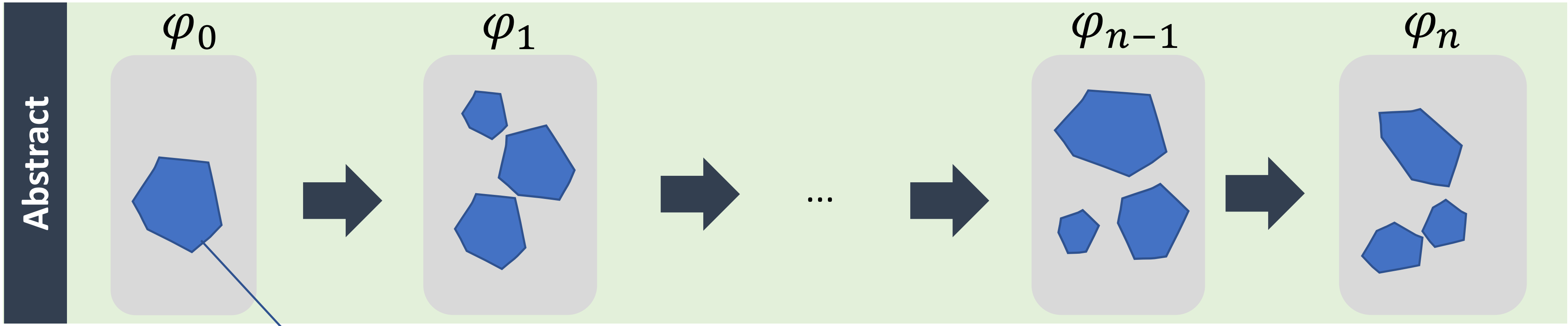
L_∞ ball: $\text{Ball}_\varepsilon(x) = \{y \mid \|x - y\|_\infty < \varepsilon\}$



Step 2: Attack tries to find image y in region where NN(x) $\neq$ NN(y)

Our goal: prove Step 2 never succeeds

Prove absence of brightness attack



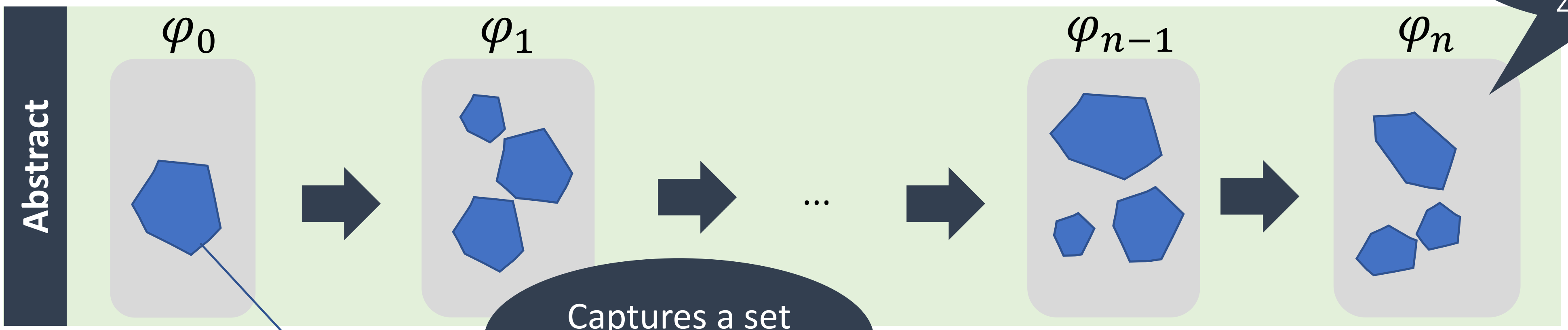
Adversarial region φ_0

$x_0 = 0$
$x_1 = 0.975 + 0.025\epsilon_1$
$x_2 = 0.125$
...
$x_{784} = 0.938 + 0.062\epsilon_{784}$
$\forall i. \epsilon_i \in [0,1]$

some pixels range over an interval now, but not all

Prove absence of brightness attack

bounded powerset of zonotopes



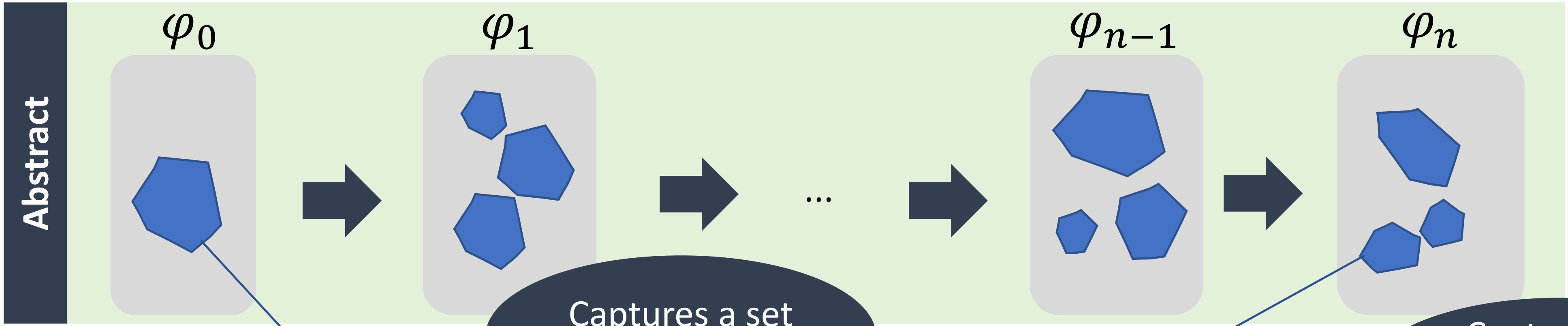
Captures a set of images

Adversarial region φ_0

$x_0 = 0$
$x_1 = 0.975 + 0.025\epsilon_1$
$x_2 = 0.125$
…
$x_{784} = 0.938 + 0.062\epsilon_{784}$
$\forall i. \epsilon_i \in [0,1]$

some pixels range over an interval now, but not all

Prove absence of brightness attack



Captures a set of images

Captures all possible output vectors

Adversarial region φ_0

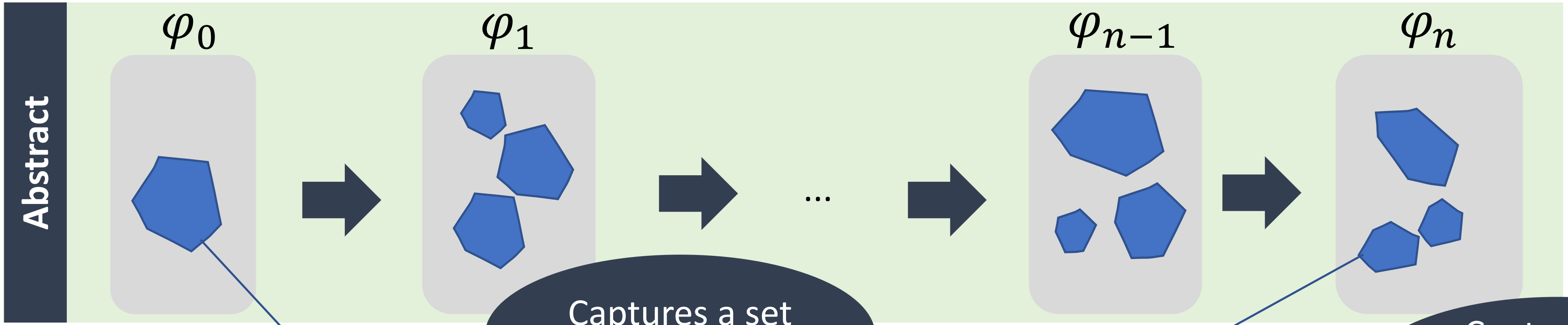
$x_0 = 0$
$x_1 = 0.975 + 0.025\epsilon_1$
$x_2 = 0.125$
…
$x_{784} = 0.938 + 0.062\epsilon_{784}$
$\forall i.\, \epsilon_i \in [0,1]$

Output constraint φ_n

$x_0 = 0$
$x_1 = 2.60 + 0.015\epsilon_0 + 0.023\epsilon_1 + 5.181\epsilon_2 + \cdots$
$x_2 = 4.63 - 0.005\epsilon_0 - 0.006\epsilon_1 + 0.023\epsilon_2 + \cdots$
…
$x_9 = 0.12 - 0.125\epsilon_0 + 0.102\epsilon_1 + 3.012\epsilon_2 + \cdots$
$\forall i.\, \epsilon_i \in [0,1]$

some pixels range over an interval now, but not all

Prove absence of brightness attack



Captures a set of images

Captures all possible output vectors

Adversarial region φ_0

$x_0 = 0$
$x_1 = 0.975 + 0.025\epsilon_1$
$x_2 = 0.125$
…
$x_{784} = 0.938 + 0.062\epsilon_{784}$
$\forall i.\, \epsilon_i \in [0,1]$

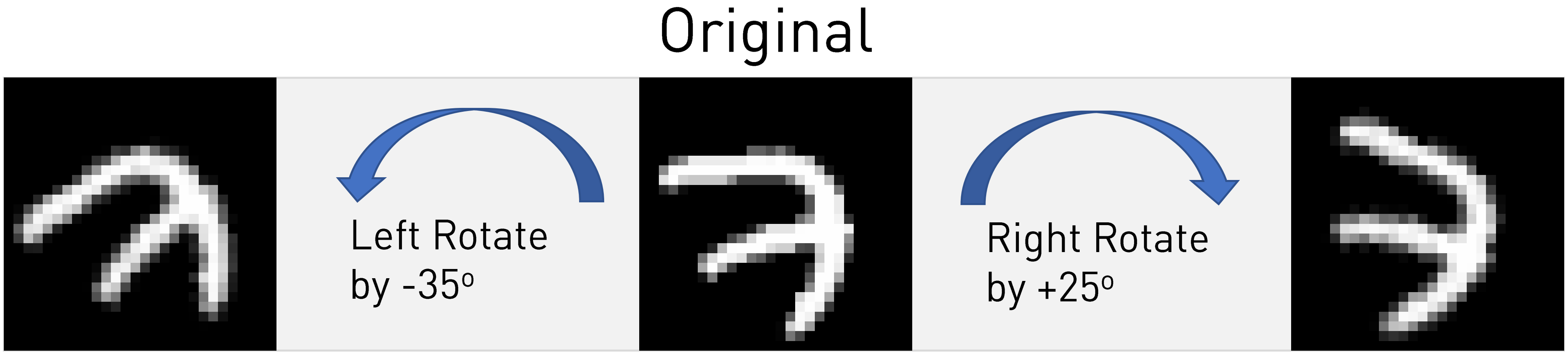
Output constraint φ_n

$x_0 = 0$
$x_1 = 2.60 + 0.015\epsilon_0 + 0.023\epsilon_1 + 5.181\epsilon_2 + \cdots$
$x_2 = 4.63 - 0.005\epsilon_0 - 0.006\epsilon_1 + 0.023\epsilon_2 + \cdots$
…
$x_9 = 0.12 - 0.125\epsilon_0 + 0.102\epsilon_1 + 3.012\epsilon_2 + \cdots$
$\forall i.\, \epsilon_i \in [0,1]$

some pixels range over an interval now, but not all

Label i is possible iff: $\varphi_n \sqcap \{\forall j.\, x_i \geq x_j\} \neq \bot$

More complex perturbation: rotations



First time we are able to prove rotations: We can prove network classifies any image in this adversarial region to 7

L_∞ and brightness adversarial regions can be exactly captured by boxes, but boxes cannot capture rotations exactly. To verify: use refinement.

Analysis can benefit training of networks

Idea: define abstract loss to include AI, apply automatic differentiation on AI

Training Method	Accuracy %	Attack Success %	Certified %
Baseline	98.4	2.4	2.8
Madry et al.	98.8	1.6	11.2
DiffAI (our method)	99.0	2.8	96.4

Convolutional Network with 124,000 neurons, L_∞ with $\varepsilon = 0.1$

Differentiable Abstract Interpretation for Provably Robust Neural Networks, ICML 2018
Matthew Mirman, Timon Gehr, M.V.

Differentiable AI training scales better than all prior work

System		Model	#Neurons	#Weights	Train 1 Epoch (s)
DiffAI	(Mirman, Gehr, V ICML 2018)	ConvSuper	$\sim$124k	$\sim$16mill	74
		Resnet18	$\sim$500k	$\sim$15mill	93
		ConvHuge	$\sim$500k	$\sim$65mill	142
Wong et al. (2018)		Large	$\sim$62k	$\sim$2.5mill	466
		Resnet	$\sim$107k	$\sim$4.2mill	1685
Wong & Kolter (2018)		MNIST Conv	$\sim$4k	$\sim$10k	180
Raghunathan et al. (2018)		MNIST 2 layer FFNN	$\sim$1k	$\sim$650k	-
Dvijotham et al. (2018)		Convnets	$\sim$21k	$\sim$650k	-

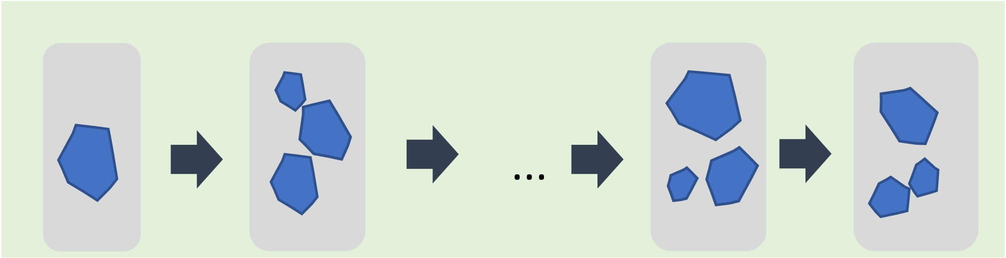
- Numbers as reported by prior work and not rerun on our hardware
- When hidden unit numbers and weight numbers were included, they were approximated using the network specifications in the paper with over-approximations where the specifications were not complete as in Dvijotham et al. (2018); Raghunathan et al. (2018)

Summary

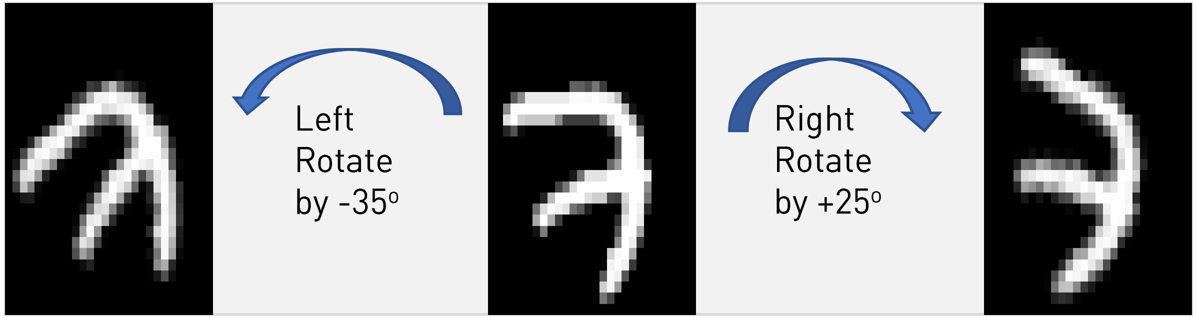
Certification of neural nets is important

DRV_C1: right DRV_C2: right DRV_C3: right

Key idea: approximate nets via AI



The most scalable analyzer for neural nets



Applications: training, explaining

Training Method	Accuracy %	Attack Success %	Certified %
Baseline	98.4	2.4	2.8
Madry et al.	98.8	1.6	11.2
DiffAi (our)	99.0	2.8	96.4

More at: safeai.ethz.ch