SVFNN: Safety and Robustness Certification of Neural Networks with SVF

Kaijie Liu

July 31, 2024

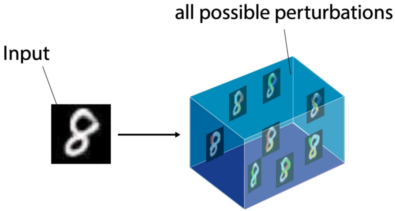
The University of New South Wales

Adversarial Attack

A small global perturbation can trick a well-trained model

Example of FGSM attack produced by Goodfellow et al. (2014)

Attack Methods



$$\textit{Ball}_{\epsilon}(\text{input}) = \{\text{attack} \mid \|\text{input} - \text{attack}\|_{\infty} \leq \epsilon\}$$

Problem Statement

Given

- - a neural network N
- - a property over inputs φ
- - a property over outputs ψ

check whether $\forall i \in I.\, i \models \varphi \implies N(i) \models \psi$ 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

PRIOR WORK

Satisfiability Modulo Theories (SMT)

Description: Converts robustness problem into logical constraints.

- Katz, G. et al. (2017). ”Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks”.

Mixed-Integer Linear Programming (MILP)

Description: Converts robustness problem into an MILP problem.

- Tjeng, V. et al. (2017). ”Verifying Neural Networks with Mixed Integer Programming”.

Abstract Interpretation

Description: Uses abstract domains to approximate output ranges.

- Gehr, T. et al. (2018). ”AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation”.

Abstract Interpretation

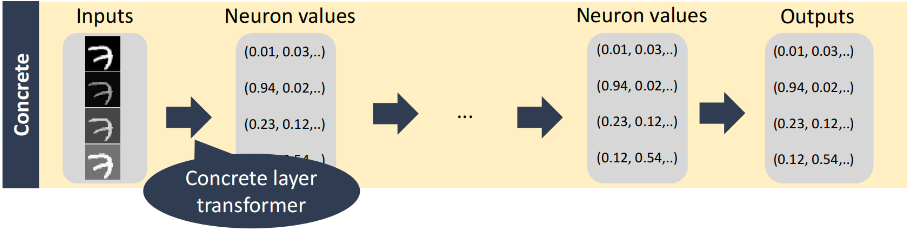
Deep Neural Nets:
Affine transforms + Restricted non-linearity

- **Affine Transforms:** Linear transformations that maintain the points, straight lines, and planes.
- **Restricted Non-linearity:** Activation functions like ReLU, Sigmoid, Tanh, which introduce non-linearity while maintaining computational efficiency.

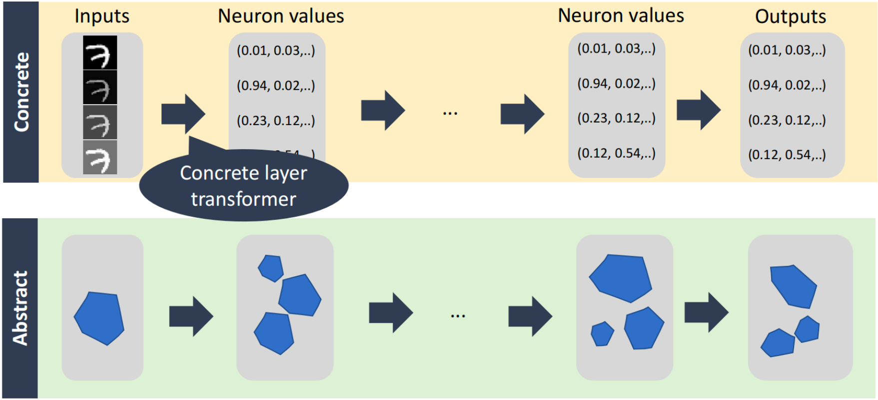
Abstract Interpretation:
Scalable and Precise Numerical Domains

- **Scalable Numerical Domains:** Techniques that scale with the size of the input and can handle large datasets.
- **Precise Analysis:** Ensures accurate modeling of the numerical properties of the network, enhancing robustness.

Concrete Domain



Abstract Domain



Abstract Interpretation

- Theory for approximating program behaviors.
- Sound, but (usually) incomplete.

Abstract Domain A

- Abstract element $a \in A$ represents set $\gamma(a) \subseteq \mathbb{R}^n$.

Abstract Transformers

- Abstract transformer $T_f^{\#}(a) = a'$ where $f(\gamma(a)) \subseteq \gamma(a')$.

Standard Abstract Domain Operations

- **Meet operator** ($\sqcap$): $\gamma(a \sqcap c) \supseteq \{x \in \gamma(a) \mid c(x)\}$.
- **Join operator** ($\sqcup$): $\gamma(a \sqcup b) \supseteq \gamma(a) \cup \gamma(b)$.
- **Affine transformer** $T_f^{\#}(a)$ for $f(x) = A \cdot x + b$.

Many abstract domains developed over the last 40 years.
Basis for much of static program analysis.

ReLU:

$$\mathrm{ReLU}(\bar{x}) = (\mathrm{ReLU}(x_1), \ldots, \mathrm{ReLU}(x_n))$$

$$\mathrm{ReLU} = \mathrm{ReLU}_n \circ \mathrm{ReLU}_{n-1} \circ \ldots \circ \mathrm{ReLU}_1$$

$$\mathrm{ReLU}_i(\bar{x}) = \begin{cases} x_i & \text{if } x_i \geq 0 \\ I_{i,t=0} \cdot \bar{x} & \text{if } x_i < 0 \end{cases}$$

Fully connected layer: $FC_{W,\bar{b}}(\bar{x}) = \mathrm{ReLU}(W \cdot \bar{x} + \bar{b})$

Convolutional layer:

$$F^{p,q} = (F_1^{p,q}, \ldots, F_t^{p,q})$$

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

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

$$FC_{W^F,\bar{b}^F}(\bar{x}) = \mathrm{ReLU}(W^F \cdot \bar{x}^v + \bar{b}^F)$$

SVFNN - Affine Function

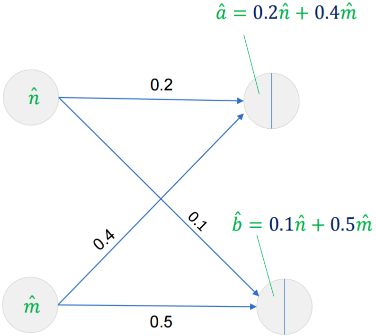
MaxPool:

$$\text{MaxPooling}_{p,q} : \mathbb{R}^{m \times n \times r} \to \mathbb{R}^{\frac{m}{p} \times \frac{n}{q} \times r}$$

$$y_{i,j,k} = \max\left(\{x_{i,j,k} \mid p \cdot (i-1) < i' \leq p \cdot i,\ q \cdot (j-1) < j' \leq q \cdot j\}\right)$$

$$\text{MaxPool}'_{p,q} : \mathbb{R}^{m \cdot n \cdot r} \to \mathbb{R}^{\frac{m}{p} \cdot \frac{n}{q} \cdot r}$$

$$\text{MaxPool}'_{p,q} = f_{\frac{m}{p} \cdot \frac{n}{q} \cdot r} \circ \ldots \circ f_1 \circ f^{MP}$$

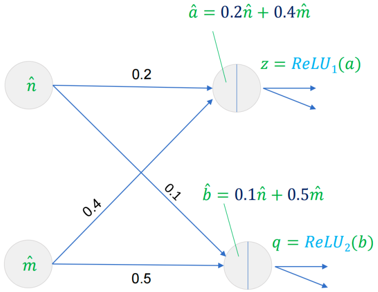


Step I: *compute affine transform:*

Affine $\equiv$

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

SVFNN - Analyse a Simple Neural Network



Step I: *compute affine transform:*

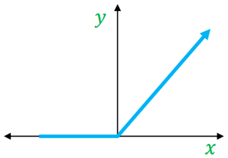
Affine $\equiv$

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

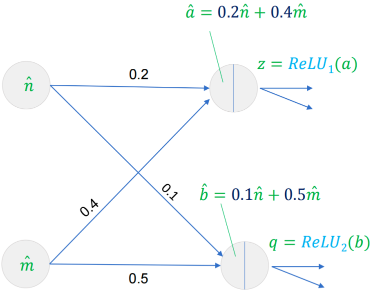
Step II: *compute effect of ReLU:*

Activation function: $y =$

ReLU(x) = max$(0, x)$



SVFNN - Analyse a Simple Neural Network



Step I: *compute affine transform:*

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

Step II: *compute effect of ReLU:*

$$f^{\#}_{\text{ReLU}} = \text{ReLU}^{\#}_{2}(b) \circ \text{ReLU}^{\#}_{1}(a) \quad (\text{Affine})$$

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

$$\psi_0 = \begin{cases} x_i = 0(\psi) & \text{if } (\psi \sqcap \{x_i < 0\}) \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

SVFNN

How to use SVFNN to analyse Neural Network?

SVFNN

How to use SVFNN to analyse Neural Network?

I will show you.