

Introduction to Neural Network Verification by Abstract Interpretation

LATTICE THEORY FOR PARALLEL PROGRAMMING

Yi-Nung Tsao

yi-nung.tsao@uni.lu

26th Nov 2025

University of Luxembourg



Neural Networks

Some of neural network applications - image generator

Real or Fake?



<https://britannicaeducation.com/blog/quiz-real-or-ai/>

Some of neural network applications - image generator

Real or Fake?



<https://britannicaeducation.com/blog/quiz-real-or-ai/>

Some of neural network applications - image generator

Real or Fake?



<https://britannicaeducation.com/blog/quiz-real-or-ai/>

Some of neural network applications - text generator

c How many countries start with the letter v?

There are no countries that start with the letter "V".

c Are you sure about that?

Yes, I'm sure. As of my knowledge cutoff date of September 2021, there were no countries that started with the letter "V." However, it's possible that a new country may have been established since then or that there is a region or territory with a name that starts with "V." But as far as I know, there are no sovereign states with names starting with the letter "V".

https://www.reddit.com/r/ChatGPT/comments/11709xq/chat_gpt_giving_wrong_answers_to_an_easy_question/

Some of neural network applications - self-driving car



<https://www.tesla.com/AI>

The limitation of neural network - robustness issue

Why is the robustness issue important?



Stop

Max Speed 100

This is a **safety and critical issue**.

<https://branditprintit.com.ng/five-important-traffic-signage-types-and-their-meaning-in-nigeria/>

Robustness issue in natural language process

Task: Sentiment Analysis. **Classifier:** CNN. **Original label:** 99.8% Negative. **Adversarial label:** 81.0% Positive.

Text: I love these awful awf ul 80's summer camp movies. The best part about "Party Camp" is the fact that it #literally literally has ne No plot. The cliches clichs here are limitless: the nerds vs. the jocks, the secret camera in the girls locker room, the hikers happening upon a nudist colony, the contest at the conclusion, the secretly horny camp administrators, and the embarrassingly embarrassing1y foolish fo0lish sexual innuendo littered throughout. This movie will make you laugh, but never intentionally. I repeat, never.

Task: Sentiment Analysis. **Classifier:** Amazon AWS. **Original label:** 100% Negative. **Adversarial label:** 89% Positive.

Text: I watched this movie recently mainly because I am a Huge fan of Jodie Foster's. I saw this movie was made right between her 2 Oscar award winning performances, so my expectations were fairly high. Unfortunately UnFortunately, I thought the movie was terrible terrib1e and I'm still left wondering how she was ever persuaded to make this movie. The script is really weak wea k.

Task: Toxic Content Detection. **Classifier:** LSTM. **Original label:** 96.7% Toxic. **Adversarial label:** 83.5% Non-toxic.

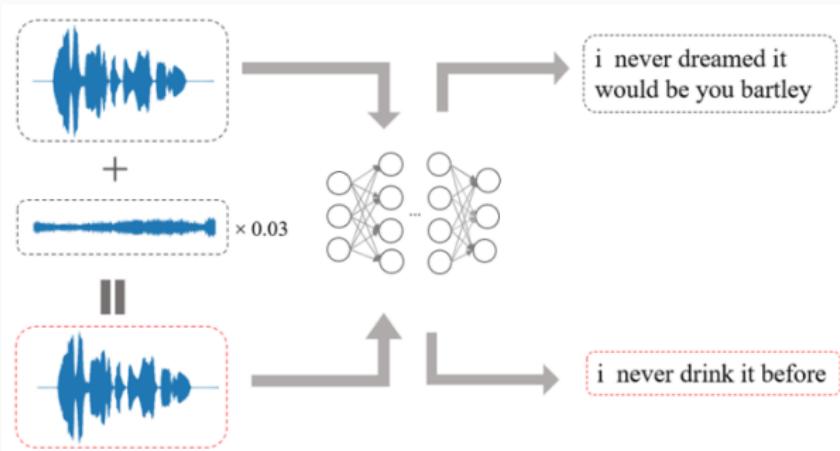
Text: hello how are you? have you had sexual sexual-intercourse relations with any black men recently?

Task: Toxic Content Detection. **Classifier:** Perspective. **Original label:** 92% Toxic. **Adversarial label:** 78% Non-toxic.

Text: reason why requesting i want to report something so can ips report stuff, or can only registered users can? if only registered users can, then i 'll request an account and it 's just not fair that i cannot edit because of this anon block shit shti c'mon, fucking fucking hell helled.

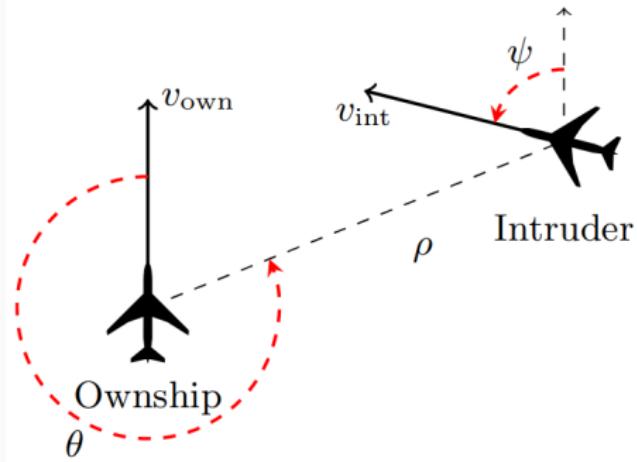
Li et al. (2018). Textbugger: Generating adversarial text against real-world applications. arXiv preprint arXiv:1812.05271.

Robustness issue in speech recognition



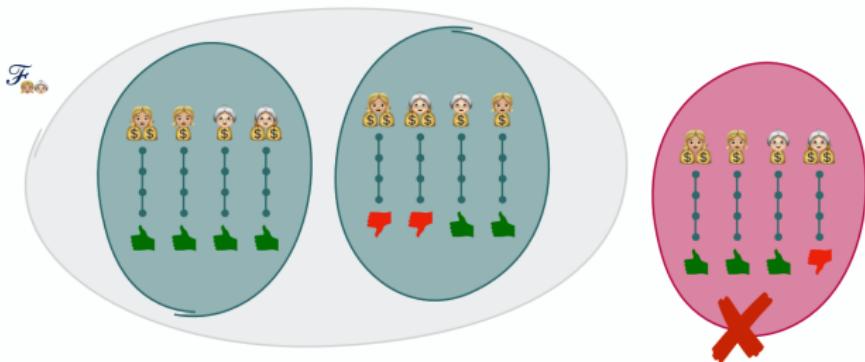
Wang et al. (2020, October). Adversarial examples attack and countermeasure for speech recognition system: A survey. In International Conference on Security and Privacy in Digital Economy.

Beyond robustness issue in neural networks - safety issue



Katz et al. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In Computer Aided Verification: 29th International Conference, CAV 2017.

Dependency Fairness



How can we find the adversarial examples? - Attack

$$\begin{array}{ccc} \text{Image of a panda} & + .007 \times & \text{Image of a panda with noise} \\ x & \text{sign}(\nabla_x J(\theta, x, y)) & = \\ \text{"panda"} & \text{"nematode"} & \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \\ 57.7\% \text{ confidence} & 8.2\% \text{ confidence} & 99.3 \% \text{ confidence} \end{array}$$

Goodfellow et al. (2014). Explaining and harnessing adversarial examples.

How can we find the adversarial examples? - Attack

$$\begin{array}{ccc} \text{Image of a panda} & + .007 \times & \text{Image of a panda} \\ x & \text{sign}(\nabla_x J(\theta, x, y)) & x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \\ \text{"panda"} & \text{"nematode"} & \text{"gibbon"} \\ 57.7\% \text{ confidence} & 8.2\% \text{ confidence} & 99.3 \% \text{ confidence} \end{array}$$

But ... what if the attack cannot find any adversarial examples?

Attack is not sound ...

		$p \models \mathcal{P}$	
		True	False
analysis(p)	True	correct	false positive
	False	false negative	correct

Soundness

For every program $p \in \mathbb{L}$, $\text{analysis}(p) = \text{true} \implies p \models \mathcal{P}$.
 \Leftrightarrow there is no false positive.

Completeness

For every program $p \in \mathbb{L}$, $\text{analysis}(p) = \text{true} \iff p \models \mathcal{P}$.
 \Leftrightarrow there is no false negative.

How to ensure the reliability of neural networks?

The complexity of verifying properties in deep neural networks with ReLUs is **NP-Complete**.

Katz et al. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. CAV 2017.

How to ensure the reliability of neural networks?

The complexity of verifying properties in deep neural networks with ReLUs is **NP-Complete**.

- **Enumeration:**

Enumerating all perturbed images in $\Phi(\mathbf{x}_0, \epsilon)$ is practically **infeasible**.

Katz et al. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. CAV 2017.

How to ensure the reliability of neural networks?

The complexity of verifying properties in deep neural networks with ReLUs is **NP-Complete**.

- **Enumeration:**

Enumerating all perturbed images in $\Phi(\mathbf{x}_0, \epsilon)$ is practically **infeasible**.

- **Attack:**

Find adversarial example → Complete, **Unsound**

Katz et al. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. CAV 2017.

How to ensure the reliability of neural networks?

The complexity of verifying properties in deep neural networks with ReLUs is **NP-Complete**.

- **Enumeration:**

Enumerating all perturbed images in $\Phi(x_0, \epsilon)$ is practically **infeasible**.

- **Attack:**

Find adversarial example → Complete, **Unsound**

- Formal Verification - Abstract Interpretation → **Sound**

Katz et al. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. CAV 2017.

How to ensure the reliability of neural networks?

The complexity of verifying properties in deep neural networks with ReLUs is **NP-Complete**.

- **Enumeration:**

Enumerating all perturbed images in $\Phi(\mathbf{x}_0, \epsilon)$ is practically **infeasible**.

- **Attack:**

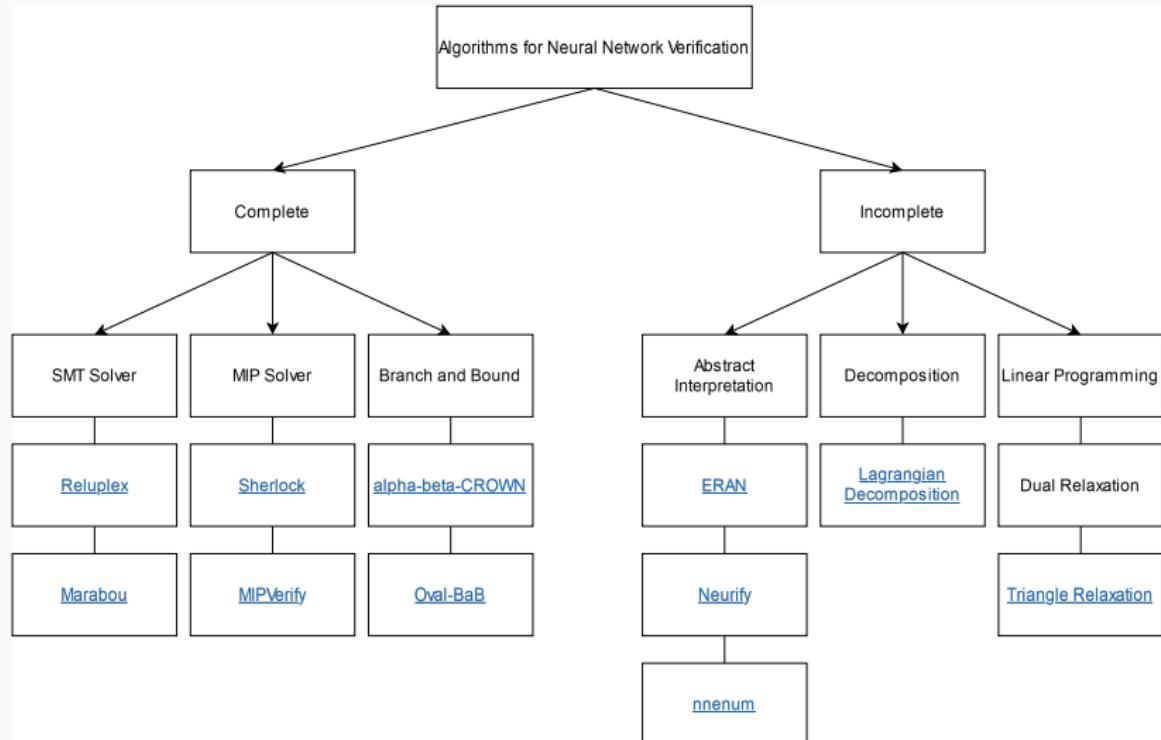
Find adversarial example → Complete, **Unsound**

- Formal Verification - Abstract Interpretation → **Sound**

- Formal Verification - Abstract Interpretation → Incomplete

Katz et al. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. CAV 2017.

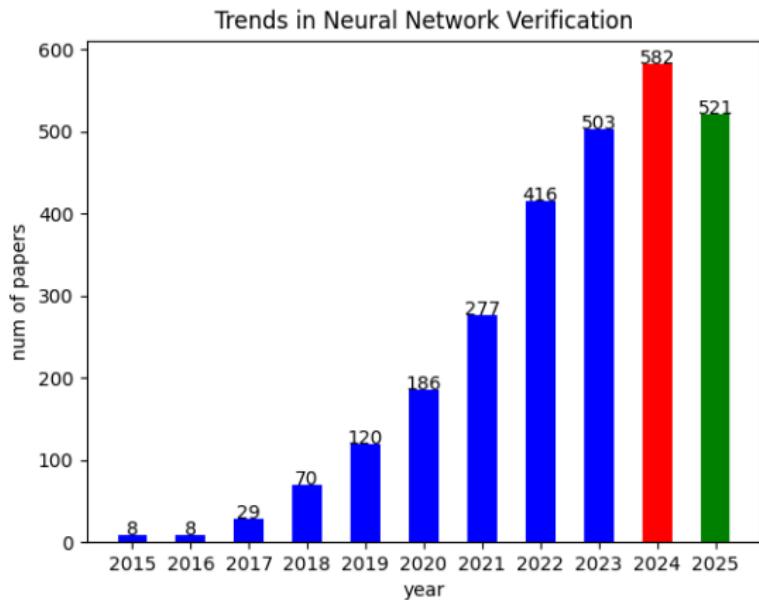
How to ensure the reliability of neural networks?



Precision vs Scalability

How is this research field growing?

Keywords: "neural network verification" on GoogleScholar until 17th Nov 2025.



Neural Network Verification

To ensure the reliability of neural networks

A **neural network** consists of *an input layer, several hidden layers, and an output layer* where each layer is made up of several neurons.

Definition (Layer)

Let n_ℓ be the number of neurons in the layer ℓ . Then a layer function $N_\ell: \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell+1}}$ is defined as follows:

$$N_\ell(\mathbf{x}) \triangleq \sigma(\mathbf{W}_\ell \mathbf{x} + \mathbf{b}_\ell),$$

where σ is an element-wise nonlinear activation function i.e., relu, sigmoid, etc.

Definition (Neural Network)

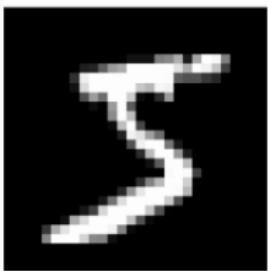
Let \mathbf{L} be the index set of layers. A neural network is a function $N: \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_{|\mathbf{L}|}}$ defined as follows:

$$N \triangleq N_{|\mathbf{L}|-1} \circ N_{|\mathbf{L}|-2} \circ \cdots \circ N_0.$$

To ensure the reliability of neural networks

Definition (Preconditions)

The preconditions in the input layer are defined by the set of formulas
 $\Phi(\mathbf{x}_0, \epsilon) \triangleq \{\mathbf{x} \in \mathbb{R}^{n_0} \mid p(\mathbf{x}, \mathbf{x}_0) \leq \epsilon\}$, where $p: \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \rightarrow \mathbb{R}$ is a function defining a perturbation for each input dimension and $\epsilon \in \mathbb{R}$ is the maximum perturbation.



Origin Image



L infinity

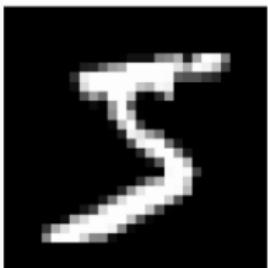


Rotation

To ensure the reliability of neural networks

Definition (Postconditions)

The postconditions in the output layer are defined by the set of formulas Ψ to describe that the output for perturbed inputs should be the same as the true output for the original input.



Origin Image



L infinity



Rotation

All of them have to be classified as 5.

Syntax of First-Order Logic (FOL) - recap

Let $S = \langle X, F, P \rangle$ be a *first-order signature* where X set of variables, F set of function symbols and P set of predicate symbols.

$\langle t \rangle ::= x$	<i>variable</i> $x \in X$
$f(t, \dots, t)$	<i>function</i> $f \in F$
$\langle \varphi \rangle ::= p(t, \dots, t)$	<i>predicate</i> $p \in P$
$\neg \varphi$	<i>negation</i>
$\varphi \diamond \varphi$	<i>connector</i> $\diamond \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$
$\exists x, \varphi$	<i>existential quantifier</i>
$\forall x, \varphi$	<i>universal quantifier</i>

Let Φ the set of well-formed formulas.

Semantics of FOL - recap

A *structure A* is a tuple $(\mathbb{U}, \llbracket \cdot \rrbracket_F, \llbracket \cdot \rrbracket_P)$ where

1. \mathbb{U} is a non-empty set of elements—called the *universe of discourse*,
2. $\llbracket \cdot \rrbracket_F$ is a function mapping function symbols $f \in F$ with arity n to interpreted functions $\llbracket f \rrbracket_F : \mathbb{U}^n \rightarrow \mathbb{U}$, and
3. $\llbracket \cdot \rrbracket_P$ is a function mapping predicate symbols $p \in P$ with arity n to interpreted predicates $\llbracket p \rrbracket_P \subseteq \mathbb{U}^n$.

An *assignment* is a function $X \rightarrow \mathbb{U}$ mapping variables to values. We denote the set of assignment by **Asn**. Let $\rho \in \mathbf{Asn}$, we write $\rho[x \mapsto d]$ the assignment in which we updated the value of x by d in ρ .

Entailment - recap

The syntax and semantics are related by the ternary relation $A \models_\rho \varphi$, called the *entailment*, where A is a structure, $\rho \in \mathbf{Asn}$ and $\varphi \in \Phi$. It is read as “the formula φ is satisfied by the assignment ρ in the structure A ”. We first give the interpretation function $\llbracket \cdot \rrbracket_\rho$ for evaluating the terms of the language:

$$\begin{aligned}\llbracket x \rrbracket_\rho &= \rho(x) \text{ if } x \in X \\ \llbracket f(t_1, \dots, t_n) \rrbracket_\rho &= \llbracket f \rrbracket_F(\llbracket t_1 \rrbracket_\rho, \dots, \llbracket t_n \rrbracket_\rho)\end{aligned}$$

The relation \models is defined inductively as follows:

$A \models_\rho p(t_1, \dots, t_n)$	iff $(\llbracket t_1 \rrbracket_\rho, \dots, \llbracket t_n \rrbracket_\rho) \in \llbracket p \rrbracket_P$
$A \models_\rho \varphi_1 \wedge \varphi_2$	iff $A \models_\rho \varphi_1$ and $A \models_\rho \varphi_2$
$A \models_\rho \varphi_1 \vee \varphi_2$	iff $A \models_\rho \varphi_1$ or $A \models_\rho \varphi_2$
$A \models_\rho \neg \varphi$	iff $A \models_\rho \varphi$ does not hold
$A \models_\rho \exists x, \varphi$	iff there exists $d \in \mathbb{U}$ such that $A \models_{\rho[x \mapsto d]} \varphi$
$A \models_\rho \forall x, \varphi$	iff for all $d \in \mathbb{U}$, we have $A \models_{\rho[x \mapsto d]} \varphi$

Neural network verification

By *neural network* N , *preconditions* $\Phi(\mathbf{x}_0, \epsilon)$, and *postconditions* Ψ , we can define the neural network verification problem as:

$$A \models_{N(\mathbf{x})} \neg\Psi, \quad \forall \mathbf{x} \in \Phi(\mathbf{x}_0, \epsilon) \quad (1)$$

If (1) is true (satisfied), return **SAT** and at least one counter example.
Otherwise, return **UNSAT**.

Neural network verification

By *neural network* N , *preconditions* $\Phi(\mathbf{x}_0, \epsilon)$, and *postconditions* Ψ , we can define the neural network verification problem as:

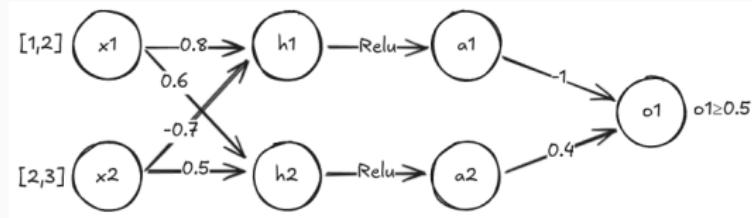
$$A \models_{N(\mathbf{x})} \neg\Psi, \quad \forall \mathbf{x} \in \Phi(\mathbf{x}_0, \epsilon) \quad (1)$$

If (1) is true (satisfied), return **SAT** and at least one counter example.
Otherwise, return **UNSAT**.

Note that, for clarity, we will use the simplified notations in the following lecture:

$$N(\mathbf{x}) \models \neg\Psi, \quad \forall \mathbf{x} \in \Phi(\mathbf{x}_0, \epsilon)$$

Example - neural network verification



Neural network N : the given directed acyclic graph in above.

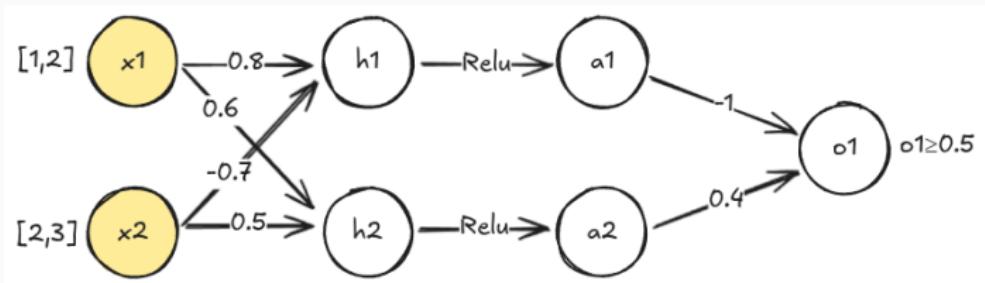
Preconditions $\Phi(\mathbf{x}_0, \epsilon)$: $1 \leq x_1 \leq 2 \wedge 2 \leq x_2 \leq 3$.

Postconditions Ψ : $o_1 \geq 0.5$

$$N(x_1, x_2) \models o_1 < 0.5, \text{ where } 1 \leq x_1 \leq 2 \wedge 2 \leq x_2 \leq 3$$

* In this lecture, we only focus on ReLU activation function in N .

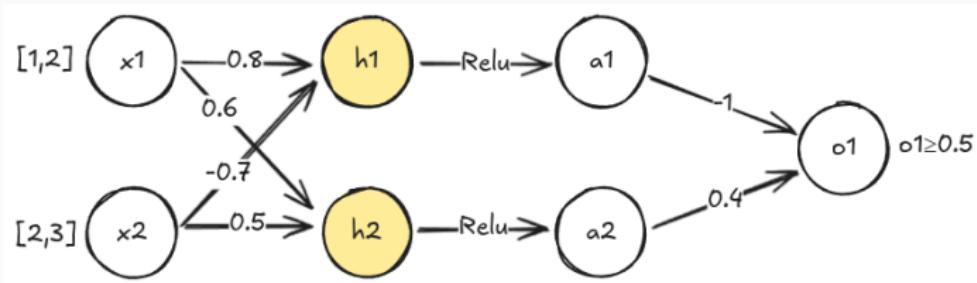
Example - How to verify?



The bounds of neurons in the input layer are given by preconditions:

$$1 \leq x_1 \leq 2 \wedge 2 \leq x_2 \leq 3, \text{ where } x_1, x_2 \in \mathbb{R}$$

Example - How to verify?

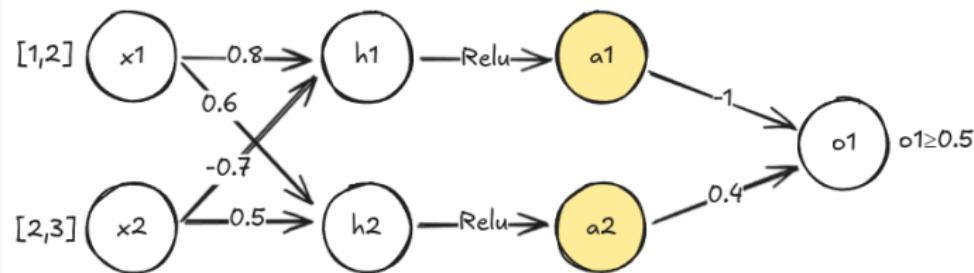


The first layer, fully connected layer, can be formulated by the following constraints:

$$h_1 = 0.8x_1 - 0.7x_2, \text{ where } h_1, x_1, x_2 \in \mathbb{R}$$

$$h_2 = 0.6x_1 + 0.5x_2, \text{ where } h_2, x_1, x_2 \in \mathbb{R}$$

Example - How to verify?

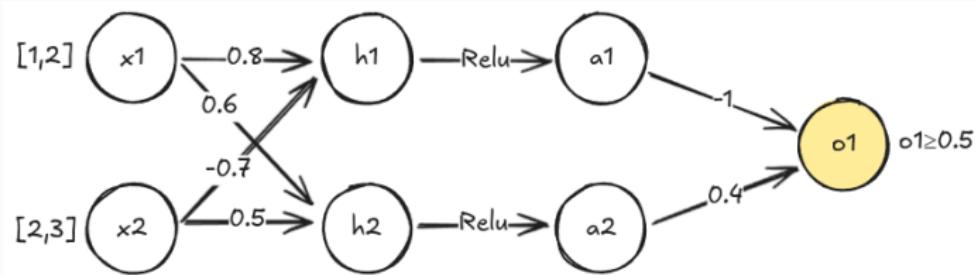


The second layer, Relu activation layer, can be formulated by the following constraints:

$$a_1 = \text{ReLU}(0, h_1) = \max(0, h_1), \text{ where } a_1 \in \mathbb{R}^+ \cup \{0\}, h_1 \in \mathbb{R}$$

$$a_2 = \text{ReLU}(0, h_2) = \max(0, h_2), \text{ where } a_2 \in \mathbb{R}^+ \cup \{0\}, h_2 \in \mathbb{R}$$

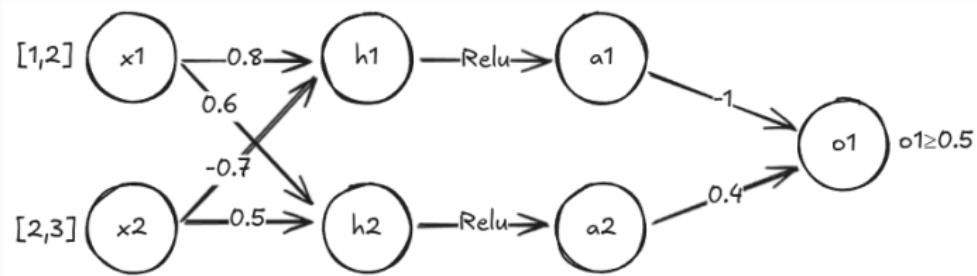
Example - How to verify?



Last, the output layer can be formulated as the following constraints:

$$o_1 = -a_1 + 0.4a_2, \text{ where } o_1 \in \mathbb{R}, a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$$

Example - How to verify?



$$1 \leq x_1 \leq 2 \wedge 2 \leq x_2 \leq 3$$

$$h_1 = 0.8x_1 - 0.7x_2 \wedge h_2 = 0.6x_1 + 0.5x_2$$

$$a_1 = \max(0, h_1) \wedge a_2 = \max(0, h_2)$$

$$o_1 = -a_1 + 0.4a_2$$

$$\textcolor{red}{x_1}, \textcolor{red}{x_2}, h_1, h_2, o_1 \in \mathbb{R}, a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$$

Formal Verification - Abstract Interpretation

What is abstract interpretation?

Abstract interpretation is a **sound** and **incomplete** framework for analyzing programs by overapproximating (abstract domain) the program semantics (concrete domain).

Example

Let x be a real number, its possible value is between 1 and 2.

$$\Psi \triangleq \{x + 5 \leq 7\}, \neg\Psi \triangleq \{x + 5 > 7\}.$$

We cannot enumerate all possible values of x to check if $\neg\Psi$ is satisfied.

Instead, we can *overapproximate* x by an interval abstract domain $[1, 2]$. Then, we use interval arithmetic to compute the overapproximated bounds of $x + 5$:

$$x + 5 \in [1 + 5, 2 + 5] = [6, 7].$$

Therefore, we can conclude that $x + 5 > 7$ is **UNSAT**. □

Concrete Domain of Neural Network Verification

Concrete domain - recap

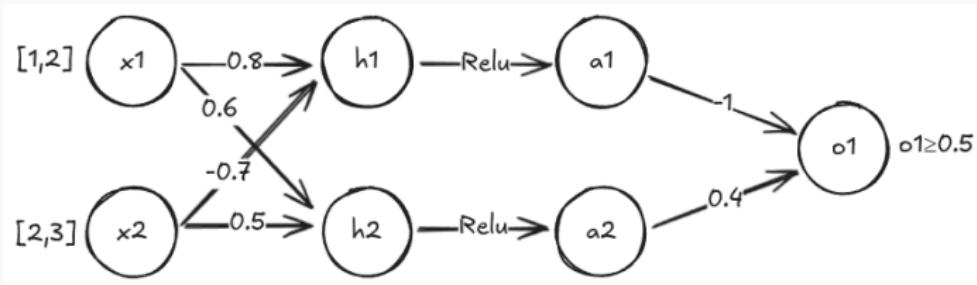
Given a structure A , we define the *concrete interpretation function* as:

$$\begin{aligned}\llbracket . \rrbracket^\flat &: \Phi \rightarrow \mathcal{P}(\mathbf{Asn}) \\ \llbracket \varphi \rrbracket^\flat &= \{\rho \in \mathbf{Asn} \mid A \models_\rho \varphi\}\end{aligned}$$

- We call the *concrete domain* the lattice $D^\flat \triangleq \langle \mathcal{P}(\mathbf{Asn}), \subseteq \rangle$ with $\llbracket . \rrbracket^\flat$.
- A *solution* of the formula φ is an assignment $s \in \llbracket \varphi \rrbracket^\flat$.
- **Example** in the theory of standard integer arithmetics (and $X = \{x, y\}$):

$$\begin{aligned}\llbracket x < y \wedge x \geq 0 \rrbracket^\flat &= \{ \\ &\quad \{x \mapsto 0, y \mapsto 1\} \\ &\quad \{x \mapsto 0, y \mapsto 2\} \\ &\quad \dots \\ &\quad \{x \mapsto 1, y \mapsto 2\} \\ &\quad \dots \\ &\quad \}\end{aligned}$$

Concrete domain of neural network verification



$$D^b = \{\rho \in \mathbf{Asn} \mid \llbracket \Phi \rrbracket^b\},$$

where $\Phi = \bigwedge \{$

$$1 \leq x_1 \leq 2, 2 \leq x_2 \leq 3,$$

$$h_1 = 0.8x_1 - 0.7x_2, h_2 = 0.6x_1 + 0.5x_2,$$

$$a_1 = \max(0, h_1), a_2 = \max(0, h_2),$$

$$o_1 = -a_1 + 0.4a_2, o_1 < 0.5,$$

$$\mathbf{x}_1, \mathbf{x}_2, h_1, h_2, o_1 \in \mathbb{R}, a_1, a_2 \in \mathbb{R}^+ \cup \{0\}\}$$

Abstract Domains of Neural Network Verification

Abstract Domains - recap

Definition

An abstract domain (for constraint reasoning) is a bounded lattice $\langle A^\sharp, \sqsubseteq, \sqcup, \sqcap, \perp, \top, \mathcal{F}^\sharp[\cdot] \rangle$ such that:

- Every element of A^\sharp is representable in a machine.
- The operations on A^\sharp are efficiently computable.
- $\mathcal{F}^\sharp[\cdot] : \Phi \rightarrow (A^\sharp \rightarrow A^\sharp)$ is a sound abstraction of $\mathcal{F}[\cdot]$.

The concrete and abstract semantics are connected by a Galois connection:

$$\langle \mathcal{P}(X \rightarrow \mathbb{U}), \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A^\sharp, \sqsubseteq \rangle$$

Soundness: The abstract function $\mathcal{F}^\sharp[\varphi]$ should not remove any solution!

Interval Abstract Domain

Interval Lattice

Definition

The lattice of interval $\langle \mathcal{I}, \sqsubseteq, \sqcup, \sqcap, \perp, [-\infty, \infty] \rangle$ is defined as:

$$\mathcal{I} \triangleq \{[a, b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{\infty\}, a \sqsubseteq b\} \cup \{\perp\}$$

with the following operations:

- $[a, b] \sqsubseteq [c, d] \Leftrightarrow a \geq c \wedge b \leq d.$
- $[a, b] \sqcup [c, d] \triangleq [\min(a, c), \max(b, d)].$
- $[a, b] \sqcap [c, d] \triangleq [\max(a, c), \min(b, d)].$

We also define projection functions $\lfloor [a, b] \rfloor \triangleq a$ and $\lceil [a, b] \rceil \triangleq b.$

Interval abstract domain

Let interval abstract element be a function $d \in X \rightarrow \mathcal{I}$ and interval abstract domain $A^\sharp \triangleq \langle X \rightarrow \mathcal{I}, \sqsubseteq \rangle$ be a set of functions.

- $\mathcal{I}[\cdot] \in \Phi \times (X \rightarrow \mathcal{I}) \rightarrow (X \rightarrow \mathcal{I})$

Example

Let $x \in \mathbb{R}$, its possible value is between -1.5 and 3.0.

Therefore, its interval abstract element is defined as:

$$\mathcal{I}[-1.5 \leq x \leq 3.0]d = \{x \mapsto [-1.5, 3.0]\}.$$

Furthermore, we denote by $A_{n_\ell}^\sharp \triangleq \langle X_{n_\ell} \rightarrow \mathcal{I}_{n_\ell}, \sqsubseteq \rangle$ an interval abstract domain over all the neurons at layer ℓ .

Interval abstract domain - preconditions

Suppose that the preconditions $\Phi(\mathbf{x}_0, \epsilon)$ will provide a lower and upper bound for each input dimension, i.e., $\forall i \in \{1, \dots, n_0\}, \ell_i \leq x_i \leq u_i$. Therefore, the interval abstract domain for each input neuron can be defined as:

$$\mathcal{I}[\ell_i \leq x_i \leq u_i]d \triangleq \{x_i \mapsto [\ell_i, u_i]\}.$$

Interval abstract domain - affine abstract transformer

For a single neuron $y_i \in \mathbb{R}$ at affine layer, its value through an affine function $N_\ell: \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell+1}}$ is defined as:

$$\forall i \in \{1, \dots, n_{\ell+1}\}, y_i = \sum_j w_{ij}x_j + b_i$$

Let $\forall j, x_j \mapsto [\ell_j, u_j]$, the affine abstract transformer $N_\ell^\#: A_{n_\ell}^\# \rightarrow A_{n_{\ell+1}}^\#$ is defined as:

$$\mathcal{I}[y_i = \sum_j w_{ij}x_j + b_i]d \triangleq \{y_i \mapsto \sum_j w_{ij}x_j + [b_i, b_i]\}$$

Interval abstract domain - relu abstract transformer

For a single neuron $y_i \in \mathbb{R}$ at relu layer, its value through a relu function $N_\ell: \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell+1}}$ is defined as:

$$\forall i \in \{1, \dots, n_{\ell+1}\}, y_i = \max(x_j, 0), \text{ where } i = j.$$

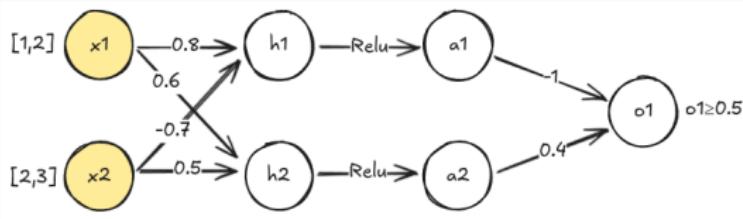
Let $\forall j, x_j \mapsto [\ell_j, u_j]$, the relu abstract transformer $N_\ell^\#: A_{n_\ell}^\# \rightarrow A_{n_{\ell+1}}^\#$ is defined as:

$$\mathcal{I}[y_i = \max(x_j, 0)]d \triangleq \{y_i \mapsto [\max(\ell_j, 0), \max(u_j, 0)]\}$$

A General Forward Bound Propagation

```
function VERIFY( $N, \mathbf{L}, \Phi(\mathbf{x}_0, \epsilon), \Psi$ )
     $bound_0 \mapsto \Phi(\mathbf{x}_0, \epsilon)$ 
    for all  $\ell \in \langle 0, \dots, |\mathbf{L}| - 1 \rangle$  do
         $bound_{\ell+1} \mapsto N_\ell^\sharp(bound_\ell)$ 
    end for
    if  $bound_{|\mathbf{L}|} \models \neg \Psi$  then
        return UNSAT
    else
        return SAT
    end if
end function
```

Example - Interval Abstract Domain



Concrete preconditions:

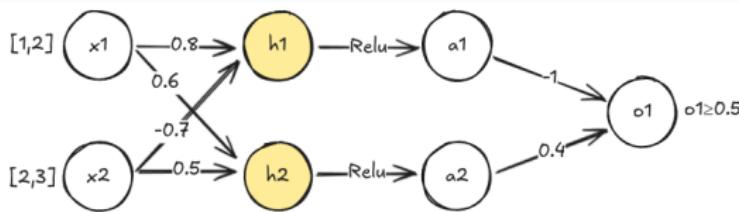
$$1 \leq x_1 \leq 2 \wedge 2 \leq x_2 \leq 3, \text{ where } x_1, x_2 \in \mathbb{R}$$

Abstract interval preconditions:

$$\mathcal{I}[1 \leq x_1 \leq 2]d = \{x_1 \mapsto [1, 2]\}$$

$$\mathcal{I}[2 \leq x_2 \leq 3]d = \{x_2 \mapsto [2, 3]\}$$

Example - Interval Abstract Domain



The first layer, fully connected layer, computes the hidden neurons $h_1, h_2 \in \mathbb{R}$ by a linear combination:

$$h_1 = 0.8x_1 - 0.7x_2, \text{ where } h_1, x_1, x_2 \in \mathbb{R}$$

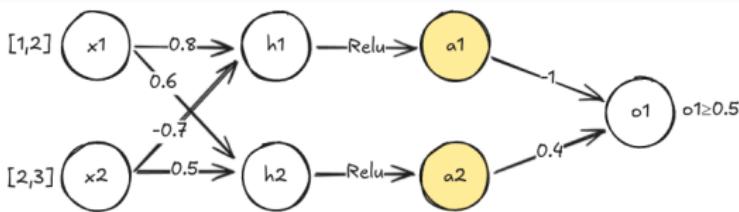
$$h_2 = 0.6x_1 + 0.5x_2, \text{ where } h_2, x_1, x_2 \in \mathbb{R}$$

Abstract interval bounds:

$$\mathcal{I}[h_1 = 0.8x_1 - 0.7x_2]d = \{h_1 \mapsto [-1.3, 0.2]\}$$

$$\mathcal{I}[h_2 = 0.6x_1 + 0.5x_2]d = \{h_2 \mapsto [1.6, 2.7]\}$$

Example - Interval Abstract Domain



The second layer, Relu activation layer, computes the activated neurons $a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$ by the ReLU function:

$$a_1 = \text{ReLU}(0, h_1) = \max(0, h_1), \text{ where } a_1 \in \mathbb{R}^+ \cup \{0\}, h_1 \in \mathbb{R}$$

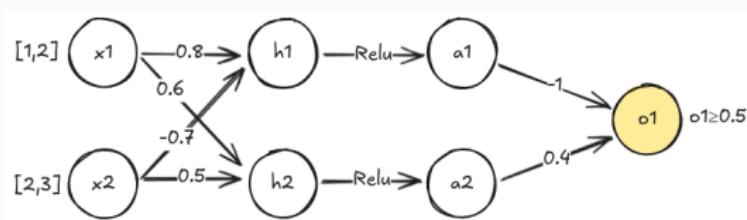
$$a_2 = \text{ReLU}(0, h_2) = \max(0, h_2), \text{ where } a_2 \in \mathbb{R}^+ \cup \{0\}, h_2 \in \mathbb{R}$$

Abstract interval bounds:

$$\mathcal{I}[a_1 = \max(0, h_1)]d = \{a_1 \mapsto [0, 0.2]\}$$

$$\mathcal{I}[a_2 = \max(0, h_2)]d = \{a_2 \mapsto [1.6, 2.7]\}$$

Example - Interval Abstract Domain



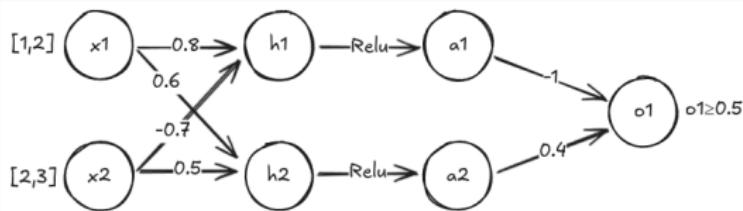
Last, the output layer computes the output neuron $o_1 \in \mathbb{R}$ by a linear combination:

$$o_1 = -a_1 + 0.4a_2, \text{ where } o_1 \in \mathbb{R}, a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$$

Abstract interval bounds:

$$\mathcal{I}[\![o_1 = -a_1 + 0.4a_2]\!]d = \{o_1 \mapsto [0.44, 1.08]\}$$

Example - Interval Domains



By interval arithmetic, we can compute the interval bounds for each neuron in the network.

$$x_1 \in [1, 2], x_2 \in [2, 3],$$

$$h_1 \in [-1.3, 0.2], h_2 \in [1.6, 2.7],$$

$$a_1 \in [0, 0.2], a_2 \in [1.6, 2.7],$$

$$o_1 \in [0.44, 1.08]$$

Is the property really unsatisfied for this neural network? or false negative?

What is the drawback of interval abstract domain?

Example

Let x, y be real numbers and $0 \leq x \leq 1 \wedge y = x - x$.

What is the interval range of y ?

By interval abstract domain,

$$\mathcal{I}[\![0 \leq x \leq 1 \wedge y = x - x]\!]d = \{x \mapsto [0, 1], y \mapsto [-1, 1]\}.$$

What is the drawback of interval abstract domain?

Example

Let x, y be real numbers and $0 \leq x \leq 1 \wedge y = x - x$.

What is the interval range of y ?

By interval abstract domain,

$$\mathcal{I}[\![0 \leq x \leq 1 \wedge y = x - x]\!]d = \{x \mapsto [0, 1], y \mapsto [-1, 1]\}.$$

Yet, we know y should be 0.

What is the drawback of interval abstract domain?

Example

Let x, y be real numbers and $0 \leq x \leq 1 \wedge y = x - x$.

What is the interval range of y ?

By interval abstract domain,

$$\mathcal{I}[\![0 \leq x \leq 1 \wedge y = x - x]\!]d = \{x \mapsto [0, 1], y \mapsto [-1, 1]\}.$$

Yet, we know y should be 0.

What is the problem here?

What is the drawback of interval abstract domain?

Example

Let x, y be real numbers and $0 \leq x \leq 1 \wedge y = x - x$.

What is the interval range of y ?

By interval abstract domain,

$$\mathcal{I}[\![0 \leq x \leq 1 \wedge y = x - x]\!]d = \{x \mapsto [0, 1], y \mapsto [-1, 1]\}.$$

Yet, we know y should be 0.

What is the problem here?

Interval abstract domain is a non-relational abstract domain!

Symbolic Interval Abstract Domain

What is symbolic interval abstract domain?

Idea: each neuron maintains interval bounds with symbolic expressions such that the relation between each neuron can be kept.

Definition (Symbolic Interval Set)

$$\mathcal{S} \triangleq \{\langle \text{expr}_\ell, \text{expr}_u, \ell, u \rangle \mid \ell \in \mathbb{R}, u \in \mathbb{R}\}$$

- $\text{expr}_\ell: \mathbf{E}_\ell \mathbf{x} + \mathbf{b}_\ell$
- $\text{expr}_u: \mathbf{E}_u \mathbf{x} + \mathbf{b}_u$
- $\ell = \min: \text{expr}_\ell$
- $u = \max: \text{expr}_u$

Note: $\mathbf{E}_\ell(\mathbf{b}_\ell)$ and $\mathbf{E}_u(\mathbf{b}_u)$ are lower and upper coefficient matrix (bias vector), respectively.

Wang et al. (2018). Formal security analysis of neural networks using symbolic intervals. In 27th USENIX Security Symposium.

Symbolic interval abstract domain

Let symbolic interval abstract element be a function $d \in X \rightarrow \mathcal{S}$ and symbolic interval abstract domain $A^\sharp \triangleq \langle X \rightarrow \mathcal{S}, \sqsubseteq \rangle$ be a set of functions.

- $\mathcal{I}[\cdot] \in \Phi \times (X \rightarrow \mathcal{S}) \rightarrow (X \rightarrow \mathcal{S})$

Example

Let $x \in \mathbb{R}$, its possible value is between -1.5 and 3.0.

Therefore, its symbolic interval abstract element is defined as:

$$\mathcal{I}[-1.5 \leq x \leq 3.0]d = \{x \mapsto \langle x, x, -1.5, 3.0 \rangle\}.$$

Furthermore, we denote by $A_{n_\ell}^\sharp \triangleq \langle X_{n_\ell} \rightarrow \mathcal{S}_{n_\ell}, \sqsubseteq \rangle$ a symbolic interval abstract domain over all the neurons at layer ℓ .

Symbolic interval abstract domain - preconditions

Suppose that the preconditions $\Phi(\mathbf{x}_0, \epsilon)$ will provide a lower and upper bound for each input dimension, i.e., $\forall i \in \{1, \dots, n_0\}, \ell_i \leq x_i \leq u_i$. Therefore, the interval abstract domain for each input neuron can be defined as:

$$\mathcal{I}[\ell_i \leq x_i \leq u_i]d \triangleq \{x_i \mapsto \langle x_i, x_i, \ell_i, u_i \rangle\}.$$

Symbolic interval abstract domain - affine abstract transformer

For a single neuron $y_i \in \mathbb{R}$ at affine layer, its value through an affine function $N_\ell: \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell+1}}$ is defined as:

$$\forall i \in \{1, \dots, n_{\ell+1}\}, y_i = \sum_j w_{ij}x_j + b_i$$

Let $\forall j, x_j \mapsto \langle \text{expr}_{\ell,j}, \text{expr}_{u,j}, \ell_j, u_j \rangle$, the affine abstract transformer $N_\ell^\#: A_{n_\ell}^\# \rightarrow A_{n_{\ell+1}}^\#$ is defined as:

$$\begin{aligned} \mathcal{I}[\![y_i = \sum_j w_{ij}x_j + b_i]\!]_d &\triangleq \{y_i \mapsto \\ &\langle \sum_j (\max(0, w_{ij})\text{exp}_{\ell,j} + \min(0, w_{ij})\text{exp}_{u,j}) + b_i, \sum_j (\min(0, w_{ij})\text{exp}_{\ell,j} + \\ &\max(0, w_{ij})\text{exp}_{u,j}) + b_i, \ell_i, u_i \rangle\}. \end{aligned}$$

Wang et al. (2018). Formal security analysis of neural networks using symbolic intervals. In 27th USENIX Security Symposium.

Symbolic interval abstract domain - relu abstract transformer

For a single neuron $y_i \in \mathbb{R}$ at relu layer, its value through a relu function $N_{\ell,i}: \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell+1}}$ is defined as:

$$\forall i \in \{1, \dots, n_{\ell+1}\}, y_i = \max(x_j, 0), \text{ where } i = j.$$

Let $\forall j, x_j \mapsto \langle \text{expr}_{\ell,j}, \text{expr}_{u,j}, \ell_j, u_j \rangle$, the relu abstract transformer $N_{\ell,i}^\#: A_{n_\ell}^\# \rightarrow A_{n_{\ell+1}}^\#$ is defined as:

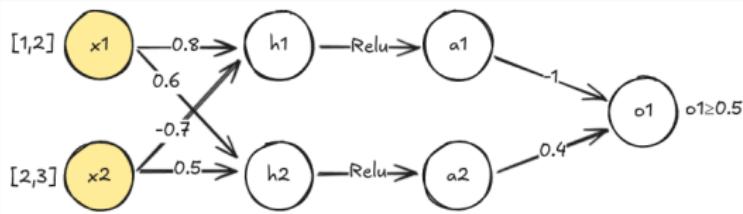
$$\mathcal{I}[\![y_i = \max(x_j, 0)]\!] d \triangleq \{y_i \mapsto$$

$$\begin{cases} \langle 0, 0, 0, 0 \rangle & \text{if } [\ell_j, u_j] \\ \langle \text{expr}_{\ell,j}, \text{expr}_{u,j}, \ell_j, u_j \rangle & \text{if } [\ell_j, u_j] \\ \langle y, y, 0, u_j \rangle & \text{if } [\ell_j, u_j] \rightsquigarrow \text{forget the dependency!} \end{cases}$$

Note: $\textcolor{red}{x} \triangleq x \leq 0$, $\textcolor{green}{x} \triangleq x > 0$

Wang et al. (2018). Formal security analysis of neural networks using symbolic intervals. In 27th USENIX Security Symposium.

Example - symbolic interval abstract domain



Concrete preconditions:

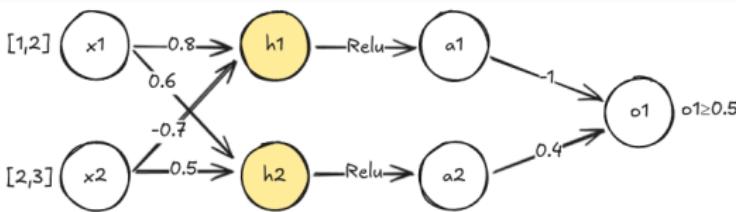
$$1 \leq x_1 \leq 2 \wedge 2 \leq x_2 \leq 3, \text{ where } x_1, x_2 \in \mathbb{R}$$

Abstract symbolic interval preconditions:

$$\mathcal{I}[1 \leq x_1 \leq 2]d = \{x_1 \mapsto \langle x_1, x_1, 1, 2 \rangle\}$$

$$\mathcal{I}[2 \leq x_2 \leq 3]d = \{x_2 \mapsto \langle x_2, x_2, 2, 3 \rangle\}$$

Example - symbolic interval abstract domain



The first layer, fully connected layer, computes the hidden neurons $h_1, h_2 \in \mathbb{R}$ by a linear combination:

$$h_1 = 0.8x_1 - 0.7x_2, \text{ where } h_1, x_1, x_2 \in \mathbb{R}$$

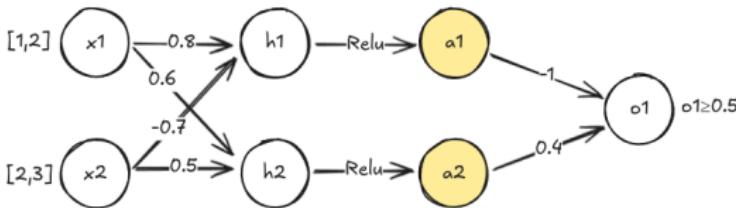
$$h_2 = 0.6x_1 + 0.5x_2, \text{ where } h_2, x_1, x_2 \in \mathbb{R}$$

Abstract symbolic interval bounds:

$$\mathcal{I}[h_1 = 0.8x_1 - 0.7x_2]d = \{h_1 \mapsto \langle 0.8x_1 - 0.7x_2, 0.8x_1 - 0.7x_2, -1.3, 0.2 \rangle\}$$

$$\mathcal{I}[h_2 = 0.6x_1 + 0.5x_2]d = \{h_2 \mapsto \langle 0.6x_1 + 0.5x_2, 0.6x_1 + 0.5x_2, 1.6, 2.7 \rangle\}$$

Example - symbolic interval abstract domain



The second layer, Relu activation layer, computes the activated neurons $a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$ by the ReLU function:

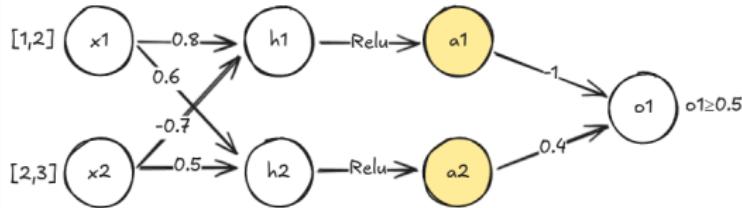
$$a_1 = \text{ReLU}(0, h_1) = \max(0, h_1), \text{ where } a_1 \in \mathbb{R}^+ \cup \{0\}, h_1 \in \mathbb{R}$$

Abstract symbolic interval bounds:

$$h_1 \mapsto \langle 0.8x_1 - 0.7x_2, 0.8x_1 - 0.7x_2, -1.3, 0.2 \rangle$$

$$\mathcal{I}[\![a_1 = \max(0, h_1)]\!]d = \{a_1 \mapsto \underbrace{\langle a_1, a_1, 0, 0.2 \rangle}_{\text{interval bias}}\}$$

Example - symbolic interval abstract domain



The second layer, Relu activation layer, computes the activated neurons $a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$ by the ReLU function:

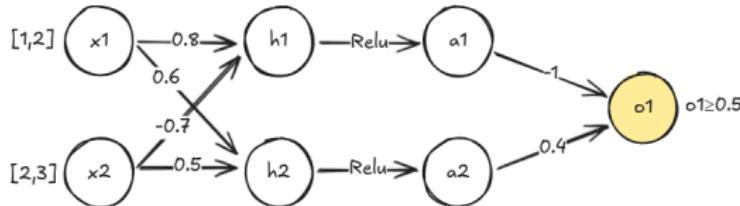
$$a_2 = \text{ReLU}(0, h_2) = \max(0, h_2), \text{ where } a_2 \in \mathbb{R}^+ \cup \{0\}, h_2 \in \mathbb{R}$$

Abstract symbolic interval bounds:

$$h_2 \mapsto \langle 0.6x_1 + 0.5x_2, 0.6x_1 + 0.5x_2, [1.6, 2.7] \rangle$$

$$\mathcal{I}[\![a_2 = \max(0, h_2)]\!]d = \{a_2 \mapsto \langle 0.6x_1 + 0.5x_2, 0.6x_1 + 0.5x_2, 1.6, 2.7 \rangle\}$$

Example - symbolic interval abstract domain



Last, the output layer computes the output neuron $o_1 \in \mathbb{R}$ by a linear combination:

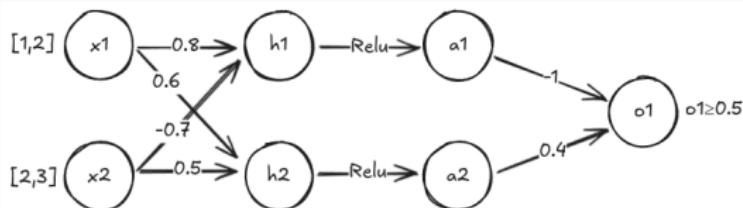
$$o_1 = -a_1 + 0.4a_2, \text{ where } o_1 \in \mathbb{R}, a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$$

Abstract symbolic interval bounds:

$$\mathcal{I}[o_1 = -a_1 + 0.4a_2]d$$

$$= \{o_1 \mapsto \langle -a_1 + 0.24x_1 + 0.2x_2, -a_1 + 0.24x_1 + 0.2x_2, \textcolor{red}{0.44}, 1.08 \rangle\}$$

Example - symbolic interval abstract domain



By using symbolic interval domain, we obtain the interval bounds for each neuron in the network.

$$x_1 \in [1, 2], x_2 \in [2, 3],$$

$$h_1 \in [-1.3, 0.2], h_2 \in [1.6, 2.7],$$

$$a_1 \in [0, 0.2], a_2 \in [1.6, 2.7],$$

$$o_1 \in [0.44, 1.08]$$

Why the overapproximated bounds for o_1 is still the same as using interval abstract domain?

DeepPoly Abstract Domain

What is DeepPoly domain?

Definition (DeepPoly Interval Set)

$$\mathcal{D} \triangleq \{\langle expr_\ell, expr_u, \ell, u \rangle \mid \ell \in \mathbb{R}, u \in \mathbb{R}\}$$

- Let $\overleftarrow{s}: A_{n_\ell}^\# \rightarrow A_{n_0}^\#$ be a back-substitution function. it will be applied recursively until reaching the input layer.
- $expr_\ell: \mathbf{E}_\ell x + \mathbf{b}_\ell$
- $expr_u: \mathbf{E}_u x + \mathbf{b}_u$
- $\ell = \min: \overleftarrow{s}(expr_\ell)$
- $u = \max: \overleftarrow{s}(expr_u)$

Note: $\mathbf{E}_\ell(\mathbf{b}_\ell)$ and $\mathbf{E}_u(\mathbf{b}_u)$ are lower and upper coefficient matrix (bias vector), respectively.

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain

Let deeppoly abstract element be a function $d \in X \rightarrow \mathcal{D}$ and deeppoly abstract domain $A^\sharp \triangleq \langle X \rightarrow \mathcal{D}, \sqsubseteq \rangle$ be a set of functions.

- $\mathcal{I}[\cdot] \in \Phi \times (X \rightarrow \mathcal{D}) \rightarrow (X \rightarrow \mathcal{D})$

Example

Let $x \in \mathbb{R}$, its possible value is between -1.5 and 3.0.

Therefore, its deeppoly abstract element is defined as:

$$\mathcal{I}[-1.5 \leq x \leq 3.0]d = \{x \mapsto \langle x, x, -1.5, 3.0 \rangle\}$$

Furthermore, we denote by $A_{n_\ell}^\sharp \triangleq \langle X_{n_\ell} \rightarrow \mathcal{D}_{n_\ell}, \sqsubseteq \rangle$ a deeppoly abstract domain over all the neurons at layer ℓ .

DeepPoly vs CROWN

Basically, they are the same and published at almost same time.
The only difference is that

- DeepPoly in *abstract interpretation* perspective:
<https://github.com/eth-sri/eran>.
- CROWN in *optimization* perspective:
<https://github.com/Verified-Intelligence/alpha-beta-CROWN>.

DeepPoly domain - preconditions

Suppose that the preconditions $\Phi(\mathbf{x}_0, \epsilon)$ will provide a lower and upper bound for each input dimension, i.e., $\forall i \in \{1, \dots, n_0\}, \ell_i \leq x_i \leq u_i$. Therefore, the interval abstract domain for each input neuron can be defined as:

$$\mathcal{I}[\ell_i \leq x_i \leq u_i]d \triangleq \{x_i \mapsto \langle x_i, x_i, \ell_i, u_i \rangle\}.$$

DeepPoly domain - affine abstract transformer

For a single neuron $y_i \in \mathbb{R}$ at affine layer, its value through an affine function $N_\ell: \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell+1}}$ is defined as:

$$\forall i \in \{1, \dots, n_{\ell+1}\}, y_i = \sum_j w_{ij}x_j + b_i$$

Let $\forall j, x_j \mapsto \langle \text{expr}_{\ell,j}, \text{expr}_{u,j}, \ell_j, u_j \rangle$, the affine abstract transformer $N_\ell^\sharp: A_{n_\ell}^\sharp \rightarrow A_{n_{\ell+1}}^\sharp$ is defined as:

$$\mathcal{I}[y_i = \sum_j w_{ij}x_j + b_i]d \triangleq \{y_i \mapsto \langle \sum_j w_{ij}x_j + b_i, \sum_j w_{ij}x_j + b_i, \boxed{\ell_i, u_i} \rangle\}.$$

To compute ℓ_i and u_i , we need to apply *back-substitution procedure*. To substitute each variable at layer ℓ by the variables at layer $\ell - 1$ until reaching the input layer. (we will see an example later.)

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain - relu abstract transformer

For a single neuron $y_i \in \mathbb{R}$ at relu layer, its value through a relu function $N_\ell: \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell+1}}$ is defined as:

$$\forall i \in \{1, \dots, n_{\ell+1}\}, y_i = \max(x_j, 0), \text{ where } i = j.$$

Let $\forall j, x_j \mapsto \langle \text{expr}_\ell, \text{expr}_u, \ell_j, u_j \rangle$, the relu abstract transformer

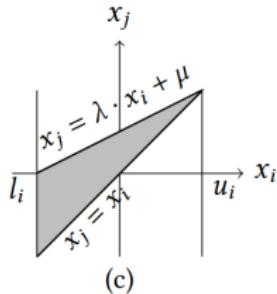
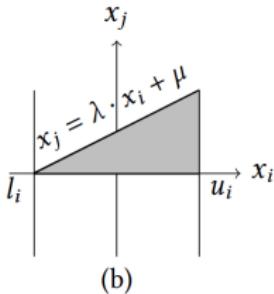
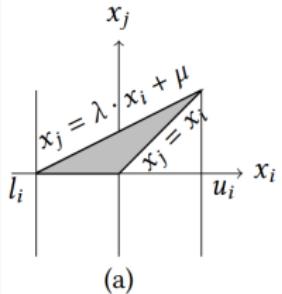
$N_\ell^\#: A_{n_\ell}^\# \rightarrow A_{n_{\ell+1}}^\#$ is defined as:

$$\mathcal{I}[\![y_i = \max(x_j, 0)]\!] d \triangleq \{y_i \mapsto \begin{cases} \langle 0, 0, 0, 0 \rangle & \text{if } [\ell_j, u_j] \\ \langle \text{expr}_\ell, \text{expr}_u, \ell_j, u_j \rangle & \text{if } [\ell_j, u_j] \\ \langle 0, u_j(x_j - \ell_j)/(u_j - \ell_j), 0, u_j \rangle & \text{if } [\ell_j, u_j] \wedge u_j \leq -\ell_j \\ \langle x_j, u_j(x_j - \ell_j)/(u_j - \ell_j), \ell_j, u_j \rangle & \text{if } [\ell_j, u_j] \wedge u_j > -\ell_j \end{cases}$$

Note: $\boxed{x} \triangleq x \leq 0$, $\boxed{x} \triangleq x > 0$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain - relu abstract transformer



Let $y = \max(0, x)$, $x \in [\ell, u]$, $b \in \{0, 1\}$, its MIP formulations are:

$$x \leq y$$

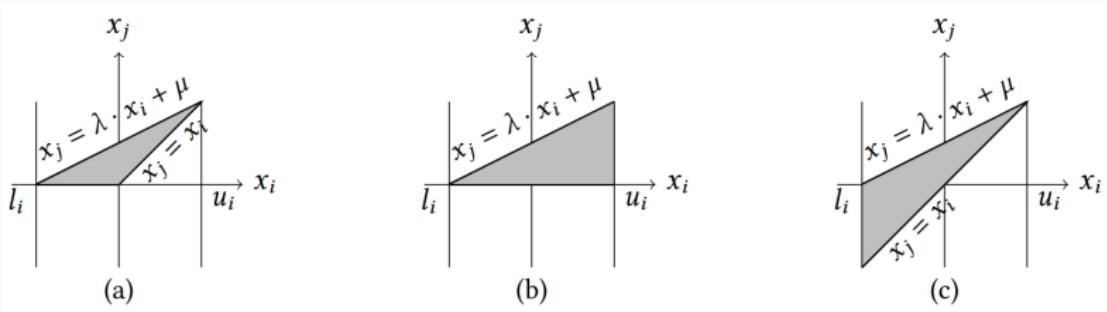
$$0 \leq y$$

$$y \leq x - (1 - b)\ell$$

$$y \leq bu$$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain - relu abstract transformer



$$x \leq y$$

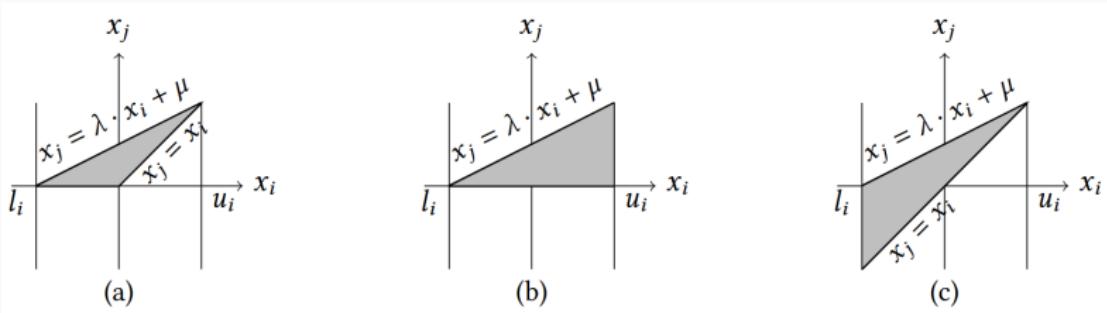
$$0 \leq y$$

$$y \leq x - (1 - b)\ell$$

$$y \leq bu$$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain - relu abstract transformer



$$y \leq x - (1 - b)\ell$$

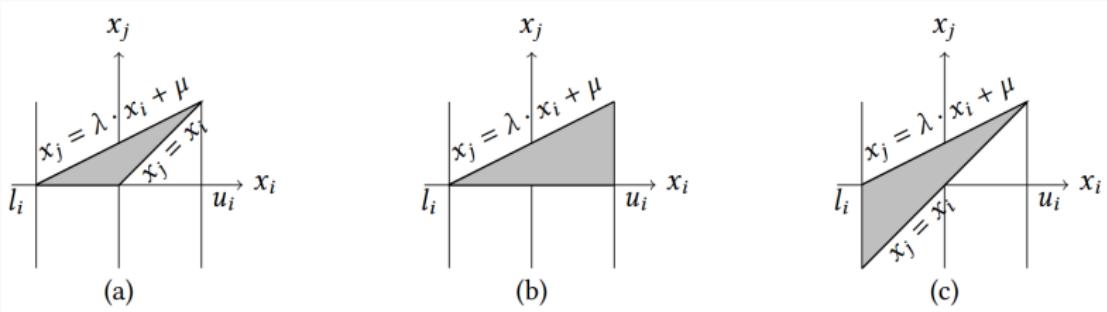
$$y \leq bu$$

$$y \leq x - \ell + b\ell$$

$$y \leq bu$$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain - relu abstract transformer



$$y \leq x - \ell + b\ell$$

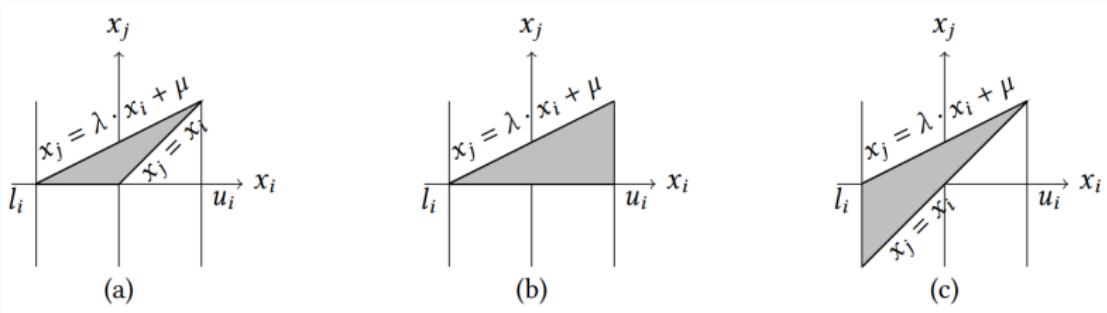
$$y \leq bu$$

$$uy \leq ux - u\ell + b\ell u$$

$$\ell y \leq b\ell u$$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain - relu abstract transformer



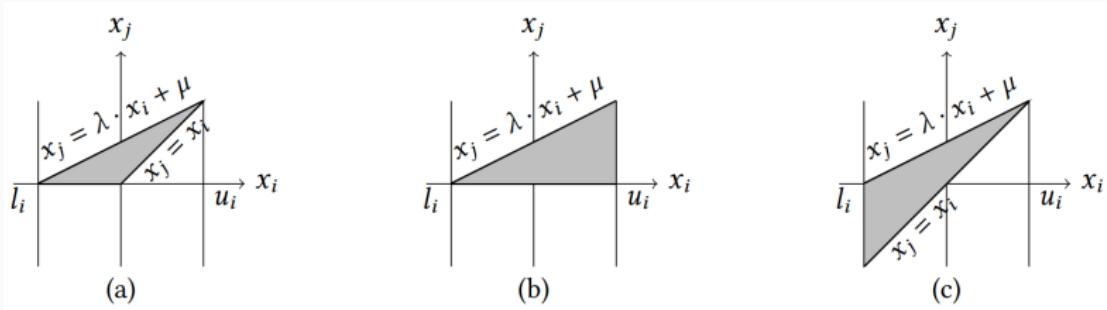
$$uy \leq ux - u\ell + b\ell u$$

$$\ell y \leq b\ell u$$

$$uy - \ell y \leq ux - u\ell$$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

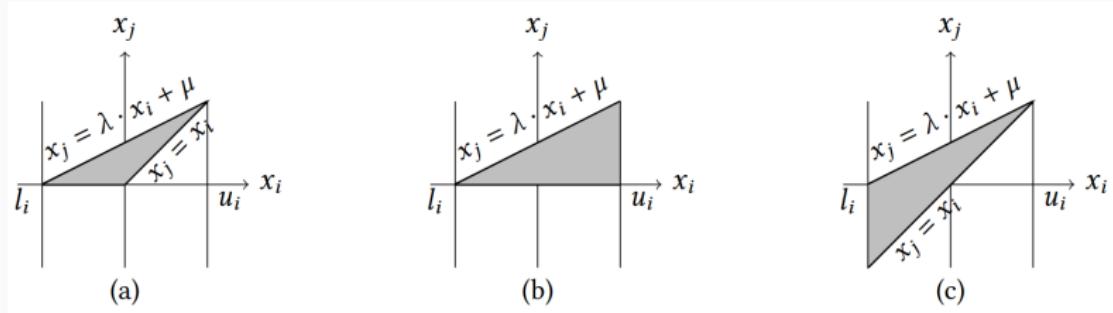
DeepPoly domain - relu abstract transformer



$$\begin{aligned}uy - \ell y &\leq ux - u\ell \\(u - \ell)y &\leq u(x - \ell) \\y &\leq u(x - \ell)/u - \ell\end{aligned}$$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

DeepPoly domain - relu abstract transformer

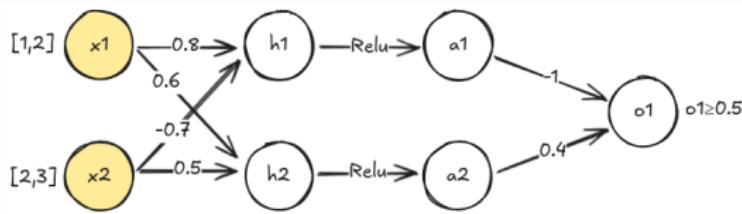


To decide which ℓ should be used
such that the overapproximated area is minimal, we need to consider:

$$y \geq \begin{cases} 0 & \text{if } u_x \leq -\ell_x \\ x & \text{o.w.} \end{cases}$$

Singh et al. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages.

Example - deeppoly domain



Concrete preconditions:

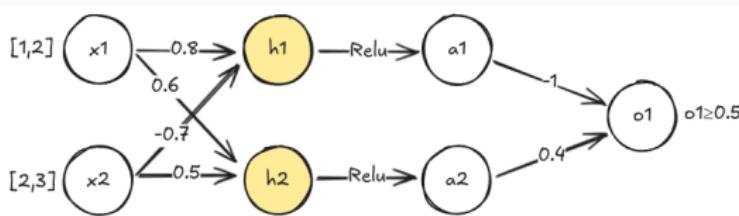
$$1 \leq x_1 \leq 2 \wedge 2 \leq x_2 \leq 3, \text{ where } x_1, x_2 \in \mathbb{R}$$

DeepPoly preconditions:

$$\mathcal{I}[1 \leq x_1 \leq 2]d = \{x_1 \mapsto \langle x_1, x_1, 1, 2 \rangle\}$$

$$\mathcal{I}[2 \leq x_2 \leq 3]d = \{x_2 \mapsto \langle x_2, x_2, 2, 3 \rangle\}$$

Example - deeppoly domain



The first layer, fully connected layer, computes the hidden neurons $h_1, h_2 \in \mathbb{R}$ by a linear combination:

$$h_1 = 0.8x_1 - 0.7x_2, \text{ where } h_1, x_1, x_2 \in \mathbb{R}$$

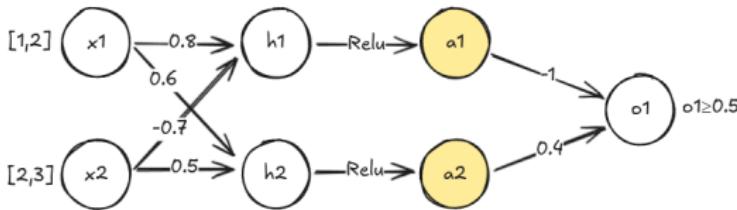
$$h_2 = 0.6x_1 + 0.5x_2, \text{ where } h_2, x_1, x_2 \in \mathbb{R}$$

DeepPoly bounds:

$$\mathcal{I}[h_1 = 0.8x_1 - 0.7x_2]d = \{h_1 \mapsto \langle 0.8x_1 - 0.7x_2, 0.8x_1 - 0.7x_2, -1.3, 0.2 \rangle\}$$

$$\mathcal{I}[h_2 = 0.6x_1 + 0.5x_2]d = \{h_2 \mapsto \langle 0.6x_1 + 0.5x_2, 0.6x_1 + 0.5x_2, 1.6, 2.7 \rangle\}$$

Example - deeppoly domain



The second layer, Relu activation layer, computes the activated neurons $a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$ by the ReLU function:

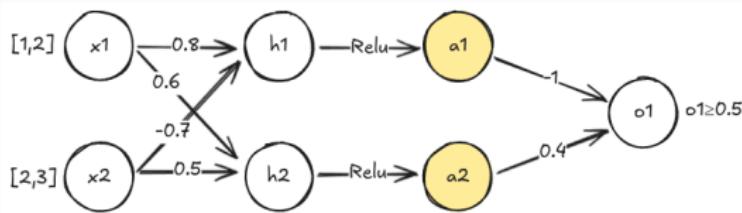
$$a_1 = \text{ReLU}(0, h_1) = \max(0, h_1), \text{ where } a_1 \in \mathbb{R}^+ \cup \{0\}, h_1 \in \mathbb{R}$$

DeepPoly bounds:

$$h_1 \mapsto \langle 0.8x_1 - 0.7x_2, 0.8x_1 - 0.7x_2, \boxed{-1.3}, \boxed{0.2} \rangle$$

$$\mathcal{I}[a_1 = \max(0, h_1)]d = \{a_1 \mapsto \langle 0, \frac{0.2}{1.5}(h_1 - (-1.3)), 0, 0.2 \rangle\}$$

Example - deeppoly domain



The second layer, Relu activation layer, computes the activated neurons $a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$ by the ReLU function:

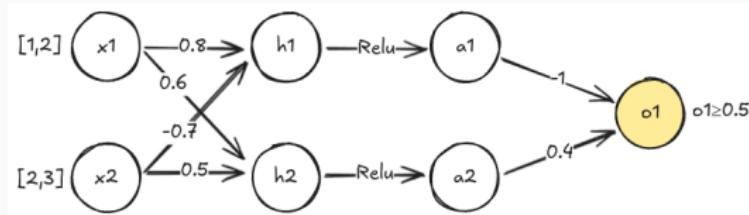
$$a_2 = \text{ReLU}(0, h_2) = \max(0, h_2), \text{ where } a_2 \in \mathbb{R}^+ \cup \{0\}, h_2 \in \mathbb{R}$$

DeepPoly bounds:

$$h_2 \mapsto \langle 0.6x_1 + 0.5x_2, 0.6x_1 + 0.5x_2, [1.6], [2.7] \rangle$$

$$\mathcal{I}[\![a_2 = \max(0, h_2)]\!]d = \{a_2 \mapsto \langle 0.6x_1 + 0.5x_2, 0.6x_1 + 0.5x_2, 1.6, 2.7 \rangle\}$$

Example - deeppoly domain



Last, the output layer computes the output neuron $o_1 \in \mathbb{R}$ by a linear combination:

$$o_1 = -a_1 + 0.4a_2, \text{ where } o_1 \in \mathbb{R}, a_1, a_2 \in \mathbb{R}^+ \cup \{0\}$$

DeepPoly bounds:

$$\mathcal{I}[o_1 = -a_1 + 0.4a_2]d = \{o_1 \mapsto \langle -a_1 + 0.4a_2, -a_1 + 0.4a_2, ?, ? \rangle\}$$

Example - deeppoly domain

Back-substitution:

$$o_1 \mapsto \langle -a_1 + 0.4a_2, -a_1 + 0.4a_2, ?, ? \rangle$$

$$\text{expr}_\ell = -a_1 + 0.4a_2$$

$$= -\frac{0.2}{1.5}(h_1 - (-1.3)) + 0.4(0.6x_1 + 0.5x_2)$$

$$= -\frac{0.2}{1.5}h_1 - \frac{0.26}{1.5} + 0.24x_1 + 0.2x_2$$

$$= -\frac{0.2}{1.5}(0.8x_1 - 0.7x_2) - \frac{0.26}{1.5} + 0.24x_1 + 0.2x_2$$

$$= -\frac{0.16}{1.5}x_1 + \frac{0.14}{1.5}x_2 - \frac{0.26}{1.5} + 0.24x_1 + 0.2x_2$$

$$= 0.1333x_1 + 0.2933x_2 - 0.1733$$

$$\ell = \min: \overleftarrow{s}(\text{expr}_\ell) = 0.54$$

Example - deeppoly domain

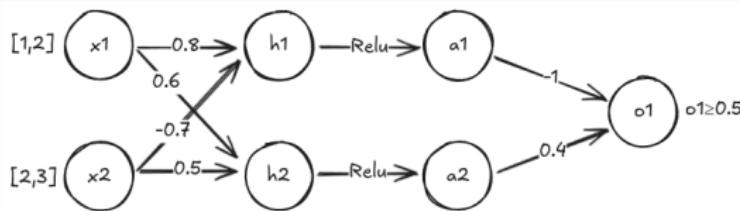
Back-substitution:

$$\begin{aligned}expr_u &= -a_1 + 0.4a_2 \\&= -0 + 0.4(0.6x_1 + 0.5x_2) \\&= 0.24x_1 + 0.2x_2 \\u = \max: \overleftarrow{s}(expr_u) &= 1.08\end{aligned}$$

We will have:

$$o_1 \mapsto \langle -a_1 + 0.4a_2, -a_1 + 0.4a_2, 0.54, 1.08 \rangle$$

Example - deepPoly domain



By using deeppoly domain, we obtain the interval bounds for each neuron in the network.

$$x_1 \in [1, 2], x_2 \in [2, 3],$$

$$h_1 \in [-1.3, 0.2], h_2 \in [1.6, 2.7],$$

$$a_1 \in [0, 0.2], a_2 \in [1.6, 2.7],$$

$$o_1 \in [0.54, 1.08]$$

Finally, we can conclude that the verification result is **UNSAT!**

Important note

Although we verify the property by DeepPoly, it does not mean that DeepPoly is always dominating other abstract domains.

You will see some examples that DeepPoly cannot prove UNSAT but SIP can in the second project .

Lattice Gym: <https://github.com/ytsao/lattice-gym/tree/main>

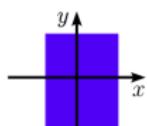
Other abstract domains

Course organisation

Course plan (2/8)

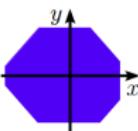
Bricks of abstraction: numerical domains

simple domains



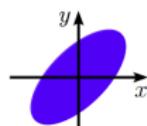
Intervals
 $x \in [a, b]$

relational domains



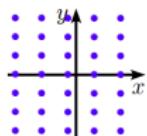
Octagons
 $\pm x \pm y \leq c$

specific domains



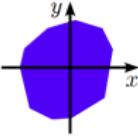
Ellipsoids
digital filters

Congruences



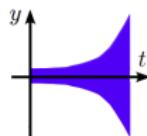
$x \in a\mathbb{Z} + b$

Polyhedra



$\sum_i \alpha_i x_i \leq \beta$

Exponentials



rounding errors

Other neural network architectures

- Layer types
 - 1. Convolutional
 - 2. Residual
 - 3. ...
- Activation functions
 - 1. Sigmoid
 - 2. Tahn
 - 3. ...
- Dataset
 - 1. MNIST
 - 2. CIFAR-10
 - 3. CIFAR-100
 - 4. ImageNet
 - 5. ...

Related conferences and competition

- The 37th International Conference on Computer Aided Verification (CAV'25)
- The 8th International Symposium on AI Verification (SAIV'25)
- The 6th International Verification of Neural Network Competition (VNN-COMP'25)

The standardized comparison:

- vnnlib: Define the pre/post-conditions.
- onnx: Store neural network structure.
- netron: Visualize network model.

Benchmarks in VNN-COMP 2024

Table 3: Overview of all scored benchmarks.

Category	Benchmark	Application	Network Types	# Params	Effective Input Dim	Track
Complex	cGAN	Image Generation & Image Prediction	Conv. + Vision Transformer	500k - 68M	5	regular
	NN4Sys	Dataset Indexing & Cardinality Prediction	ReLU + Sigmoid	33k - 37M	1-308	regular
	LinearizeNN	NN controller approximation	FC. + Conv. + Vision Transformer + Residual + ReLU	203k	4	regular
	ml4acopf	Power System	Complex (ReLU + Trigonometric + Sigmoid)	4k-680k	22 - 402	extended
	VIT	Vision	Conv. + Residual + Softmax + BatchNorm	68k - 76k	3072	extended
	Collins Aerospace	-	FC + Conv. + Residual, LeakyReLU + MaxPool + Square	1.8M	1.2M	extended
	LSNC	Lyapunov stability of NN controllers	FC + Residual, ReLU + Sin + Cos	210, 406	8	extended
CNN & ResNet	CCTSDB	-	FC + Conv. + Residual, ReLU + MaxPool + Clip	100k	2	extended
	Collins RUL CNN	Condition Based Maintenance	Conv. + ReLU, Dropout	60k - 262k	400 - 800	regular
	VGGNet16	Image Classification	Conv. + ReLU + MaxPool	138M	150k	extended
	Traffic Signs Recognition	Image Classification	Conv. + Sign + MakPool + BatchNorm	905k - 1.7M	2.7k - 12k	extended
	cifar100	Image Classification	FC + Conv. + Residual, ReLU + BatchNorm	2.5M - 3.8M	3072	regular
	tinyimagenet	Image Classification	FC + Conv. + Residual, ReLU + BatchNorm	3.6M	9408	regular
	Metaroom	-	Conv. + FC, ReLU	466k - 7.4M	5376	regular
FC	Yolo	-	FC + Conv. + Residual, ReLU + Sigmoid	22k - 37M	1 - 308	extended
	TLL Verify Bench	Two-Level Lattice NN	Two-Level Lattice NN (FC. + ReLU)	17k - 67M	2	regular
	Acas XU	Collision Detection	FC. + ReLU	13k	5	regular
	Dist Shift	Distribution Shift Detection	FC. + ReLU + Sigmoid	342k - 855k	792	regular
	safeNLP	Sentence classification	FC. + ReLU	4k	30	regular
	CORA	Image Classification	FC. + ReLU	575k, 1.1M	784, 3072	regular

Brix et al. (2024). The fifth international verification of neural networks competition (vnn-comp 2024).

Online Resources

- A Tutorial Website
- Stanford Seminar - Recent progress in verifying neural networks,
Zico Kolter
- AAAI 2022 Tutorial: "Formal Verification of Deep Neural Networks:
Theory and Practice"

References

- [1] Aws Albarghouthi. **Introduction to Neural Network Verification.** 2021.
- [2] Christopher Brix et al. **The Fourth International Verification of Neural Networks Competition (VNN-COMP 2023): Summary and Results.** 2023.
- [3] Timon Gehr et al. “**AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation**”. In: *2018 IEEE Symposium on Security and Privacy (SP)* Not available (May 2018), Not available. ISSN: Not available. DOI: 10.1109/sp.2018.00058. URL: <https://dx.doi.org/10.1109/sp.2018.00058>.

- [4] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. “**The Zonotope Abstract Domain Taylor1+**”. In: *Computer Aided Verification, Lecture Notes in Computer Science* Not available (undefined 2009), pp. 627–633. ISSN: 0302-9743, 1611-3349. DOI: 10.1007/978-3-642-02658-4_47. URL: https://dx.doi.org/10.1007/978-3-642-02658-4_47.
- [5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. **Explaining and Harnessing Adversarial Examples**. 2015.
- [6] Guy Katz et al. “**Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks**”. In: *Computer Aided Verification, Lecture Notes in Computer Science* Not available (undefined 2017), pp. 97–117. ISSN: 0302-9743, 1611-3349. DOI: 10.1007/978-3-319-63387-9_5. URL: https://dx.doi.org/10.1007/978-3-319-63387-9_5.

- [7] Changliu Liu et al. “**Algorithms for Verifying Deep Neural Networks**”. In: *CoRR* abs/1903.06758 (2019). URL: <http://arxiv.org/abs/1903.06758>.
- [8] Xavier Rival and Kwangkeun Yi. **Introduction to Static Analysis**. en. MIT Press, 2020, p. 315. ISBN: 9780262043410.
- [9] Gagandeep Singh et al. “**An abstract domain for certifying neural networks**”. In: *Proceedings of the ACM on Programming Languages* 3 (Jan. 2019), pp. 1–30. ISSN: 2475-1421. DOI: 10.1145/3290354. URL: <https://dx.doi.org/10.1145/3290354>.
- [10] Gagandeep Singh et al. “**Boosting Robustness Certification of Neural Networks**”. In: *International Conference on Learning Representations*. 2018. URL: <https://api.semanticscholar.org/CorpusID:196059499>.

- [11] Gagandeep Singh et al. “**Fast and Effective Robustness Certification**”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/f2f446980d8e971ef3da97af089481c3-Paper.pdf.
- [12] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. “**Evaluating Robustness of Neural Networks with Mixed Integer Programming**”. In: *International Conference on Learning Representations*. 2017. URL: <https://api.semanticscholar.org/CorpusID:47016770>.
- [13] Caterina Urban and Antoine Miné. “**A Review of Formal Methods applied to Machine Learning**”. In: *CoRR* abs/2104.02466 (2021). URL: <https://arxiv.org/abs/2104.02466>.
- [14] Shiqi Wang et al. “**Efficient formal safety analysis of neural networks**”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 6367–6377.

- [15] Huan Zhang et al. “**Efficient Neural Network Robustness Certification with General Activation Functions**”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL:
https://proceedings.neurips.cc/paper_files/paper/2018/file/d04863f100d59b3eb688a11f95b0ae60-Paper.pdf.
- [16] Huan Zhang et al. **General Cutting Planes for Bound-Propagation-Based Neural Network Verification**. 2022.