Abstraction-based methods for the Verification of Neural Networks with NeVer2

Dario Guidotti, Stefano Demarchi, Luca Pulina, Armando Tacchella

UNIVERSITÀ DEGLI STUDI DI SASSARI

Università di Genova

DIBRIS DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

Contents

Contents

Introduction

Context

The following slides explain the methodologies that are implemented in NeVer2
— our verification tool

Introduction

Context

The following slides explain the methodologies that are implemented in NeVer2 — our verification tool

Motivation

Verification of neural networks is a more and more important topic involving scalability and complexity problems

Introduction

Context

The following slides explain the methodologies that are implemented in NeVer2 — our verification tool

Motivation

Verification of neural networks is a more and more important topic involving scalability and complexity problems

Contribution

We present our methodology based on abstract interpretation for the verification of feed-forward neural networks with ReLU activation functions

Contents

Notation & definitions

Vectors

n-dimensional *vectors* of real numbers $x \in \mathbb{R}^n$ — also *points* or *samples* noted lowercase, e.g., x, y, z

Notation & definitions

Vectors

n-dimensional *vectors* of real numbers $x \in \mathbb{R}^n$ — also *points* or *samples* noted lowercase, e.g., x, y, z

Sets

Sets of vectors $X \subseteq \mathbb{R}^n$ noted uppercase, e.g., X, Y, Z

Notation & definitions

Vectors

n-dimensional *vectors* of real numbers $x \in \mathbb{R}^n$ — also *points* or *samples* noted lowercase, e.g., x, y, z

Sets

Sets of vectors $X \subseteq \mathbb{R}^n$ noted uppercase, e.g., X, Y, Z

Set properties

- A set X is *bounded* if there exists $r \in \mathbb{R}, r > 0$ such that $\forall x, y \in X \; ||x - y|| < r$
- A set X is *open* if for every point $x \in X$ there exist $\epsilon_x > 0$ such that $y \in \mathbb{R}^n$ belongs to X if $||x - y|| < \epsilon_x$

Notation & definitions

Set properties (seq.)

- The complement of an open set is *closed* — the one that includes its boundary. Closed and bounded sets are *compact*
- A set X is *convex* if for any $x, y \in X$ also $z \in X$ for each $z = (1 - \lambda)x + \lambda y$ with $\lambda \in [0, 1]$

Notation & definitions

Set properties (seq.)

- The complement of an open set is *closed* — the one that includes its boundary. Closed and bounded sets are *compact*
- A set X is *convex* if for any $x, y \in X$ also $z \in X$ for each $z = (1 - \lambda)x + \lambda y$ with $\lambda \in [0, 1]$

Convex hull

Given a non-empty set X, the smallest convex set $\mathcal{C}(X)$ containing X is the *convex hull* of X

Contents

Neural Networks

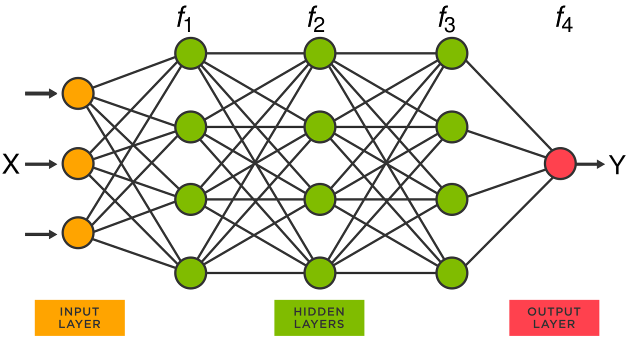
Feed-forward neural network

A function $\nu : \mathbb{R}^n \rightarrow \mathbb{R}^m$ obtained through the composition of p functions $f_1 : \mathbb{R}^n \rightarrow \mathbb{R}^{n_1}, ..., f_p : \mathbb{R}^{n_{p-1}} \rightarrow \mathbb{R}^m$ called *layers*

Neural Networks

Feed-forward neural network

A function $\nu : \mathbb{R}^n \rightarrow \mathbb{R}^m$ obtained through the composition of p functions $f_1 : \mathbb{R}^n \rightarrow \mathbb{R}^{n_1}, \ldots, f_p : \mathbb{R}^{n_{p-1}} \rightarrow \mathbb{R}^m$ called *layers*



$$Y = f_4(f_3(f_2(f_1(X))))$$

Neural Networks

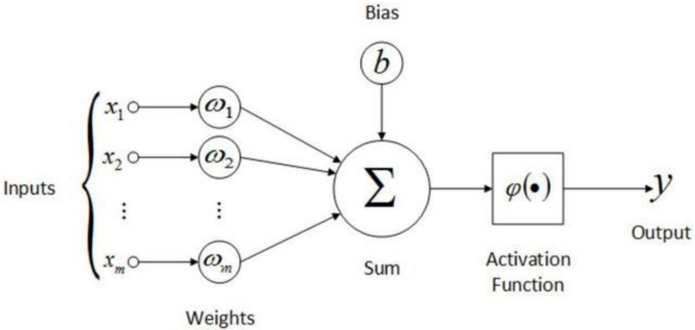
Layers

Generally, layers are a cascade of a weighted sum with the addition of a bias (linear *affine mapping*) and the application of an *activation function* $f(x) = (\varphi_1(x_1), ..., \varphi_m(x_m))$ with $\varphi_i : \mathbb{R}^m \to \mathbb{R}^m$

Neural Networks

Layers

Generally, layers are a cascade of a weighted sum with the addition of a bias (linear *affine mapping*) and the application of an *activation function* $f(x) = (\varphi_1(x_1), ..., \varphi_m(x_m))$ with $\varphi_i : \mathbb{R}^m \rightarrow \mathbb{R}^m$



Neural Networks

ReLU (Rectified Linear Unit) activation

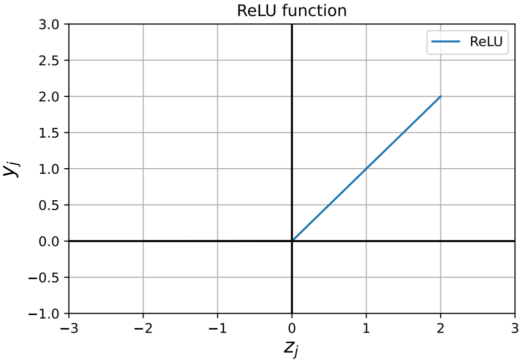
$ReLU(z_j) = max\{0, z_j\}$

Filters anything below zero, and leaves unchanged anything above

Neural Networks

ReLU (Rectified Linear Unit) activation

$ReLU(z_j) = max\{0, z_j\}$
Filters anything below zero, and leaves unchanged anything above



Neural Networks

Sigmoid (logistic) activation

$\sigma(z_j) = \frac{1}{(1+e^{-z_j})}$

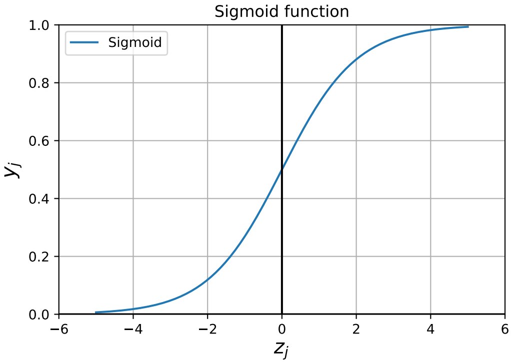
A continuous function bounded between 0 and 1

Neural Networks

Sigmoid (logistic) activation

$$\sigma(z_j) = \frac{1}{(1+e^{-z_j})}$$

A continuous function bounded between 0 and 1



Neural Networks

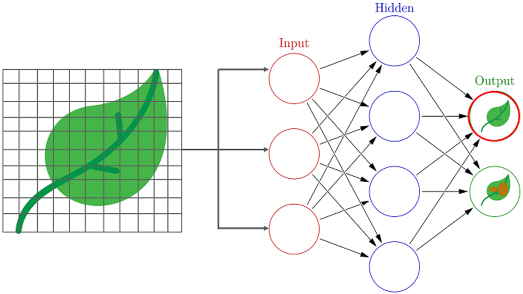
Application

The *classification* task assigns every input vector $z \in \mathbb{R}^n$ one out of m labels. The *regression* task approximates a functional mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$

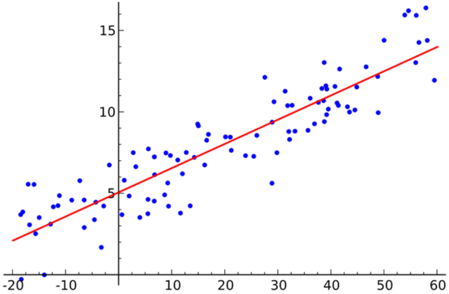
Neural Networks

Application

The *classification* task assigns every input vector $z \in \mathbb{R}^n$ one out of m labels. The *regression* task approximates a functional mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$



Classification



Regression

Verification

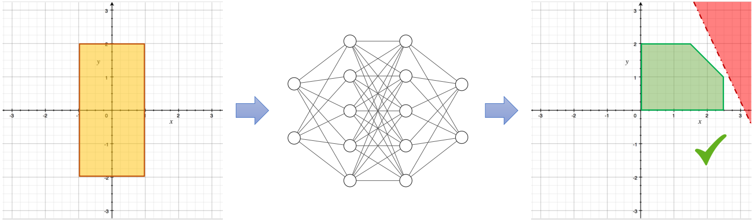
Objective

Verify algorithmically that the NN complies to some *post-conditions* on the outputs if some *pre-conditions* on the input are met

Verification

Objective

Verify algorithmically that the NN complies to some *post-conditions* on the outputs if some *pre-conditions* on the input are met



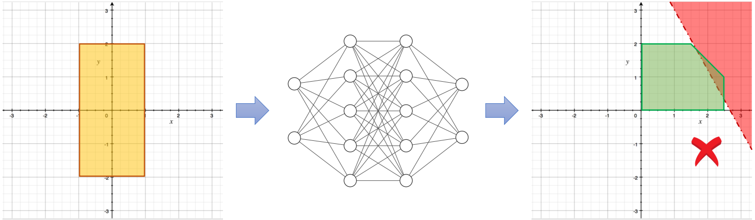
Input bound (yellow)

Output (green)
Unsafe zone(red)

Verification

Objective

Verify algorithmically that the NN complies to some *post-conditions* on the outputs if some *pre-conditions* on the input are met



Input bound (yellow)

Output (green)
Unsafe zone(red)

Verification

Assumptions

- Input $I \subset \mathbb{R}^n$ is bounded
- Output $O \subset \mathbb{R}^m$ is also bounded
  - Affine transformations of bounded sets are bounded sets
  - $f(z) = ReLU(z)$ is piecewise affine
  - $f(z) = \sigma(z)$ bounds the output on $[0, 1]$

Verification

Assumptions

- Input $I \subset \mathbb{R}^n$ is bounded
- Output $O \subset \mathbb{R}^m$ is also bounded
  - Affine transformations of bounded sets are bounded sets
  - $f(z) = ReLU(z)$ is piecewise affine
  - $f(z) = \sigma(z)$ bounds the output on $[0, 1]$

Conclusion

Pre and *post*-conditions represent *n*-dimensional space regions in which inputs and outputs should be contained

Contents

Abstract methods

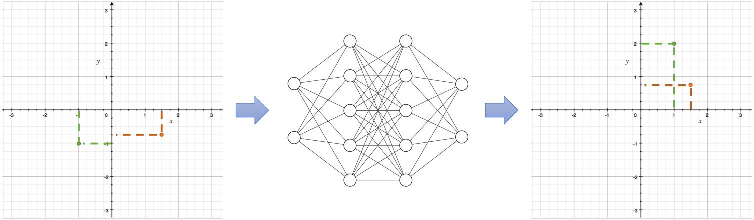
Abstract domains

We abstract the network behavior by considering the transformation on the input domain, considering the abstract domain $\langle \mathbb{R}^n \rangle \subset 2^{\mathbb{R}^n}$ of polytopes in $\mathbb{R}^n$ to abstract bounded sets into polytopes

Abstract methods

Abstract domains

We abstract the network behavior by considering the transformation on the input domain, considering the abstract domain $\langle \mathbb{R}^n \rangle \subset 2^{\mathbb{R}^n}$ of polytopes in $\mathbb{R}^n$ to abstract bounded sets into polytopes

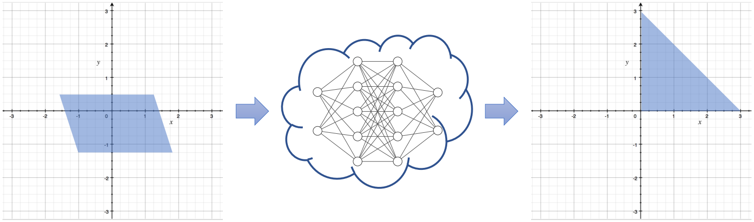


A concrete network operates a point-to-point mapping

Abstract methods

Abstract domains

We abstract the network behavior by considering the transformation on the input domain, considering the abstract domain $\langle \mathbb{R}^n \rangle \subset 2^{\mathbb{R}^n}$ of polytopes in $\mathbb{R}^n$ to abstract bounded sets into polytopes



An abstract network maps space regions

Abstraction definitions

Abstraction

Given a bounded set $X \subset \mathbb{R}^n$ an abstraction is a function $\alpha : 2^{\mathbb{R}^n} \rightarrow \langle \mathbb{R}^n \rangle$ that maps X to a polytope P such that $\mathcal{C}(X) \subseteq P$

Abstraction definitions

Abstraction

Given a bounded set $X \subset \mathbb{R}^n$ an abstraction is a function $\alpha : 2^{\mathbb{R}^n} \rightarrow \langle \mathbb{R}^n \rangle$ that maps X to a polytope P such that $\mathcal{C}(X) \subseteq P$

Concretization

Given a polytope $P \in \langle \mathbb{R}^n \rangle$ a concretization is a function $\gamma : \langle \mathbb{R}^n \rangle \rightarrow 2^{\mathbb{R}^n}$ that maps P to the set of points contained in it

Abstraction definitions

Abstraction

Given a bounded set $X \subset \mathbb{R}^n$ an abstraction is a function $\alpha : 2^{\mathbb{R}^n} \rightarrow \langle \mathbb{R}^n \rangle$ that maps X to a polytope P such that $\mathcal{C}(X) \subseteq P$

Concretization

Given a polytope $P \in \langle \mathbb{R}^n \rangle$ a concretization is a function $\gamma : \langle \mathbb{R}^n \rangle \rightarrow 2^{\mathbb{R}^n}$ that maps P to the set of points contained in it

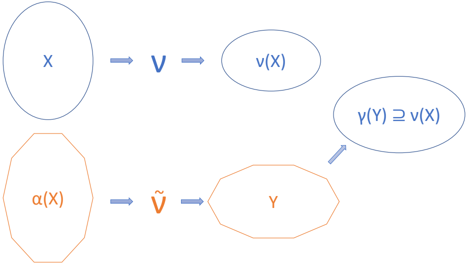
Consistent abstraction

Given a mapping $\nu : \mathbb{R}^n \rightarrow \mathbb{R}^m$, a mapping $\tilde{\nu} : \langle \mathbb{R}^n \rangle \rightarrow \langle \mathbb{R}^m \rangle$, an abstraction α and a concretization γ, $\tilde{\nu}$ is a consistent abstraction of ν over an input set X exactly when $\{\nu(x) | x \in X\} \subseteq \gamma(\tilde{\nu}(\alpha(X)))$

Abstraction definitions

Consistent abstraction (example)

Given a mapping $\nu : \mathbb{R}^n \to \mathbb{R}^m$, a mapping $\tilde{\nu} : \langle\mathbb{R}^n\rangle \to \langle\mathbb{R}^m\rangle$, an abstraction α and a concretization γ, $\tilde{\nu}$ is a consistent abstraction of ν over an input set X exactly when $\{\nu(x) | x \in X\} \subseteq \gamma(\tilde{\nu}(\alpha(X)))$



Example

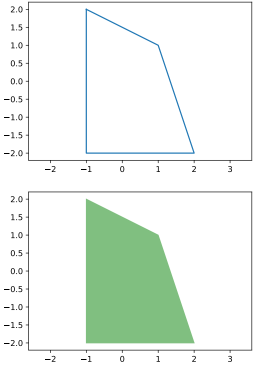
Convex approximation

There are different cases for the input set $X \subset \mathbb{R}$

Example

Convex approximation

There are different cases for the input set $X \subset \mathbb{R}$

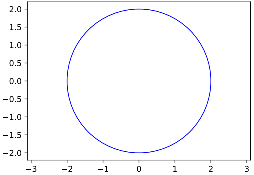


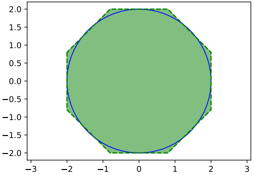
X is already convex and matches the enclosing polytope

Example

Convex approximation

There are different cases for the input set $X \subset \mathbb{R}$



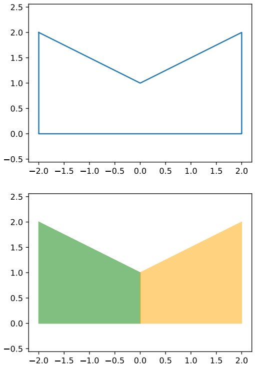


X is convex but not linear, and is over-approximated by an octagon

Example

Convex approximation

There are different cases for the input set $X \subset \mathbb{R}$



X is not convex, therefore is divided in two polytopes such that their union matches the original one

Abstract representation

Generalized star set

Given a *basis matrix* $V \in \mathbb{R}^{n \times m}$, a point $c \in \mathbb{R}^n$ called *center*, and a *predicate* $R : \mathbb{R}^m \to \{\top, \bot\}$, a generalized star set (or just *star*) is a tuple $\Theta = (c, V, R)$.
The set of points represented by the star is

$$[\![\Theta]\!] \equiv \{z \in \mathbb{R}^n \mid z = Vx + c \text{ such that } R(x_1, \ldots, x_m) = \top\}$$

Abstract representation

Generalized star set

Given a *basis matrix* $V \in \mathbb{R}^{n \times m}$, a point $c \in \mathbb{R}^n$ called *center*, and a *predicate* $R : \mathbb{R}^m \to \{\top, \bot\}$, a generalized star set (or just *star*) is a tuple $\Theta = (c, V, R)$.
The set of points represented by the star is

$$[\![\Theta]\!] \equiv \{z \in \mathbb{R}^n \mid z = Vx + c \text{ such that } R(x_1, \ldots, x_m) = \top\}$$

In practice

We consider stars where the predicate represents a polytope, i.e., $R : Cx \leq d$.
This polytope in x depends on the basis matrix V

Abstract representation

Star

$$[\![\Theta]\!] \equiv \{z \in \mathbb{R}^2 \mid z = Vx + c \text{ such that } Cx \leq d\}$$

$$\text{Let } x = \{x_0, x_1\} \text{ and } Cx \leq d = \begin{cases} x_0 \leq 1 \\ -x_0 \leq 1 \\ x_1 \leq 1 \\ -x_1 \leq 1 \end{cases}$$

Abstract representation

Star

$$[\![\Theta]\!] \equiv \{z \in \mathbb{R}^2 \mid z = Vx + c \text{ such that } Cx \le d\}$$

$$\text{Let } x = \{x_0, x_1\} \text{ and } Cx \le d = \begin{cases} x_0 \le 1 \\ -x_0 \le 1 \\ x_1 \le 1 \\ -x_1 \le 1 \end{cases}$$

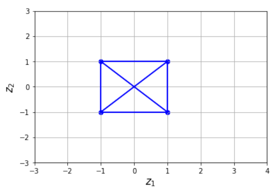
Predicate

$$\text{Predicate matrix } C = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \text{ and bias } d = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$$

Abstract representation

Identity basis

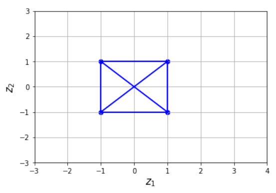
Basis matrix $V_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, center $c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$



Abstract representation

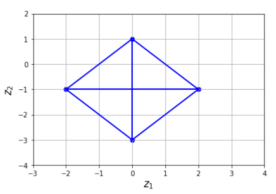
Identity basis

Basis matrix $V_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, center $c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$



Different basis

Basis matrix $V_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, center $c = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$



Activation functions abstraction

How to propagate the abstraction?

The different layers of a NN transform the abstract star

- Linear (*Fully Connected*) layers perform affine transformations
- Activation layers?

Activation functions abstraction

How to propagate the abstraction?

The different layers of a NN transform the abstract star

- Linear (*Fully Connected*) layers perform affine transformations
- Activation layers?

ReLU

Given the ReLU piecewise nature, we can either split the computation between the negative part and the positive part — for each dimension! — or compute an abstract over-approximation

Activation functions abstraction

How to propagate the abstraction?

The different layers of a NN transform the abstract star

- Linear (*Fully Connected*) layers perform affine transformations
- Activation layers?

ReLU

Given the ReLU piecewise nature, we can either split the computation between the negative part and the positive part — for each dimension! — or compute an abstract over-approximation

Sigmoid

On the other hand, the logistic function can only be over-approximated by an appropriate abstraction

Contents

Running example

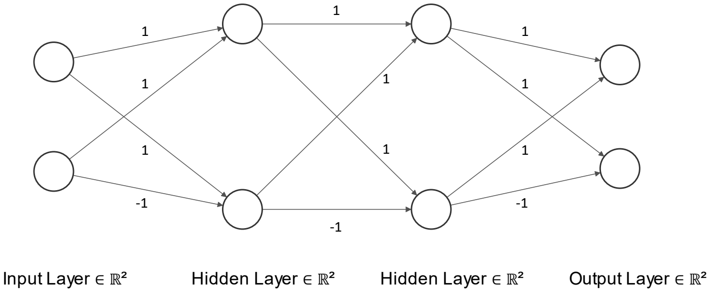
A simple 2D NN

We consider a simple bi-dimensional Fully Connected/ReLU network with two hidden layers of two neurons each

Running example

A simple 2D NN

We consider a simple bi-dimensional Fully Connected/ReLU network with two hidden layers of two neurons each



Running example

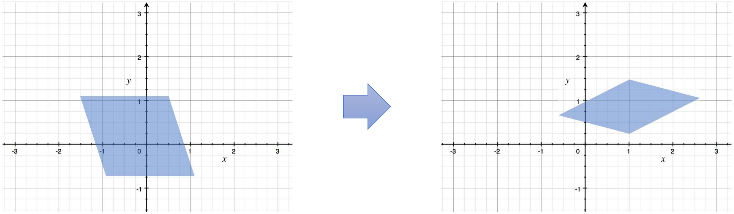
Fully Connected behavior

The linear (fully connected) layers transform the input set by means of shift, rotation and scale operations

Running example

Fully Connected behavior

The linear (fully connected) layers transform the input set by means of shift, rotation and scale operations



Running example

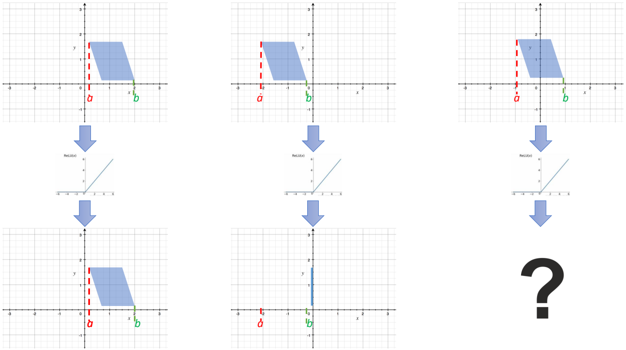
ReLU behavior

Given the properties of the input set the ReLU behaves differently: if $a \geq 0$, i.e., the set is all positive then the ReLU is the identity function, whereas if $b \leq 0$ the input is zeroed; when $a < 0 < b$ we can either split the results or over-approximate

Running example

ReLU behavior

Given the properties of the input set the ReLU behaves differently: if $a \geq 0$, i.e., the set is all positive then the ReLU is the identity function, whereas if $b \leq 0$ the input is zeroed; when $a < 0 < b$ we can either split the results or over-approximate



Running example

Bounds evaluation

The lower and upper bounds ub_j, lb_j of the star along its j-th dimension are computed with a linear-programming problem

$$
\begin{aligned}
&max/min\{z_j = \mathbf{V}[j,:]\mathbf{x} + c[j]\} && \text{(star basis)} \\
&s.t. && \\
&\mathbf{Cx} \leq \mathbf{d} && \text{(star predicates)}
\end{aligned}
$$

Running example

Bounds evaluation

The lower and upper bounds ub_j, lb_j of the star along its j-th dimension are computed with a linear-programming problem

$$
\begin{array}{ll}
max/min\{z_j = \mathbf{V}[j,:]\mathbf{x} + c[j]\} & \text{(star basis)} \\
s.t. & \\
\mathbf{Cx} \leq \mathbf{d} & \text{(star predicates)}
\end{array}
$$

Scalability issues

The linear programming problem scales with the dimension of the layers $\rightarrow$ bottleneck computing the bounds of large scale networks. In addition, abstracting ReLU layers introduces further parameters

Running example

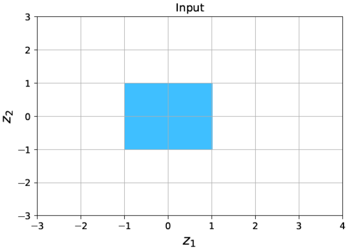
Abstract input

Let us consider again the star presented before as the network input for our example

Running example

Abstract input

Let us consider again the star presented before as the network input for our example



(a) Graphical Representation as Convex Hull

$$Input \equiv \{z \in \mathbb{R}^n \mid z = \mathbb{I}x \text{ such that } Cx \leq d\}$$

$$\begin{aligned} where\ Cx \leq d\ is\ x_0 &\leq 1 \\ -x_0 &\leq 1 \\ x_1 &\leq 1 \\ -x_1 &\leq 1 \end{aligned}$$

(b) Numerical Representation of the input starset.

Affine transformation

Layer 1 - Fully Connected

The first affine transformation with $W = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ transforms the input star by affecting only the basis matrix

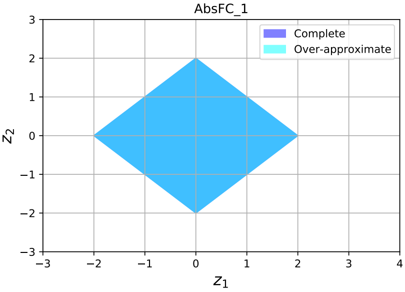
Affine transformation

Layer 1 - Fully Connected

The first affine transformation with $W = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ transforms the input star by affecting only the basis matrix

$$\hat{V}_{FC} = WV = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\hat{c}_{FC} = Wc + b = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



ReLU: exact computation

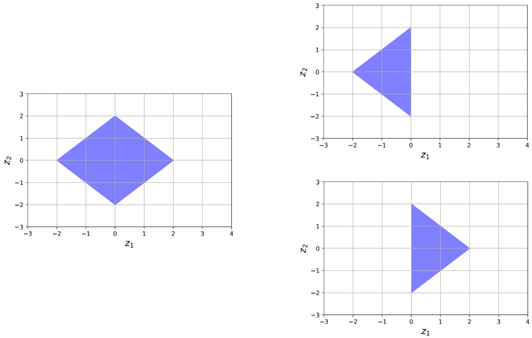
ReLU abstraction

The fine abstraction of the ReLU function is non-linear, so whenever $lb_j < 0 < ub_j$ the network splits as if the inputs were two: one less than 0 and another greater than 0

ReLU: exact computation

ReLU abstraction

The fine abstraction of the ReLU function is non-linear, so whenever $lb_j < 0 < ub_j$ the network splits as if the inputs were two: one less than 0 and another greater than 0



ReLU: exact computation

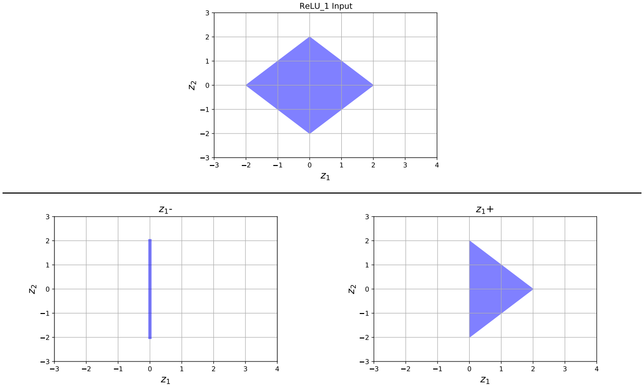
Split along z_1

Along the z_1 axis the input polytope collapses on the negative part and remains unchanged on the positive one

ReLU: exact computation

Split along z_1

Along the z_1 axis the input polytope collapses on the negative part and remains unchanged on the positive one



ReLU: exact computation

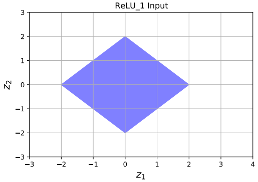
Split along z_1

This corresponds to imposing $z_1 \leq 0$ in the predicates and $z_1 = 0$ in the basis for the negative star, and $z_1 \geq 0$ in the predicates for the positive one

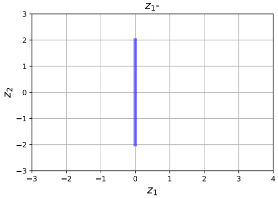
ReLU: exact computation

Split along z_1

This corresponds to imposing $z_1 \leq 0$ in the predicates and $z_1 = 0$ in the basis for the negative star, and $z_1 \geq 0$ in the predicates for the positive one



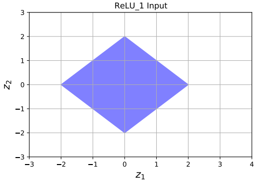
$$\begin{cases} z_1 \leq 0 \rightarrow x_0 + x_1 \leq 0 \\ V_{z_1^-} = \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix} \end{cases}$$



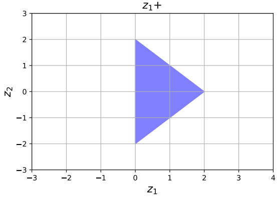
ReLU: exact computation

Split along z_1

This corresponds to imposing $z_1 \leq 0$ in the predicates and $z_1 = 0$ in the basis for the negative star, and $z_1 \geq 0$ in the predicates for the positive one



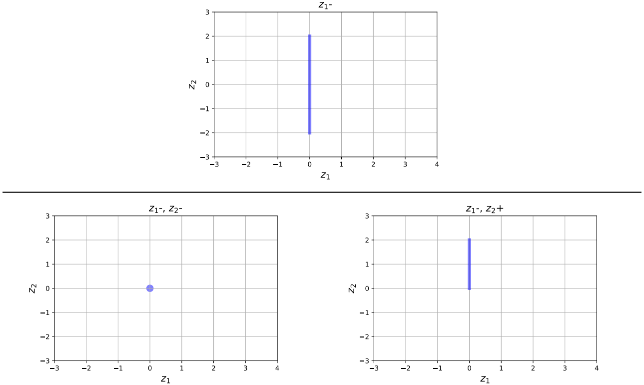
$$\begin{cases} z_1 \geq 0 \rightarrow -x_0 - x_1 \leq 0 \\ V_{z_1^+} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \end{cases}$$



ReLU: exact computation

Split along z_2 (1)

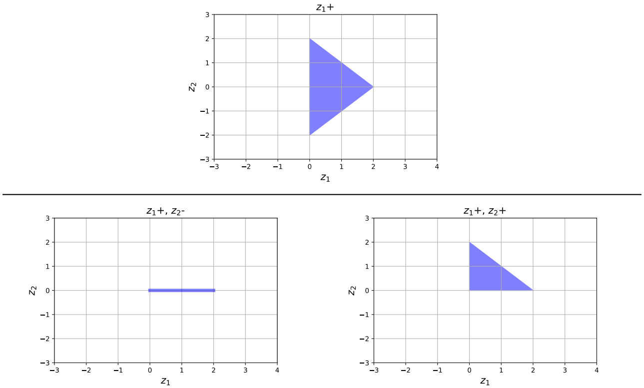
Along the z_2 axis the result of the previous split either becomes a single point for the negative part (z_1-, z_2-) or is cut (z_1-, z_2+)



ReLU: exact computation

Split along z_2 (2)

The same operation is performed on the positive output of the first split, with only z_1+, z_2+ resulting in an actual polytope



ReLU: exact computation

Starset growth

The number of stars produced by the exact computation is exponential in the worst case, where each and every ReLU has to split

ReLU: exact computation

Starset growth

The number of stars produced by the exact computation is exponential in the worst case, where each and every ReLU has to split

Predicates growth

Each new star adds a constraint (row) in the predicates matrix and the bias vector

ReLU: exact computation

Starset growth

The number of stars produced by the exact computation is exponential in the worst case, where each and every ReLU has to split

Predicates growth

Each new star adds a constraint (row) in the predicates matrix and the bias vector

Degenerate stars

Even if the exact computation produced 4 stars out of one, three of them are degenerate, i.e., they don't carry useful information. To prove whether a star is degenerate or not could improve the algorithm, but is a difficult task

ReLU: approximation

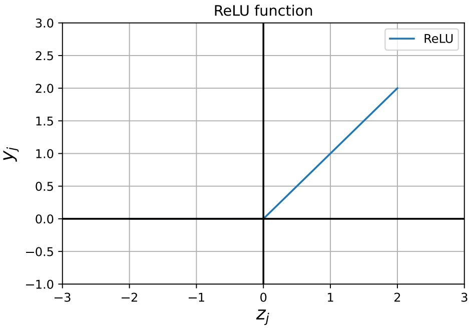
Neuron-level behaviour

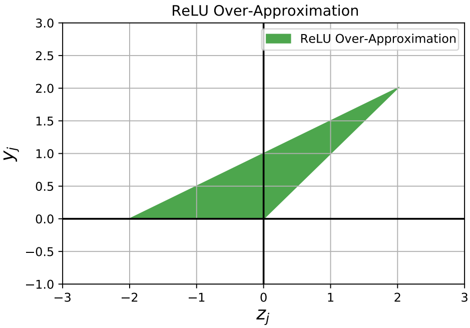
We provide a coarse abstraction of the ReLU function by picturing a triangle limited by the star bounds. This approximation — minimal area — enforces a smaller error w.r.t. other techniques such as zonotopes and abstract domains

ReLU: approximation

Neuron-level behaviour

We provide a coarse abstraction of the ReLU function by picturing a triangle limited by the star bounds. This approximation — minimal area — enforces a smaller error w.r.t. other techniques such as zonotopes and abstract domains

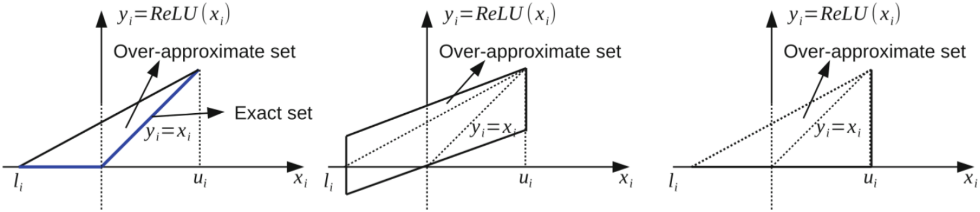




ReLU: approximation

Neuron-level behaviour

We provide a coarse abstraction of the ReLU function by picturing a triangle limited by the star bounds. This approximation — minimal area — enforces a smaller error w.r.t. other techniques such as zonotopes and abstract domains



Over-approximation with star — Over-approximation with zonotope — Over-approximation with abstract-domain

ReLU: approximation

Star POV

From the Star point of view, we define an auxiliary variable x_{m+1} in order to express the three constraints of the triangle in terms of the predicate variables

ReLU: approximation

Star POV

From the Star point of view, we define an auxiliary variable x_{m+1} in order to express the three constraints of the triangle in terms of the predicate variables

$$
\begin{aligned}
x_{m+1} &\geq 0 \\
x_{m+1} &\geq V_j \mathbf{x} + c_j \\
x_{m+1} &\leq ub_j \cdot \frac{V_j \mathbf{x} + c_j - lb_j}{ub_j - lb_j}
\end{aligned}
$$

If we reorder these constraints we can bring them in the format $C\mathbf{x} \leq \mathbf{d}$:

$$
\begin{gathered}
-x_{m+1} \leq 0 \\
V_j \mathbf{x} - x_{m+1} \leq -c_j \\
-\frac{ub_j}{ub_j - lb_j} V_j \mathbf{x} + x_{m+1} \leq \frac{ub_j}{ub_j - lb_j}(c_j - lb_j)
\end{gathered}
$$

ReLU: approximation

Star computation

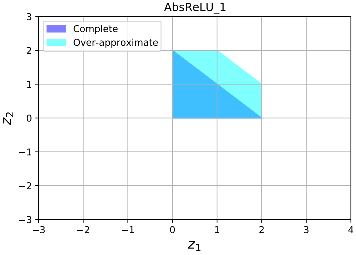
The ReLU over-approximation introduces two variables and 6 constraints in the predicate matrix, requiring to solve a total of 4 LPs

ReLU: approximation

Star computation

The ReLU over-approximation introduces two variables and 6 constraints in the predicate matrix, requiring to solve a total of 4 LPs

$$\hat{C}_{ReLU} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{-1} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{1} & 0 & 0 \\ \mathbf{0} & \mathbf{-1} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & -1 & 0 \\ -0.5 & -0.5 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & -1 \\ -0.5 & 0.5 & 0 & 1 \end{bmatrix} \quad \hat{d}_{ReLU} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



ReLU: approximation

Starset growth

The approximation propagates always a single star in the whole network. On the other hand, the price is paid in the outer approximation

ReLU: approximation

Starset growth

The approximation propagates always a single star in the whole network. On the other hand, the price is paid in the outer approximation

Predicates growth

Each neuron adds 3 constraints (rows) in the predicates matrix and the bias vector, as well as one extra variable. This impacts heavily on the LPs for bounds computation when the number of neurons is huge

ReLU: approximation

Starset growth

The approximation propagates always a single star in the whole network. On the other hand, the price is paid in the outer approximation

Predicates growth

Each neuron adds 3 constraints (rows) in the predicates matrix and the bias vector, as well as one extra variable. This impacts heavily on the LPs for bounds computation when the number of neurons is huge

Approximation benefits

The complexity introduced by the approximation grows slower than the exact approach. The major benefit is that the approximate method guarantees a sound verification with the minimal area overhead