# Reliable and Interpretable Artificial Intelligence

Lecture 4b: Mathematical Certification of Neural Networks

Martin Vechev
ETH Zurich

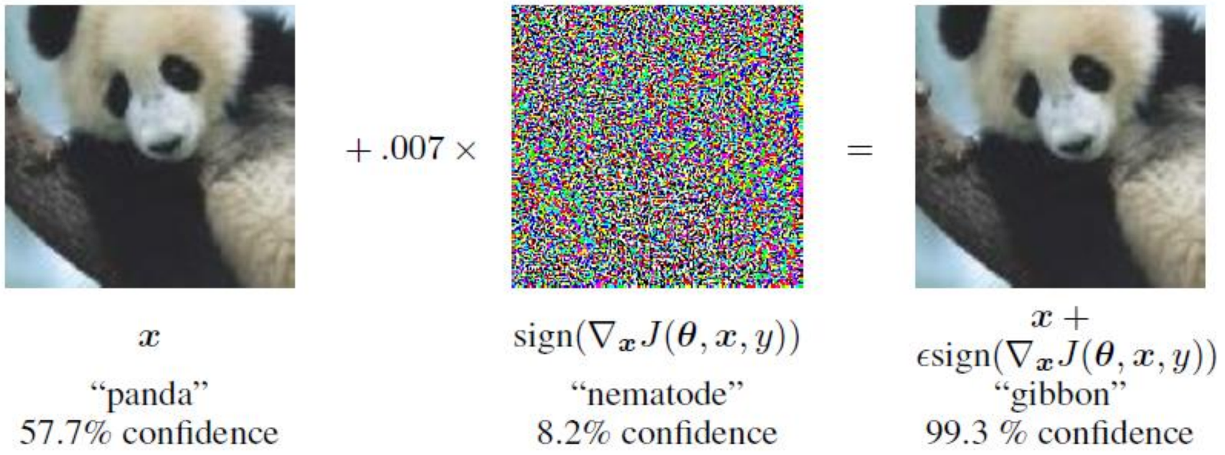
Fall 2020

http://www.sri.inf.ethz.ch

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

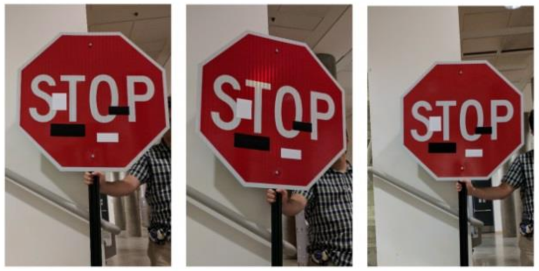
# Adversarial Examples: Attacks and Defenses

## Noisy attack: vision system thinks we now have a gibbon…



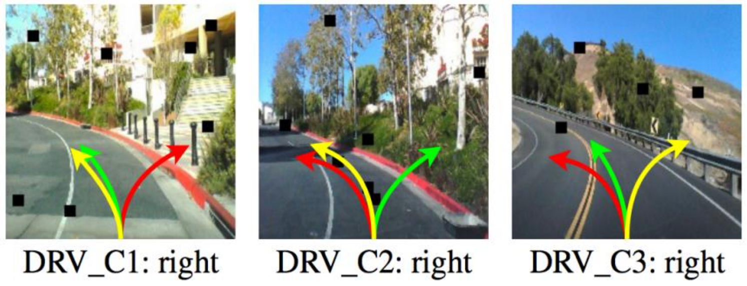
Explaining and Harnessing Adversarial Examples, ICLR '15

## Tape pieces make network predict a 45mph sign



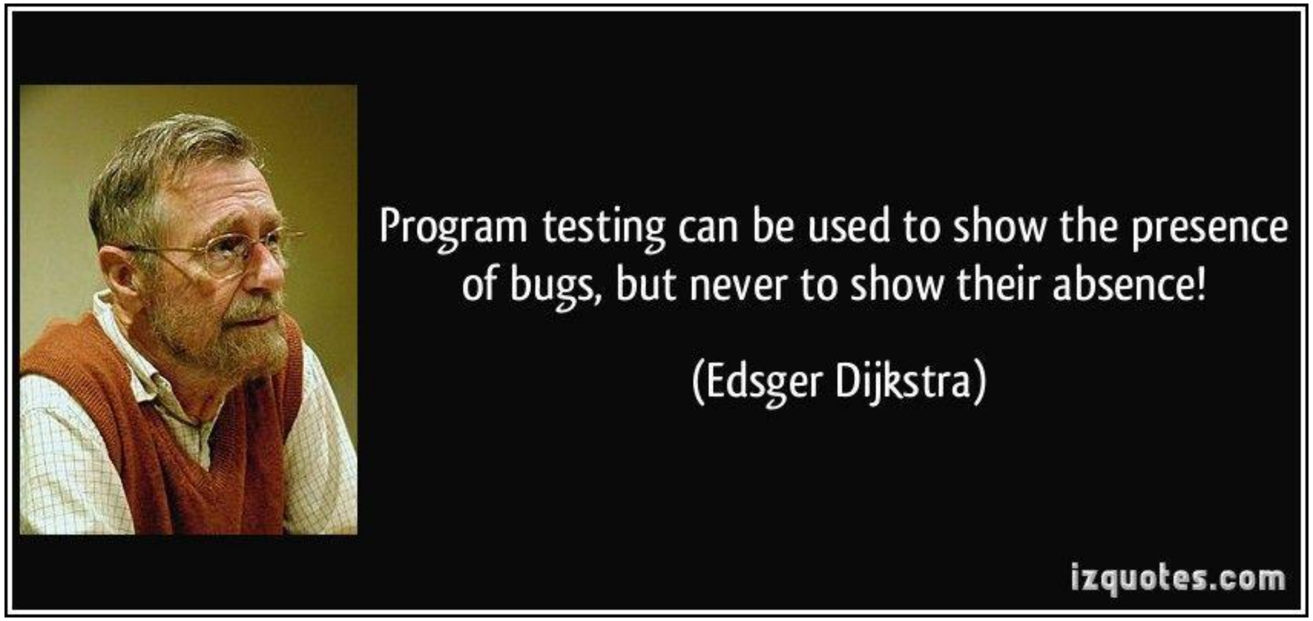
Robust Physical-World Attacks on Deep Learning Visual Classification, CVPR'18

## Self-driving car: in each picture one of the 3 networks makes a mistake…



DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP'17

# The fundamental problem


Program testing can be used to show the presence of bugs, but never to show their absence!

(Edsger Dijkstra)

izquotes.com

The attacks and defenses so far are similar to testing: they may work well in practice sometimes, but provide no formal guarantees.

## More generally we want

**Automated** verifier to prove properties of realistic networks

**Useful in:**

- Certifying large cyber-physical systems that use NN
- Proving robustness of NN
- Learning interpretable specs of the NN
- Comparing NNs

# Problem Statement

Given

- a **neural network** $\boldsymbol{N}$
- a **property over inputs** $\boldsymbol{\phi}$ [called: pre-condition]
- a **property over outputs** $\boldsymbol{\psi}$ [called: post-condition]

Prove that $\forall \boldsymbol{i} \in \boldsymbol{I}.\, \boldsymbol{i} \vDash \boldsymbol{\phi} \Longrightarrow \boldsymbol{N}(\boldsymbol{i}) \vDash \boldsymbol{\psi}$ holds or return a violation

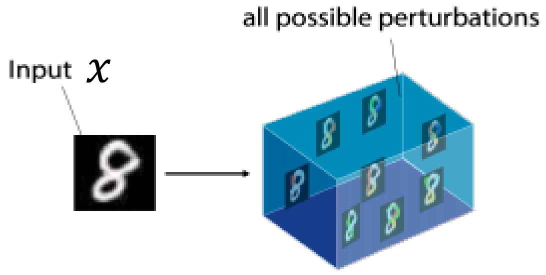
Instantiating the problem to certifying robustness of a character classification neural network, it would look as follows

# Step 1: Define $\boldsymbol{\phi}$ Formally

With robustness, its a region captured by $\boldsymbol{\phi}$:

Example $\boldsymbol{\phi}$:

$L_\infty$ ball: $\text{Ball}(x)_\epsilon = \{\, x' \mid \; || x - x' ||_\infty < \epsilon \,\}$



# Step 1: Define $\phi$ Formally

**Certification**

Region $\phi$:

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

Example: here, we capture **brightening attack** (only few pixels have $\epsilon$ )

# Step 2: Verify $\boldsymbol{\phi}$ satisfies $\boldsymbol{\psi}$

Certification

We need to prove this property:

$$\forall \boldsymbol{i} \in \boldsymbol{I}.\, \boldsymbol{i} \vDash \boldsymbol{\phi} \Longrightarrow \boldsymbol{N}(\boldsymbol{i}) \vDash \boldsymbol{\psi}$$

Region $\boldsymbol{\phi}$:

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

Property $\boldsymbol{\psi}$ : every point classifies to 3:

$$\boldsymbol{\psi} : \quad \forall j \in [0,9].\, o_3 \geq o_j$$

Here, $o$ is the output of the layer before the last softmax layer

# Challenge

Certification

We need to prove this property:

$$\forall \boldsymbol{i} \in \boldsymbol{I}. \boldsymbol{i} \vDash \boldsymbol{\phi} \Longrightarrow \boldsymbol{N}(\boldsymbol{i}) \vDash \boldsymbol{\psi}$$

How to prove this property?

Challenge: as $\phi$ can capture an unbounded set of points, one cannot simply enumerate points in $\phi$ and check each satisfies $\psi$

Region $\boldsymbol{\phi}$:

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

Property $\boldsymbol{\psi}$ : every point classifies to 3:

$$\boldsymbol{\psi}: \quad \forall j \in [0,9]. o_3 \geq o_j$$

Here, $o$ is the output of the layer before the last softmax layer

# Certification Methods: Sound vs. Unsound

A reasoning method is called **sound** if when a program violates a property, when the method terminates, the method always states the property is violated. We typically refer to these methods as **certification methods**.

A reasoning method is called **unsound** if when the program violates a property, the method could potentially terminate stating the property is satisfied. For example, adversarial attack methods are typically unsound: they may miss violations when they exist.

## Certification Methods: Complete vs. Incomplete

A  certification method is called **complete** if it is able to prove the property holds when it actually holds.

A certification method is called **incomplete** if it cannot guarantee that it can prove a property which actually holds.

# Soundness vs. Completeness vs. Automation

| Sound | Complete | Automated Algorithm |
|---|---|---|
| Yes | Yes | Discussed in next slides |
| Yes | No | Return "No, property does not hold" |
| No | Yes | Return "Yes, property always holds" |
| No | No | Random Guessing |

For certification we want:

- Soundness
- Scalable Algorithm
- Precision: ``as complete as possible''

Due to Rice's theorem, generally, it is not possible to construct an automated method that is both sound and complete. However, for some restricted types of computations, like neural networks (which are generally loop-free), it is possible to be both, sound and complete (but these may not scale to realistic networks).

# Certification of Neural Networks

We will cover two kinds of sound methods:

**Incomplete** but **scalable** methods: these include convex relaxations such as Box, Zonotope and DeepPoly.

**Complete** but **not scalable** methods: Mixed-Integer Linear Solvers (MILP) and Linear Solvers (LP).

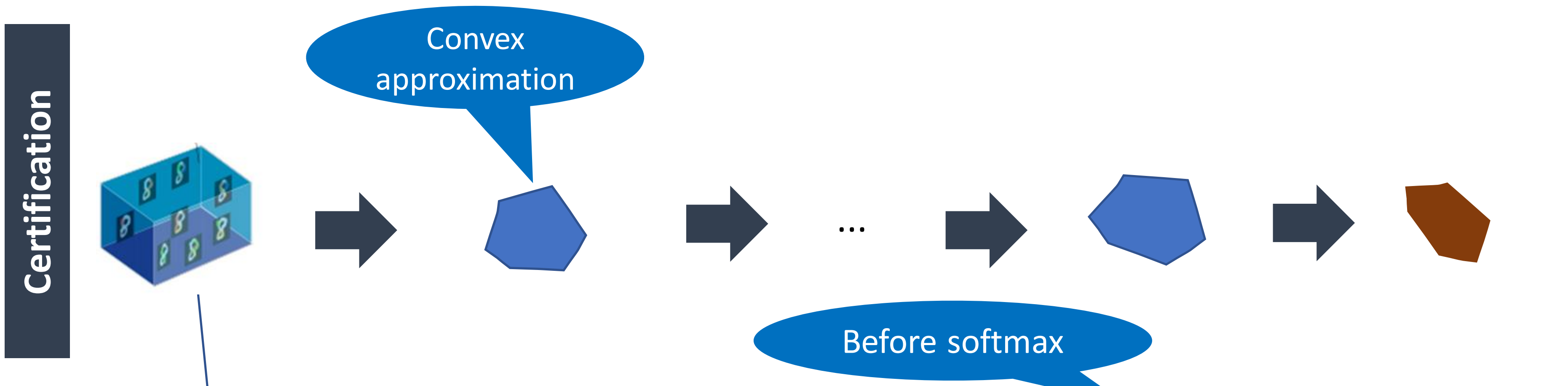
**Combinations** of both: goal is to either *scale complete methods* or to make *incomplete methods more precise*.

# Incomplete methods

We will investigate a specific type of incomplete method, based on **bound propagation** through the neural network. Starting with the initial pre-condition $\boldsymbol{\phi}$ , we will ``push'' $\boldsymbol{\phi}$ through the network, computing an **over-approximation** of the effect of each layer.

Lets see how we would prove our example property with this method

# Step 1: Compute bounds by propagating $\boldsymbol{\phi}$



Region $\boldsymbol{\phi}$:

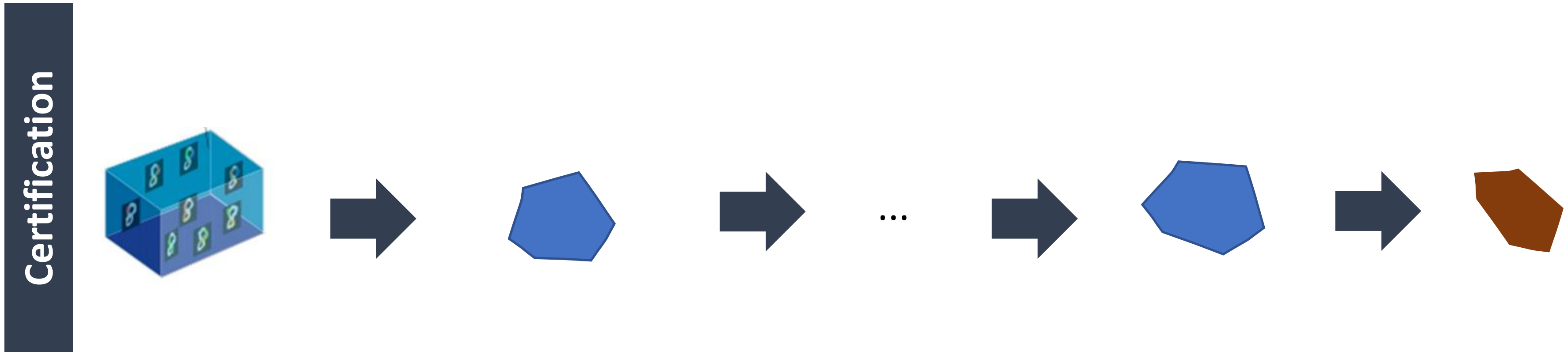
$x_0 = 0$
$x_1 = 0.975 + 0.025\epsilon_1$
$x_2 = 0.125$
…
$x_{784} = 0.938 + 0.062\epsilon_{784}$
$\forall i. \epsilon_i \in [0,1]$

Output region $\phi_n$

$o_0 = 0$
$o_1 = 2.60 + 0.015\epsilon_0 + 0.023\epsilon_1 + 5.181\epsilon_2 + \cdots$
$o_2 = 4.63 - 0.005\epsilon_0 - 0.006\epsilon_1 + 0.023\epsilon_2 + \cdots$
…
$o_9 = 0.12 - 0.125\epsilon_0 + 0.102\epsilon_1 + 3.012\epsilon_2 + \cdots$
$\forall i. \epsilon_i \in [0,1]$

Over-approximation $\phi_n$ means it is possible there are concrete points inside $\phi_n$ which cannot be produced by any concrete point inside $\boldsymbol{\phi}$.

# Step 2: Certify the property



Output region $\phi_n$

$o_0 = 0$
$o_1 = 2.60 + 0.015\epsilon_0 + 0.023\epsilon_1 + 5.181\epsilon_2 + \cdots$
$o_2 = 4.63 - 0.005\epsilon_0 - 0.006\epsilon_1 + 0.023\epsilon_2 + \cdots$
…
$o_9 = 0.12 - 0.125\epsilon_0 + 0.102\epsilon_1 + 3.012\epsilon_2 + \cdots$
$\forall i. \epsilon_i \in [0,1]$

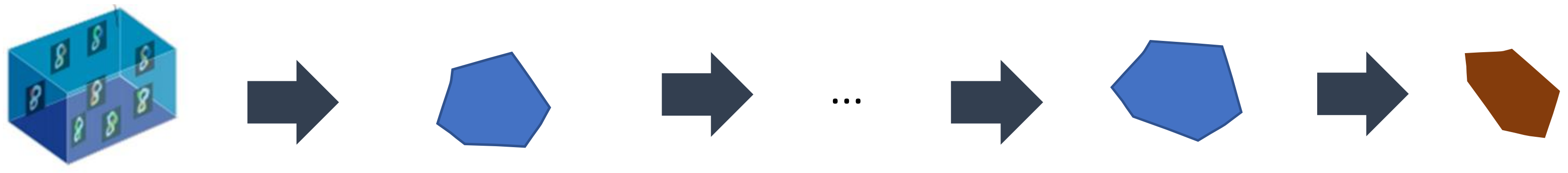
$\models$

$\boldsymbol{\psi} \equiv$ every point classifies to label 3

$\boldsymbol{\psi} \equiv \forall j \in [0,9]. o_3 \geq o_j$

The region $\phi_n$ is an over-approximation, hence if we fail to prove $\boldsymbol{\psi}$, it could be the property is really violated or there was over-approximation introduced during propagation which precludes provability.

## Key challenge: how to produce convex shapes?



To **instantiate incomplete methods** which use bound propagation, we need two parts:

1. What is the **convex approximation** ? E.g., Box, Zonotope, Polyhedra

2. How are these convex approximations produced? That is, what is the effect of the layer on a given approximation ? This effect is often called an **abstract transformer** as it transforms abstract shapes.

A **trade-off between approximation and speed** exists: transformers for Box are fast, but imprecise, while Polyhedra is more precise but in exponential complexity.

## Incomplete method I: Box

Let us answer these questions for the simplest and most efficient convex relaxation, namely, Box / Intervals. This relaxation will be useful in both, certification, as well as in provable training (which we look at in later lectures).

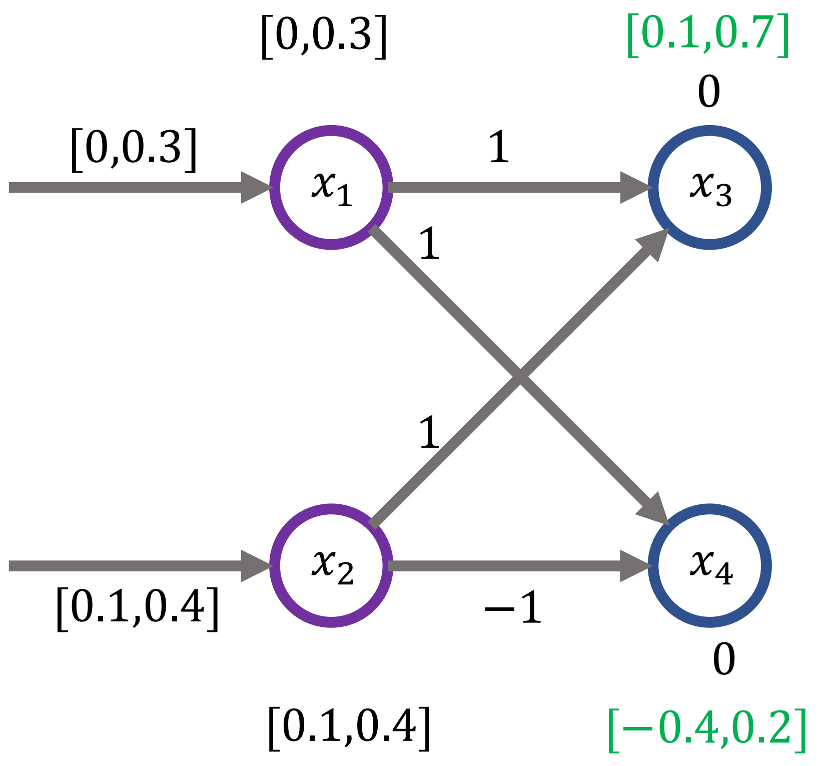
# Box Abstract Transformers Needed for Handling ReLU Neural Networks

| | |
|---|---|
| $[\boldsymbol{a}, \boldsymbol{b}] \ +^{\#} [\boldsymbol{c}, \boldsymbol{d}] = [\boldsymbol{a} + \boldsymbol{c}, \boldsymbol{b} + \boldsymbol{d}]$ | Addition transformer |
| $-^{\#}[\boldsymbol{a}, \boldsymbol{b}] = [-\boldsymbol{b}, -\boldsymbol{a}]$ | Negation transformer |
| $\boldsymbol{ReLU}^{\#}[\boldsymbol{a}, \boldsymbol{b}] = [\boldsymbol{ReLU}(\boldsymbol{a}), \boldsymbol{ReLU}(\boldsymbol{b})]$ | ReLU transformer |
| $\boldsymbol{\lambda}^{\#}[\boldsymbol{a}, \boldsymbol{b}] = [\boldsymbol{\lambda} \ * \boldsymbol{a}, \boldsymbol{\lambda} \ * \boldsymbol{b}]$ | Multiplication by a constant $\boldsymbol{\lambda} > \mathbf{0}$ |

- Here, $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{R}^{\boldsymbol{m}}$ where $\forall \boldsymbol{i}.\, \boldsymbol{a}_{\boldsymbol{i}} \leq \boldsymbol{b}_{\boldsymbol{i}}$
- $\boldsymbol{ReLU}(\boldsymbol{x}) = \boldsymbol{max}(\mathbf{0}, \boldsymbol{x})$
- $^{\#}$ denotes the **abstract** effect of the operation on the box

# Optimal Box transformer is not exact!

We have 2 pixels $(x_1, x_2)$ as input ranging over [0, 0.3] and [0.1, 0.4]



**Bounds using Box:**

$$0.1 \le x_3 \le 0.7$$
$$-0.4 \le x_4 \le 0.2$$

**Exact bounds would be:**

$$x_3 = x_1 + x_2$$
$$x_4 = x_1 - x_2$$
$$0 \le x_1 \le 0.3$$
$$0.1 \le x_2 \le 0.4$$

**This point is in the Box, but is infeasible:**

$$x_3 = 0.7 \quad x_4 = -0.4$$

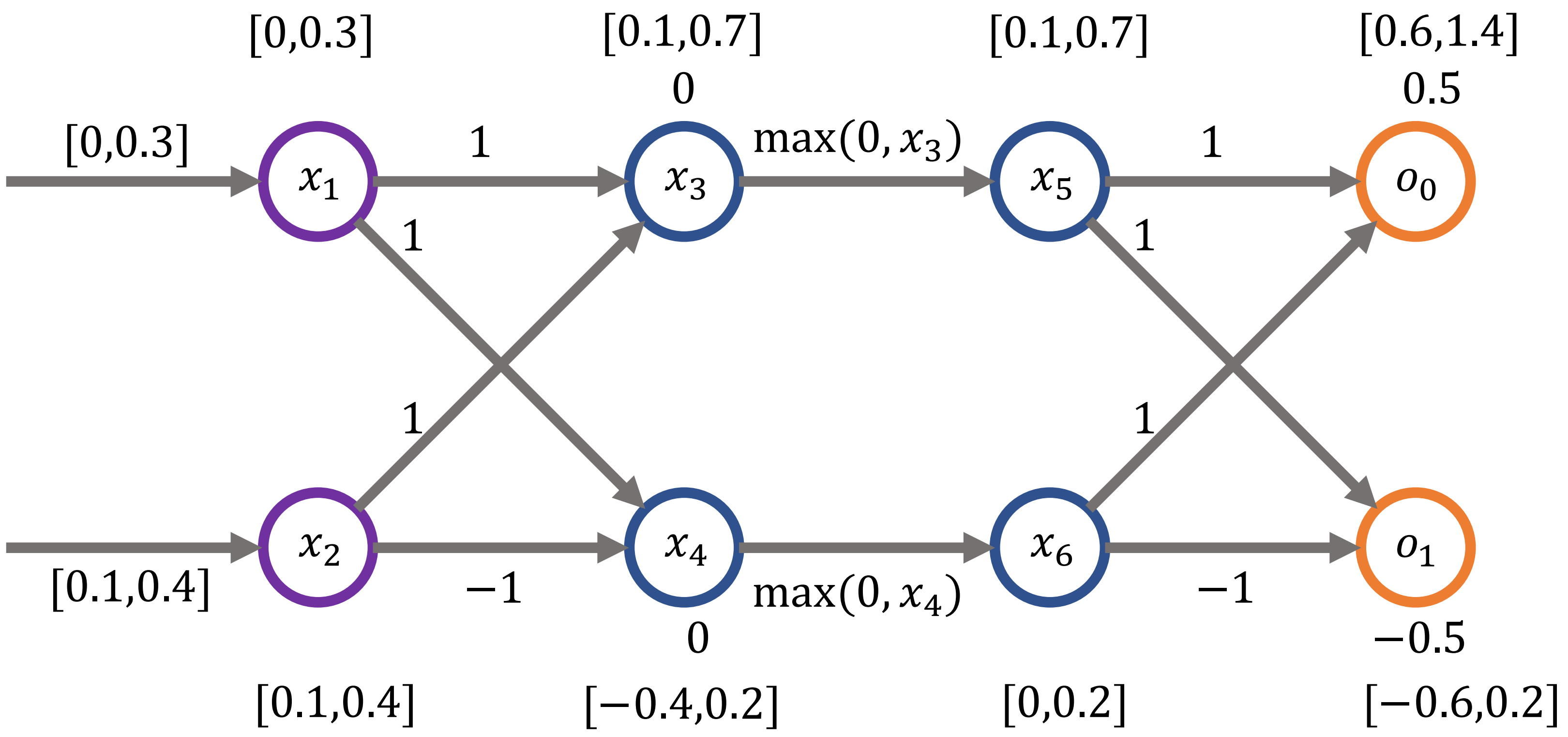
## Key Point

Even though the Box abstract transformer for the affine computation is optimal for the Box relaxation, it may not be complete (exact)! Nonetheless, even if not exact, it may be enough to verify the property of interest.

Lets see this next.

# Box succeeds in verifying robustness

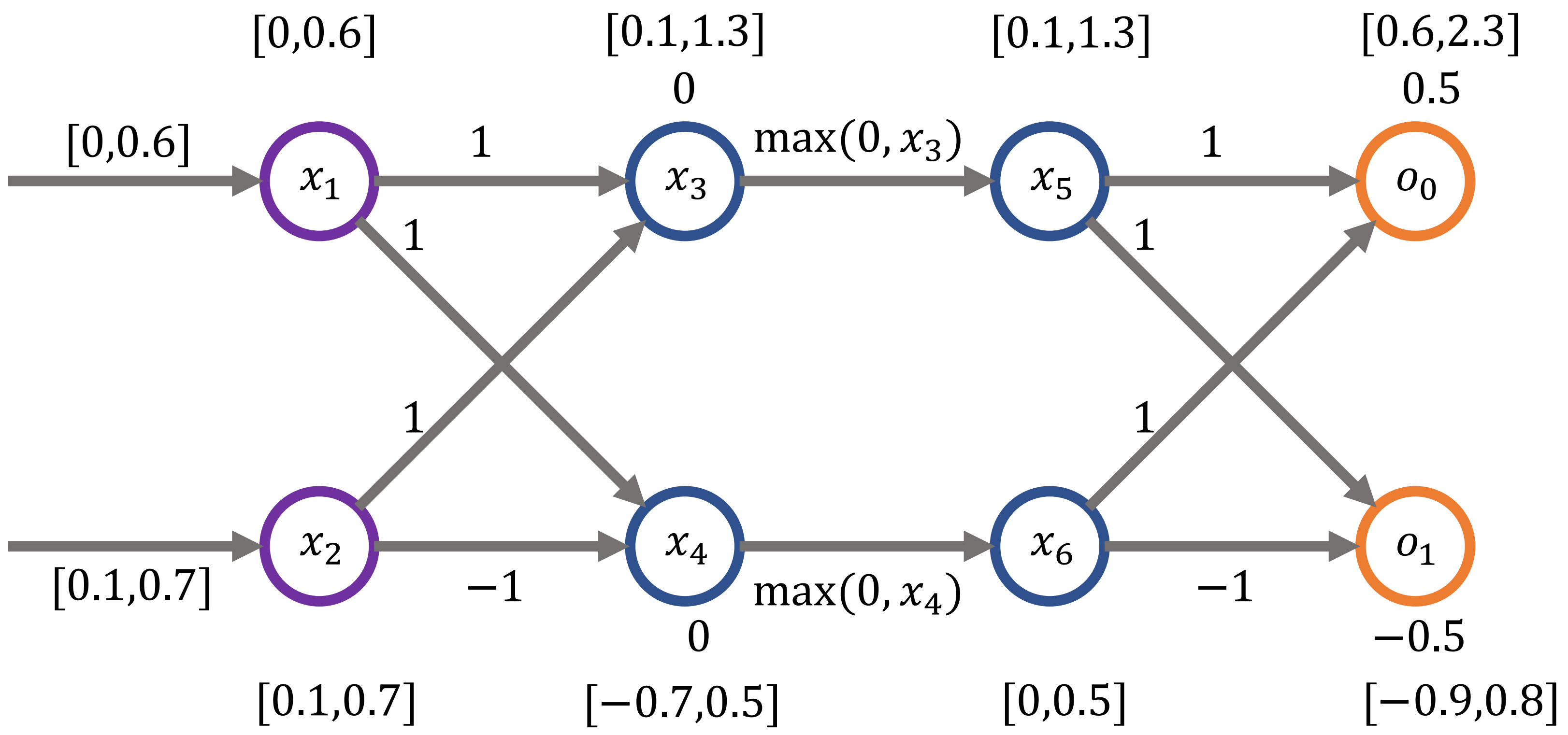
We have 2 pixels $(x_1, x_2)$ as input ranging over [0, 0.3] and [0.1, 0.4]



Using Box, we succeed in proving the network classifies any input in the range as 0. This is because **[0.6,1.4] > [-0.6, 0.2]**, provably so.

# Box **fails** in verifying robustness

Let us **slightly increase** the range of the input pixels to [0, **0.6**] and [0.1, **0.7**]



Using Box, we failed to prove the network classifies any input in the range as 0, even though property actually holds. This is because **[0.6,2.3] is not > [-0.9, 0.8]**, provably so.

# Summary So Far

- We introduced the concepts of complete and incomplete sound methods for neural network certification.

- We defined the Box convex relaxation and its best transformers, an instance of an incomplete method.

**Next:** complete certification and using Box to speed it up.