

Reliable and Trustworthy Artificial Intelligence

Certification and Box relaxations

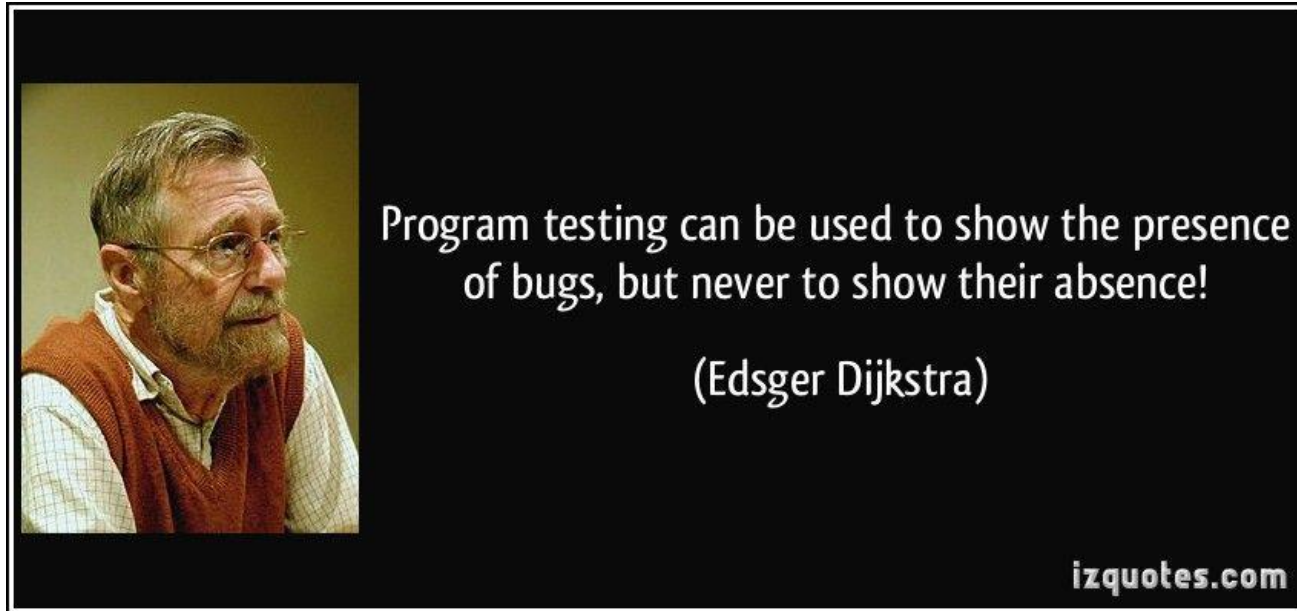
Martin Vechev
ETH Zurich

Fall 2022

Lecture outline

- Certification of neural networks: problem statement, types of guarantees, high level intuition
- Box convex relaxation

The fundamental problem



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 properties (e.g., robustness) of NN
- Learning interpretable specs of NN
- Comparing NNs

Problem Statement

Given

- a **neural network N**
- a **property over inputs ϕ** [called: pre-condition]
- a **property over outputs ψ** [called: post-condition]

Prove that $\forall i \in I. i \models \phi \Rightarrow N(i) \models \psi$ holds or return a **violation**

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

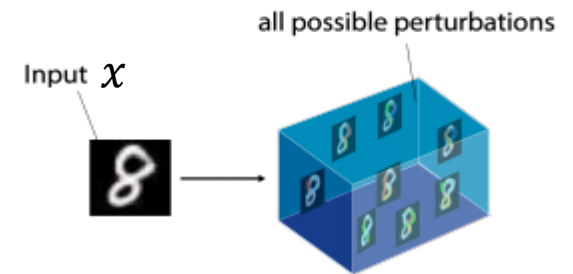
Step 1: Define ϕ Formally

For adversarial robustness, ϕ captures the input region:

Example ϕ :

L_∞ ball:

$$\text{Ball}(x)_\epsilon = \{ x' \in I \mid \|x - x'\|_\infty < \epsilon \}$$



Step 1: Define ϕ Formally

Certification



Region ϕ :

$$x_0 = 0$$

$$x_1 = 0.975 + 0.025\epsilon_1$$

$$x_2 = 0.125$$

...

$$x_{784} = 0.938 + 0.062\epsilon_{784}$$

$$\forall i. \epsilon_i \in [0,1]$$

Example: here, we capture **brightening attack** (only few pixels have ϵ)

Step 2: Verify ϕ satisfies ψ

Certification



We need to prove this property:

$$\forall i \in I. i \models \phi \Rightarrow N(i) \models \psi$$

Region ϕ :

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

Property ψ : every point classifies to 8:

$$\psi : \forall j \in [0,9], j \neq 8. o_8 > o_j$$

Here, o is the network output (pre-softmax)

Challenge

Certification



We need to prove this property:

$$\forall i \in I. i \models \phi \Rightarrow N(i) \models \psi$$

How to prove this property?

Challenge: as ϕ can capture an unbounded set of points, one cannot simply enumerate points in ϕ and check whether each satisfies ψ

Region ϕ :

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

Property ψ : every point classifies to 8:

$$\psi : \forall j \in [0,9], j \neq 8. o_8 > o_j$$

Here, o is the network output (pre-softmax)

Certification Methods: Sound vs. Unsound

A reasoning method is called **sound** if upon termination it only states that a property holds, if the analysed program actually satisfies said property. We typically refer to these methods as **certification methods**.

A reasoning method is called **unsound** if upon termination it can state the property holds, even though it does not. 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 **always** able to prove that a property holds if 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 later lecture.
Yes	No	Return “No, property does not hold” [this lecture]
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) , Branch and Bound with Lagrange Multipliers and DeepPoly

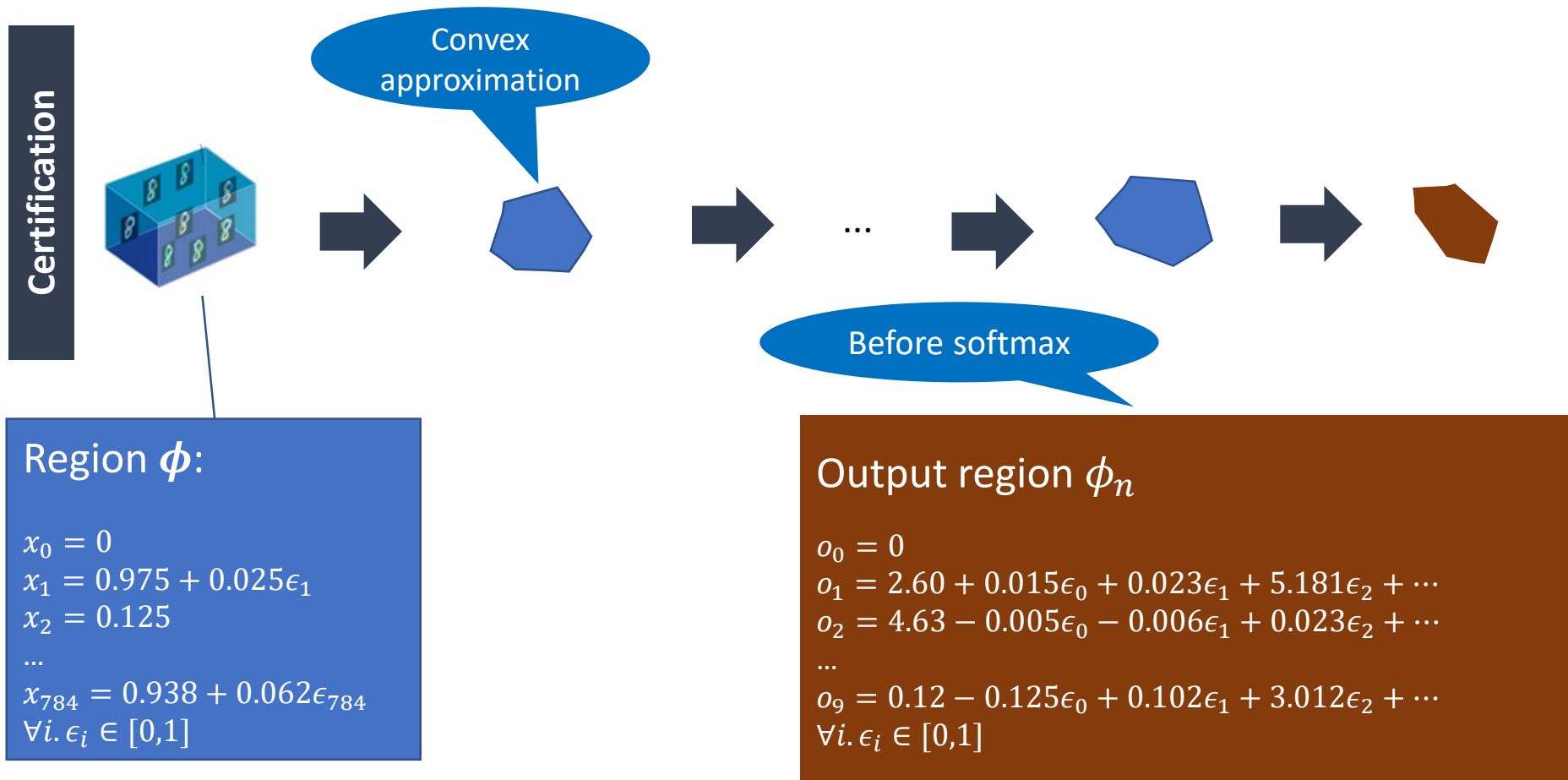
Active research area: create scalable and precise methods

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 ϕ , we will “push” ϕ 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

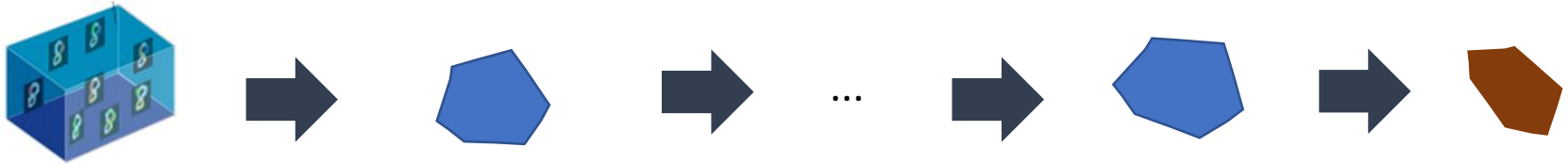
Step 1: Compute bounds by propagating ϕ



Over-approximation ϕ_n means it is possible there are concrete points inside ϕ_n which **cannot be produced** by any concrete point inside ϕ .

Step 2: Certify the property

Certification



Output region ϕ_n

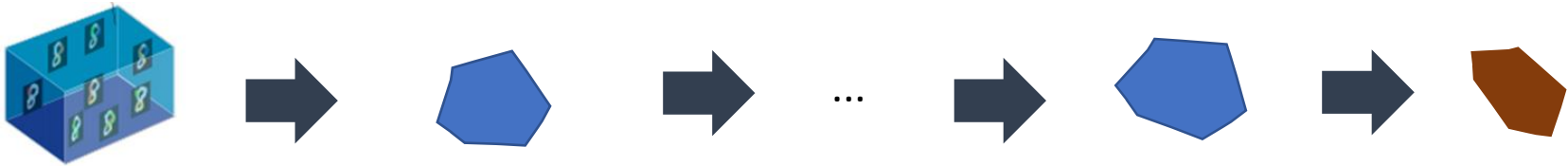
$$\begin{aligned} o_0 &= 0 \\ o_1 &= 2.60 + 0.015\epsilon_0 + 0.023\epsilon_1 + 5.181\epsilon_2 + \dots \\ o_2 &= 4.63 - 0.005\epsilon_0 - 0.006\epsilon_1 + 0.023\epsilon_2 + \dots \\ &\vdots \\ o_9 &= 0.12 - 0.125\epsilon_0 + 0.102\epsilon_1 + 3.012\epsilon_2 + \dots \\ &\forall i. \epsilon_i \in [0,1] \end{aligned}$$






$\psi \equiv$ every point classifies to label 8
 $\psi : \forall j \in [0,9], j \neq 8. o_8 > o_j$

The region ϕ_n is an **over-approximation**, hence if we fail to prove ψ , it could be the property is violated or that our method was simply not precise enough to prove it.

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.

Lecture outline

- Certification of neural networks: problem statement, types of verifiers
- **Box convex relaxation**

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 (by itself or in combination with branch and bound methods), as well as in provable training (which we study in later lectures).

Box Abstract Transformers Needed for Handling ReLU Neural Networks

$$[a, b] +^{\#} [c, d] = [a + c, b + d]$$

Addition transformer

$$-^{\#}[a, b] = [-b, -a]$$

Negation transformer

$$\mathit{ReLU}^{\#}[a, b] = [\mathit{ReLU}(a), \mathit{ReLU}(b)]$$

ReLU transformer

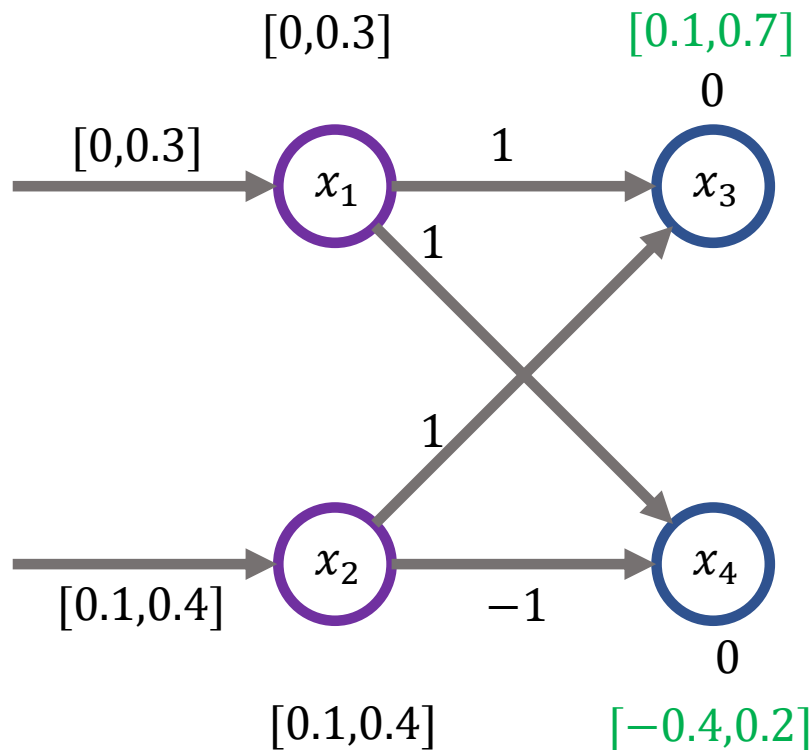
$$\lambda^{\#}[a, b] = [\lambda * a, \lambda * b]$$

Multiplication by a constant $\lambda > 0$

- Here, $a, b \in \mathbf{R}^m$ where $\forall i. a_i \leq b_i$
- $\mathit{ReLU}(x) = \mathit{max}(\mathbf{0}, x)$
- $^{\#}$ 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:

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

Exact bounds would be:

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

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

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

Key Point

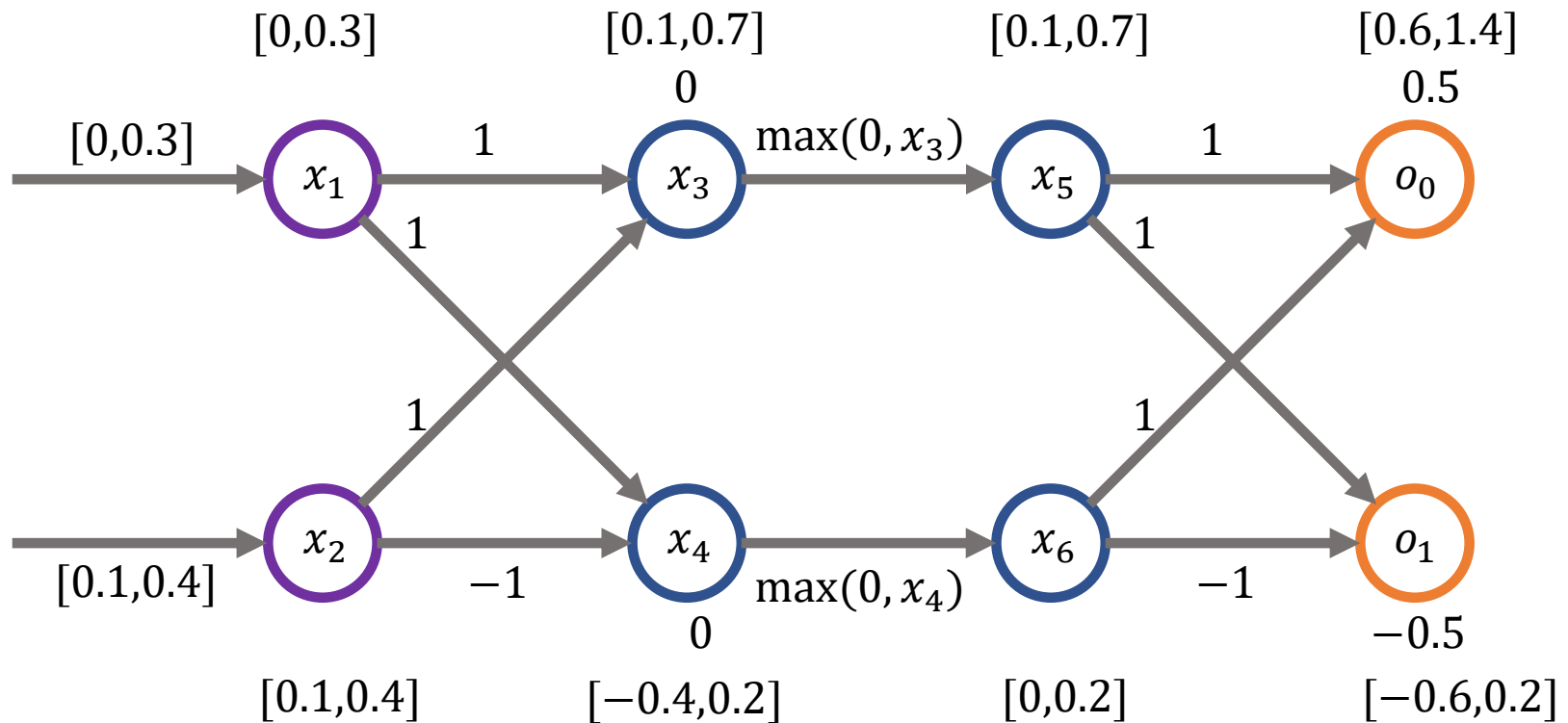
Even though the Box abstract transformer for the affine computation **is optimal for the Box relaxation**, the method **is not precise**, that is, the output Box may include elements not obtainable by applying the concrete affine function to elements of the input box!

Nonetheless, it may be **enough to verify the property of interest**.

Lets see this next.

Box **succeeds** in verifying robustness

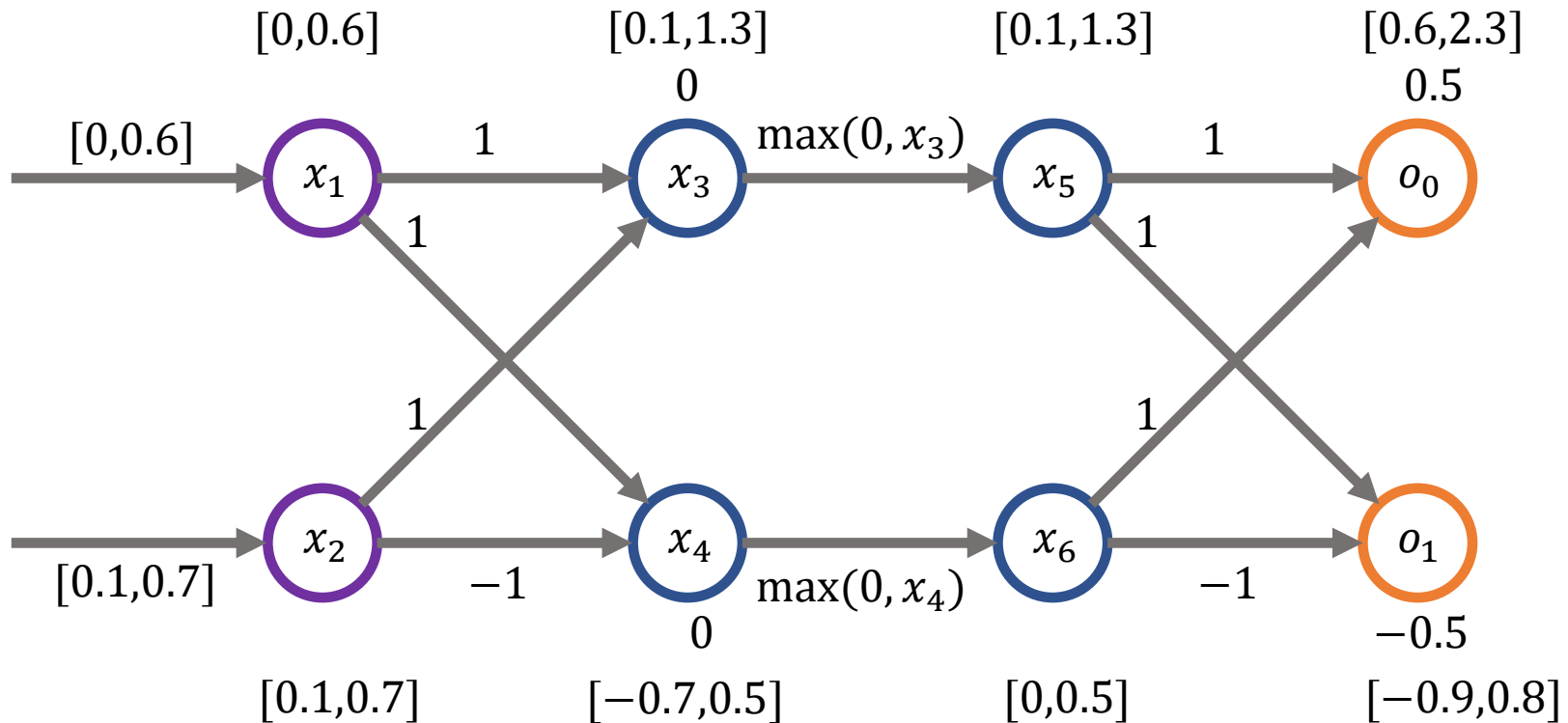
We have 2 pixels (x_1, x_2) as input ranging over $[0, 0.3]$ and $[0.1, 0.4]$



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 $[0, \underline{0.6}]$ and $[0.1, \underline{0.7}]$



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.²⁵

Transform the program to verify more

Lets include the property in the network and merge it with the last affine transformation:

$$o_0 - o_1 > 0$$

$$x_5 + x_6 + 0.5 - (x_5 - x_6 - 0.5) > 0$$

$$x_5 + x_6 + 0.5 - x_5 + x_6 + 0.5 > 0$$

$$2x_6 + 1 > 0$$

As $x_6 \geq 0$, using Box, by transforming the network a little bit, we managed to prove the property.

The problem with Box

Box is **simple and efficient**, but the problem is that it **loses too much precision**, in **both**, the ReLU abstract transformer but also in the affine abstract transformer.

The Polyhedral convex relaxation we study next is such that the **affine transformer will not lose precision** (that is, it will be precise); however, its ReLU transformer will again lose some precision.

Lecture Summary

- 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.