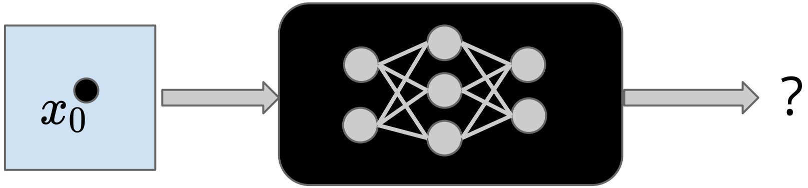
AAAI 2022 Tutorial on Neural Network Verification

Part I: Introduction to NN Verification

Huan Zhang (CMU), Kaidi Xu (Drexel), Shiqi Wang (Columbia) and **Cho-Jui Hsieh (UCLA)**

Feb 23, 2022



Can We Trust NNs in Mission-critical Tasks?

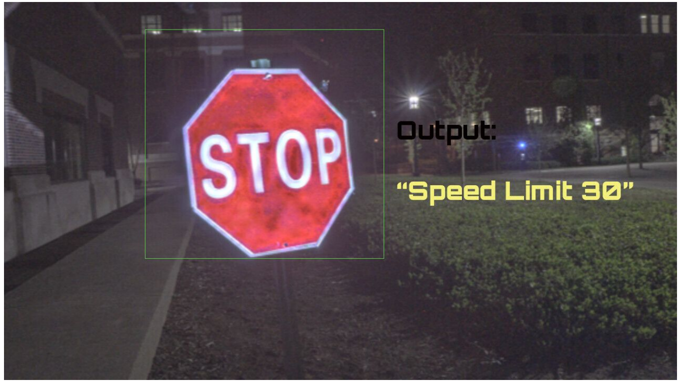
Autonomous Driving
Aircraft Autopiloting

Medical Equipments
AI-based Diagnosis

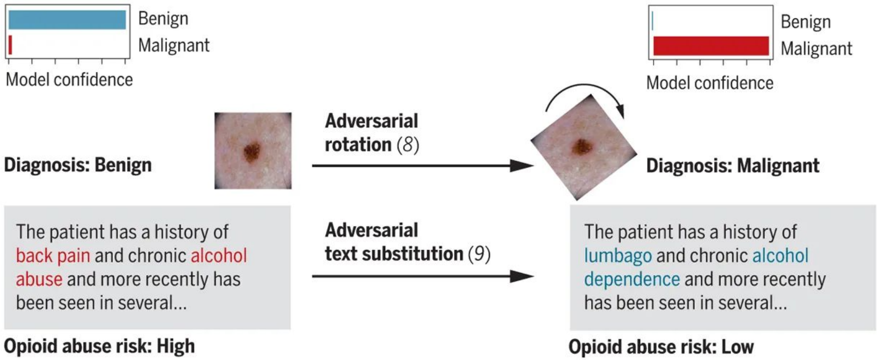
Security/Surveillance
Systems

Can We Trust NNs in Mission-critical Tasks?

Researchers say "no"...



"Optical adversarial attack" by Gnanasambandam et al., ICCV 2021

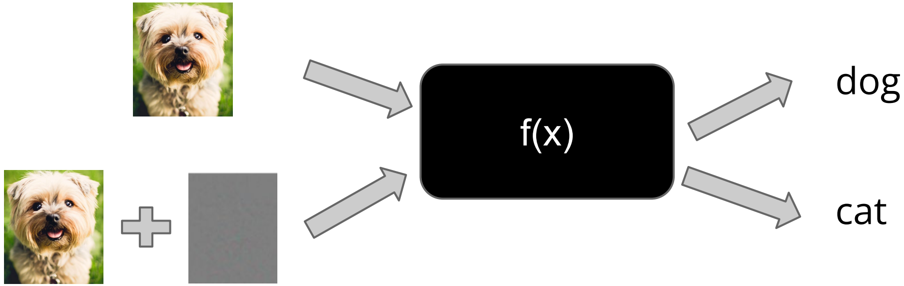


"Adversarial attacks on medical machine learning" by N. Cary et al., Science

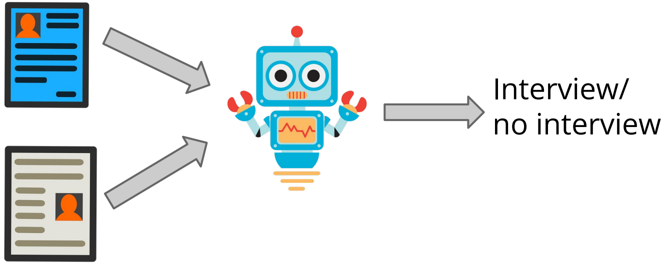
What is Neural Network Verification?

We hope to *prove* that NNs have some desired properties we can *formally* trust:

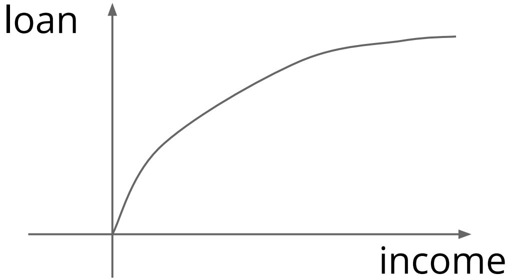
Robustness



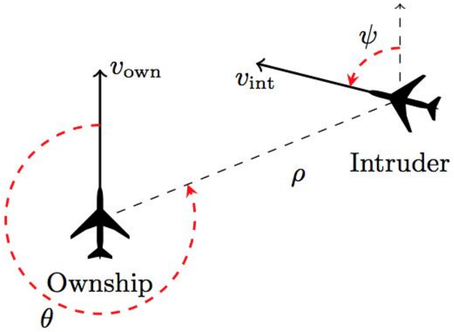
Fairness



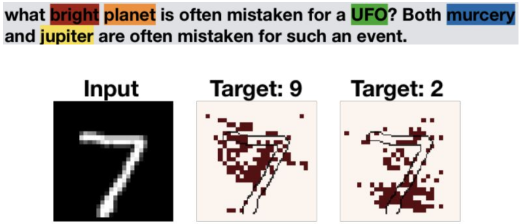
Monotonicity



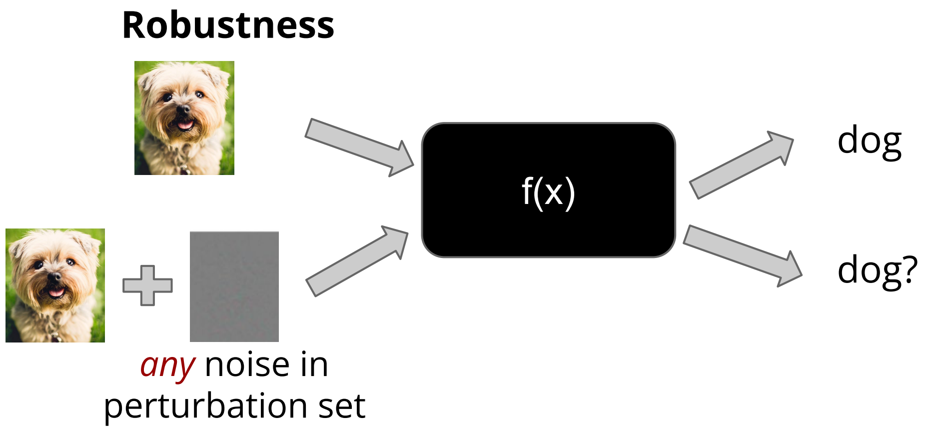
Correctness

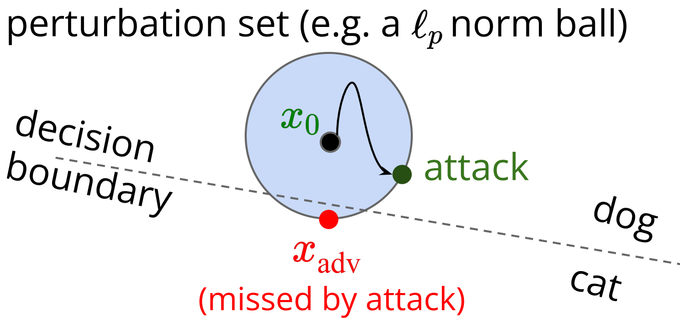


Interpretability



What is Neural Network Verification?



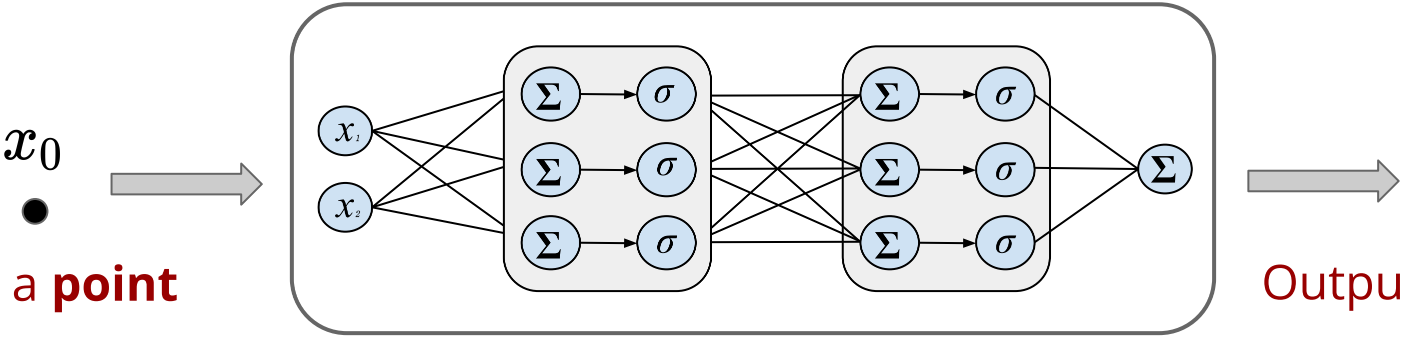


- Verification requires *a formal proof* to show the property holds
- In the robustness verification setting, a model can't be attacked ≠ Verified
- Many heuristic defense was broken under stronger attacks (e.g., Athalye et al. 2018)
- A verified model cannot be attacked by any attacks (including unforeseen ones)

The Basic Formulation of Robustness Verification

Consider a simple binary classification case:

x_0



$f(x_0) = 1.2$

Input is a **point**

Output is a score

Neural Network

$f(x_0) > 0$

Positive Example

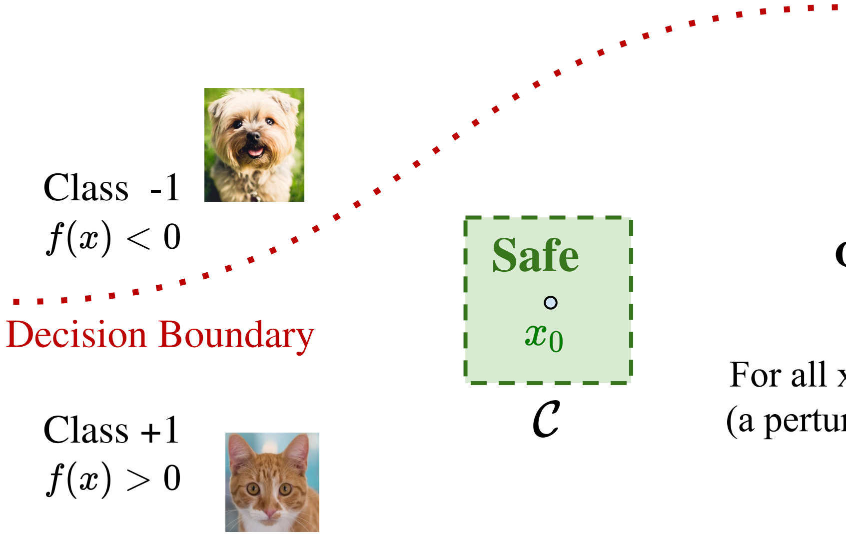
$f(x_0) \leq 0$

Negative Example

The Basic Formulation of Robustness Verification

Suppose $f(x_0) > 0$. Can we verify this property:

$$f(x) > 0, \forall x \in \mathcal{C}$$



Goal: Prove

$$f(x) > 0$$

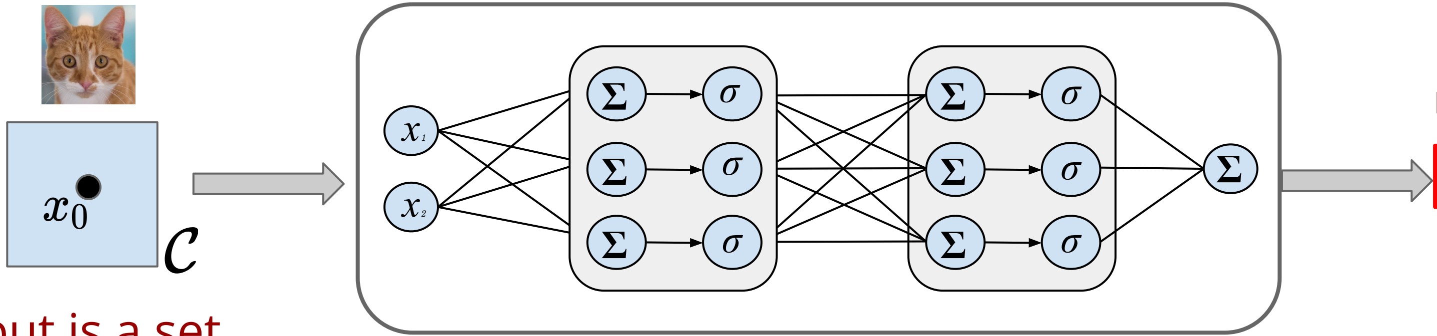
For all x in the green box

(a perturbation around x_0)

The Basic Formulation of Robustness Verification

Suppose $f(x_0) > 0$. Can we verify this property:

$$f(x) > 0, \forall x \in \mathcal{C}$$



Input is a set

Neural Network

Output is a range/set of scores

$$\boxed{0.2} \leq f(x) \leq 2.2$$

"cat" even in the worst case

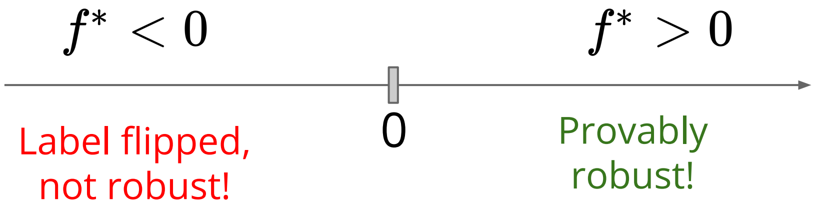
Must consider a set of infinite points as the input of the NN.

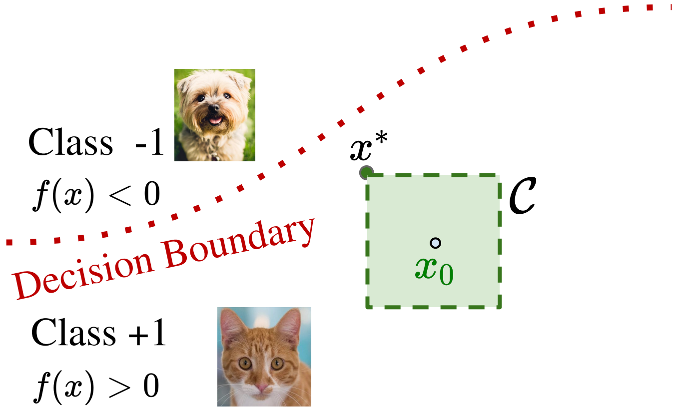
The Basic Formulation of Robustness Verification

Assuming $f(x_0) > 0$, we solve the optimization problem to find the worst case:

$$f^* = \min_{x \in \mathcal{C}} f(x)$$

$\mathcal{C}$ is usually a perturbation set "around" x_0, e.g., $\mathcal{C} := \{x \mid \|x - x_0\|_p \leq \epsilon\}$



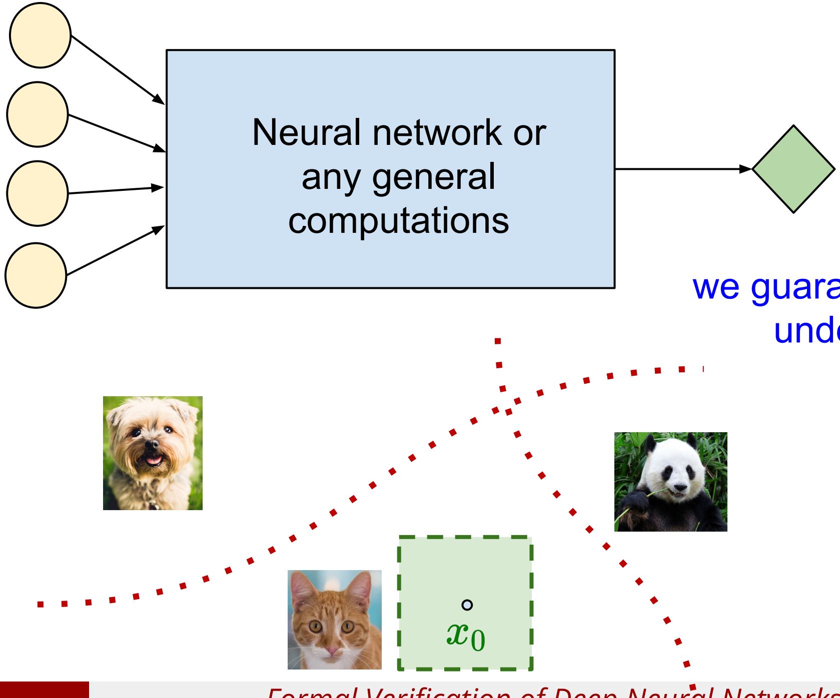


Is it a hard problem?

The Basic Formulation of Robustness Verification

Multi-class case:

Data perturbed arbitrarily within a set



output bounds

(guaranteed score ranges)

2.3 ≤ cat ≤ 4.5

-0.8 ≤ dog ≤ **1.2**

-4.2 ≤ panda ≤ **-0.1**

we guarantee that "cat" stays top-1 under input perturbations

Why the Verification Problem is Challenging?

This is the fundamental problem we want to solve (Wong & Kolter 2018, Salman et al. 2019):

$$f^* = \min z^{(L)}$$

Last layer output f(x), at layer L

pre-activation

$$\text{s.t.} \quad z^{(i)} = W^{(i)} \hat{z}^{(i-1)} + b^{(i)} \qquad i \in \{1, \cdots, L\}$$

Linear constraints

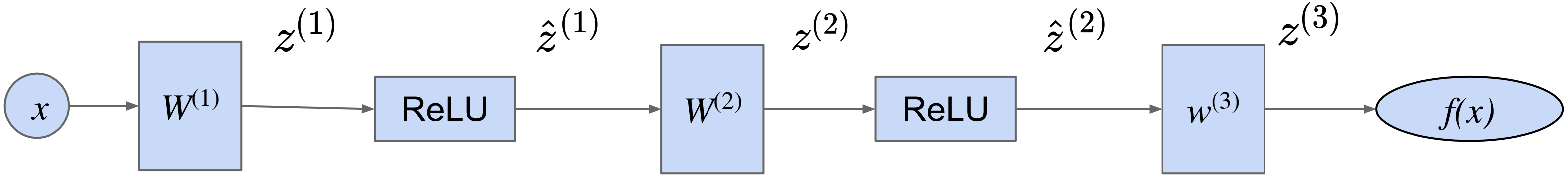
$$\hat{z}^{(i)} = \sigma(z^{(i)}) \quad i \in \{1, \cdots, L-1\}$$

Non-linear, non-convex constraints

post-activation

$$\hat{z}^{(0)} = x, \quad x \in \mathcal{C}$$

Input perturbations

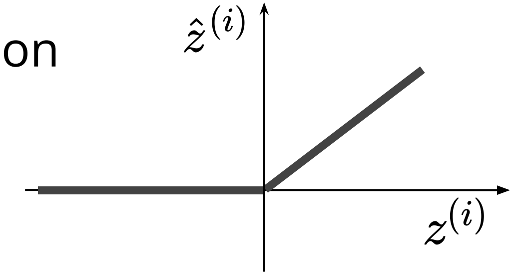


Why the Verification Problem is Challenging?

$$\hat{z}^{(i)} = \sigma(z^{(i)}), i \in \{1, \cdots, L-1\}$$

Non-convex constraints

e.g., ReLU function

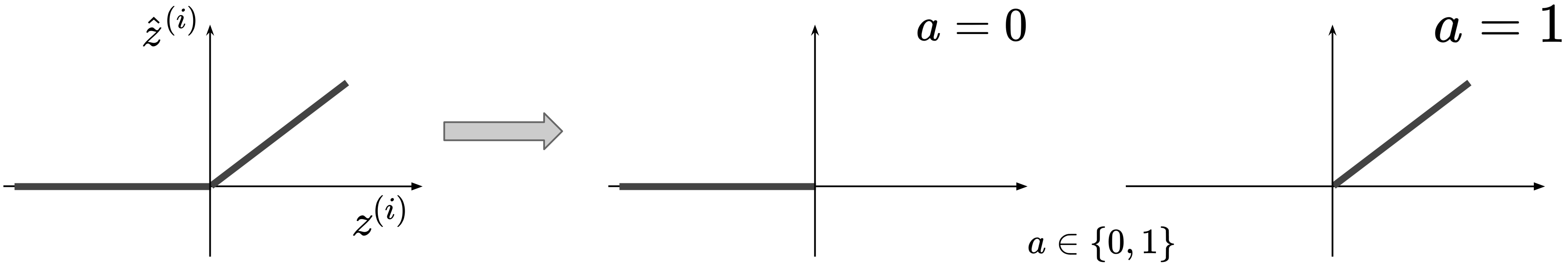


The constraint says that $(\hat{z}^{(i)}, z^{(i)}) \in \mathrm{Graph}(\mathrm{ReLU})$

Generally, NP-complete (Katz et al., 2017)

Why the Verification Problem is Challenging?

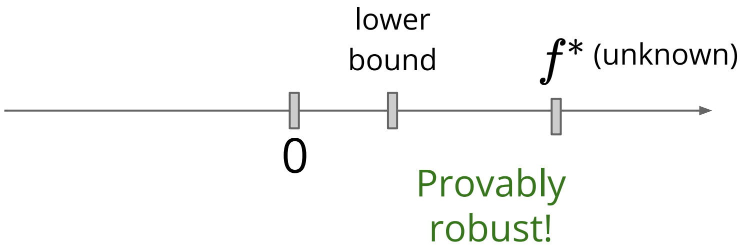
- Approach 1: Using mixed integer programming (MIP) encoding of ReLU neurons (Tjeng et al. 2017) => *Complete verification* which solves the exact f^*

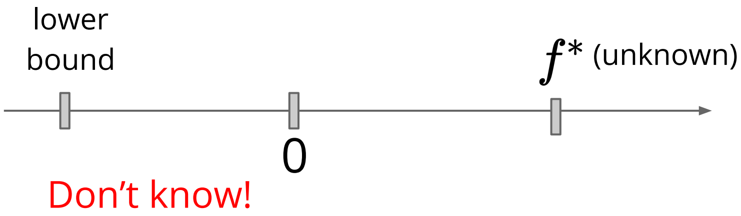


$$\hat{z}^{(i)} = \mathrm{ReLU}(z^{(i)}) \qquad \hat{z}^{(i)} = 0 \qquad \hat{z}^{(i)} = z^{(i)}$$

Why the Verification Problem is Challenging?

- Approach 2: Relax the MIP to a LP (Salman et al. 2019) => Incomplete verification: find a *lower bound* of f^*. If lower bound >0, the network is verifiably robust
  - Still requires an LP solver, which can still be slow for large networks
  - LP often produces loose bound; if lower bound << 0 it is useless





Neural Network Verification: History

- **SMT** (Huang et al., 2017; Ehlers 2017)
- **MILP** (Cheng et al 2017; Tjeng et al., 2019)
- **Reluplex** (Katz et al., 2017)

"Bound propagation"-based

- **Convex Adversarial Polytope** (Wong & Kolter 2018)
- **CROWN** (Zhang et al., 2018)
- **DeepPoly** (Singh et al., 2019)
- **Neurify** (Wang et al., 2018)
- **SDP Relaxation** (Raghunathan et al., 2018; Dathathri et al., 2020)
- **Optimal Convex** (Tjandraatmadja et al, 2020)

- **Branch and bound with LP solver** (Bunel et al., 2018; 2020; Lu & Kumar., 2019)

- **Lagrangian Decomposition** (Bunel et al., 2020)
- **α-CROWN + BaB** (Xu et al., 2020)
- **β-CROWN + BaB** (Wang et al., 2021)
- **Active Set** (De Palma et al., 2021)

2017 2018 2019 2020 2021 year

First era: formulate NN verification using existing solvers

Second era: Efficient Incomplete Verifiers

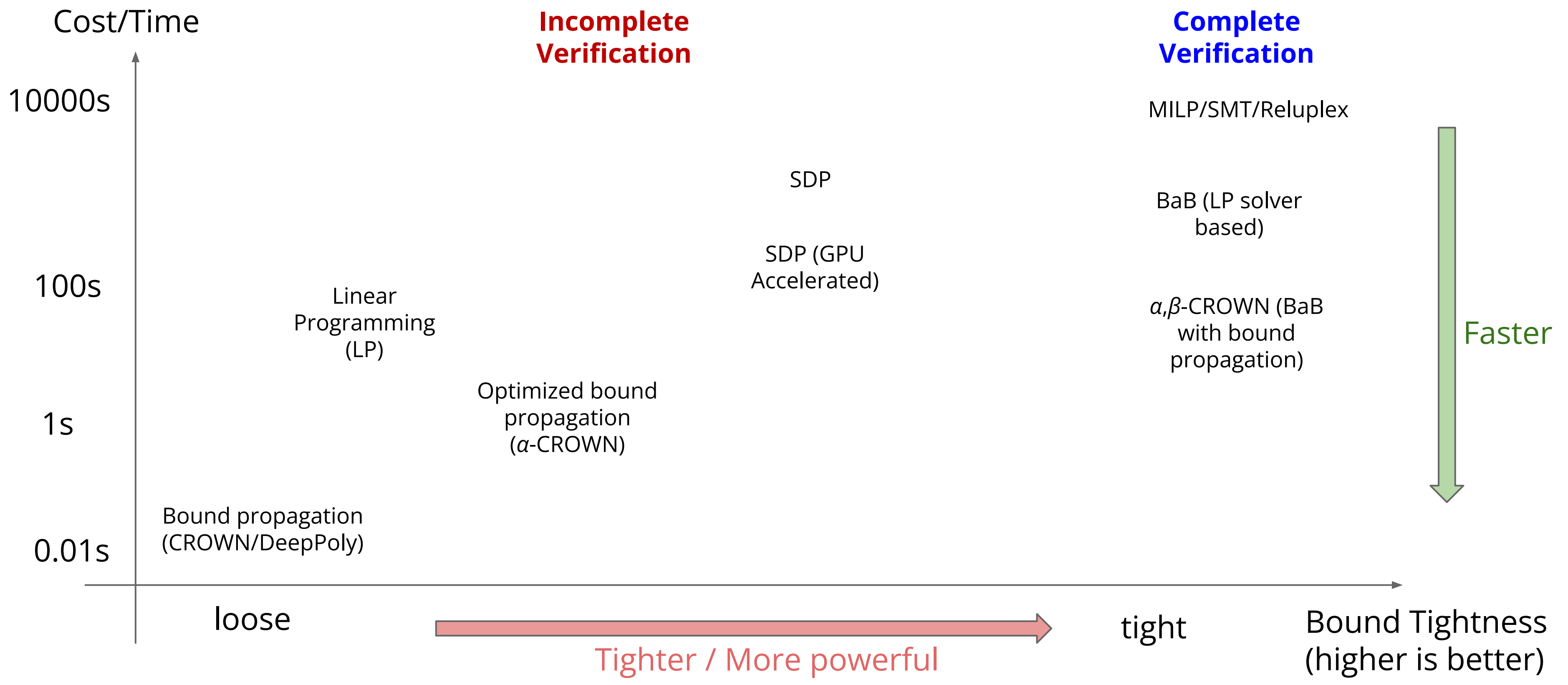
Third era: Branch & Bound (BaB) based Solvers

Fourth era: Efficient and GPU accelerated BaB

<100 neurons

CNN with >100K neurons

Neural Network Verification: Representative Algorithms



Next Part

Basic Verification Algorithms (40min)

Practical Verification Tools (1 hr)