Formal Verification of ML Models (Part II)

EECE 7398

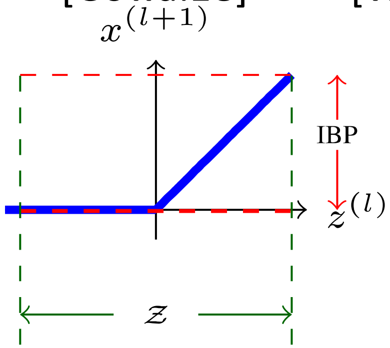
Lecture 4

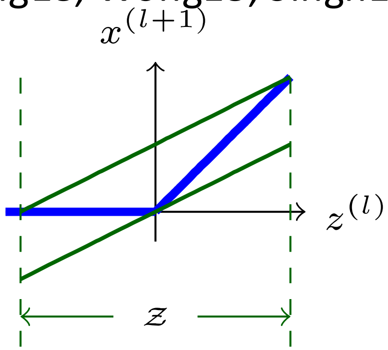
Recap of Last Lecture

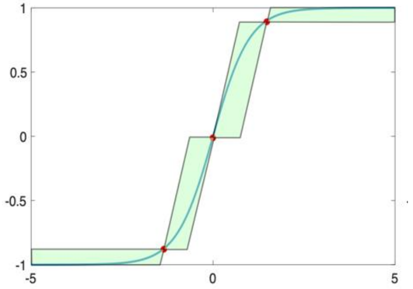
- Verification as an optimization problem
- MILP Formulation
- Soundness vs. completeness
- CROWN

Fast — Exact

Interval Prop. [Gowal18] | LP [Weng18, Wong18, Singh18] | QP/SDP [Raghunathan18, Fazlyab19] | ReLUplex [Katz17]







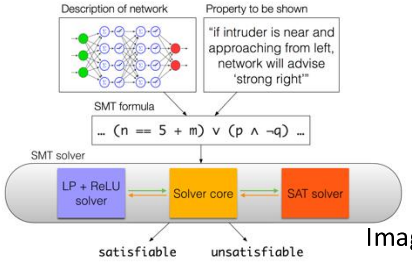


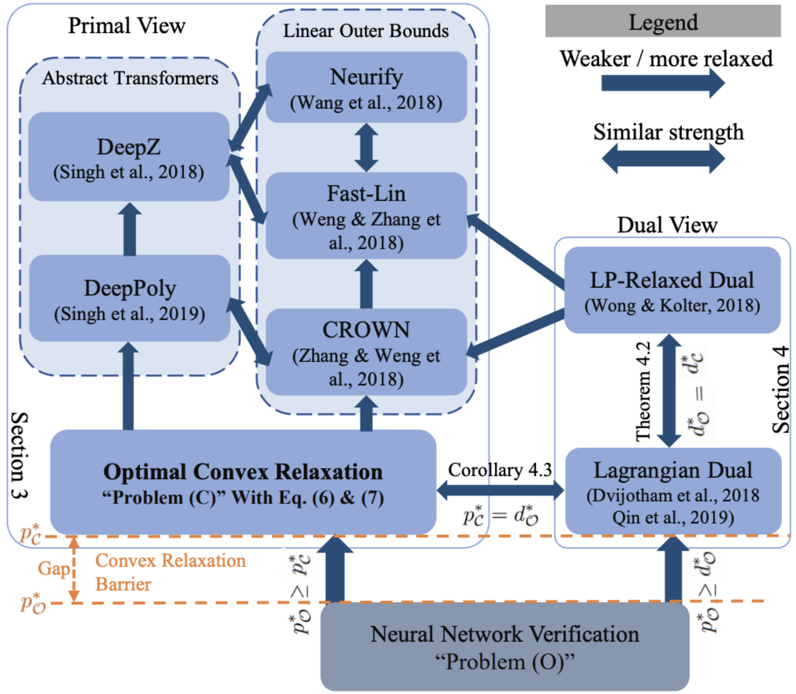
Image: Kuper

Recap of Convex Relaxation Barrier [Salman19]

- Tightness of many verification algorithms is limited by optimal layer-wise convex relaxation

Some possible reasons:

- Relaxation of nonlinearities on box domain
- Recursive calculation of pre-activation bounds → looseness accumulates with layers
- “Unstable” ReLU neurons incur relaxation gap



Today's Plan

- Branch and Bound (BaB)
- Sampling-Based Methods
- K-Neuron Constraints
- SDP
- Software Libraries for Verification

Branch and Bound

Branch and Bound (BaB / BnB)

- "Most commonly used tool for solving NP-Hard optimization problems" [Clausen99]

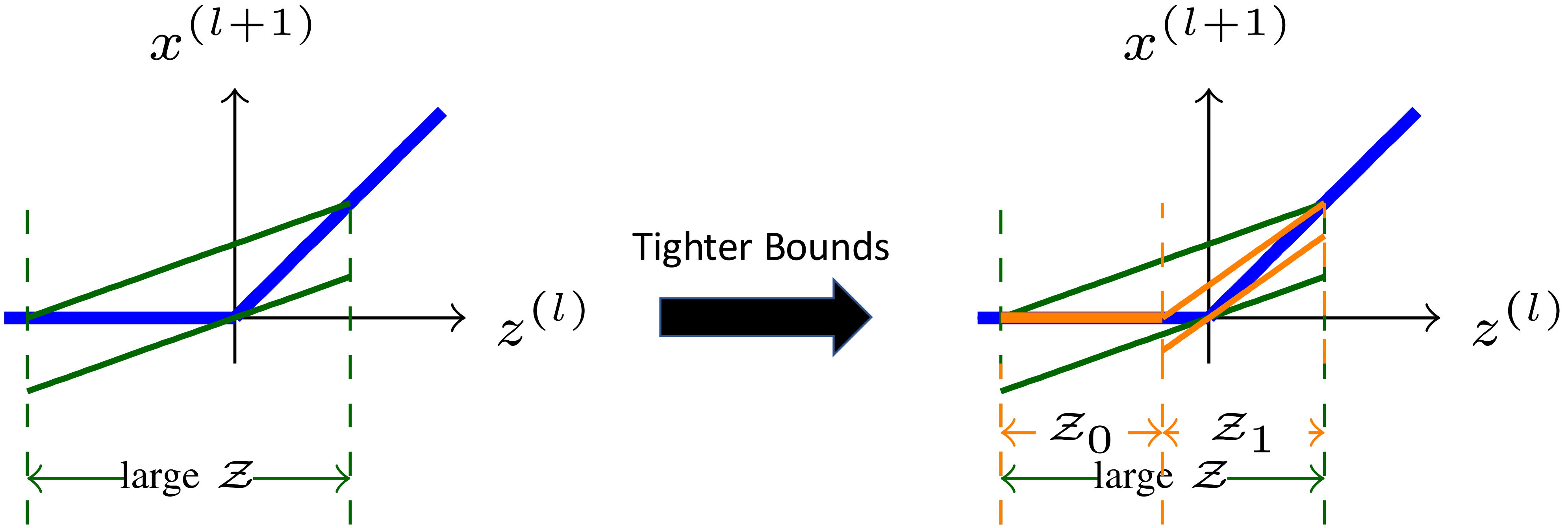
General Idea: Find reasonably tight bound on difficult optimization problem

- Split optimization problem into sub-problems (e.g., split feasible set in half)
- Bound solution to sub-problems → eliminate sub-problems that can't contain optimal solution

- **Key Insight:** Sub-problems may be solved/bounded more easily/tightly than original problem

BnB for NN Verification

- Why would we expect BnB to be useful for NN verification?
- Our relaxations are based on fixed pre-activation bounds
- Large pre-activation bounds can cause loose relaxations



BnB for NN Verification

BnB requires these components:

- **Search strategy:** how to choose which sub-problem to investigate next (e.g., MC sampling, heuristics based on gradients)
- **Branching rule:** how to split a problem into its sub-problems (e.g., split input set, split uncertain ReLU)
- **Bounding methods:** how to compute bounds on a sub-problem's solution (e.g., CROWN, Fast-Lin)

Branch and Bound for Piecewise Linear Neural Network Verification

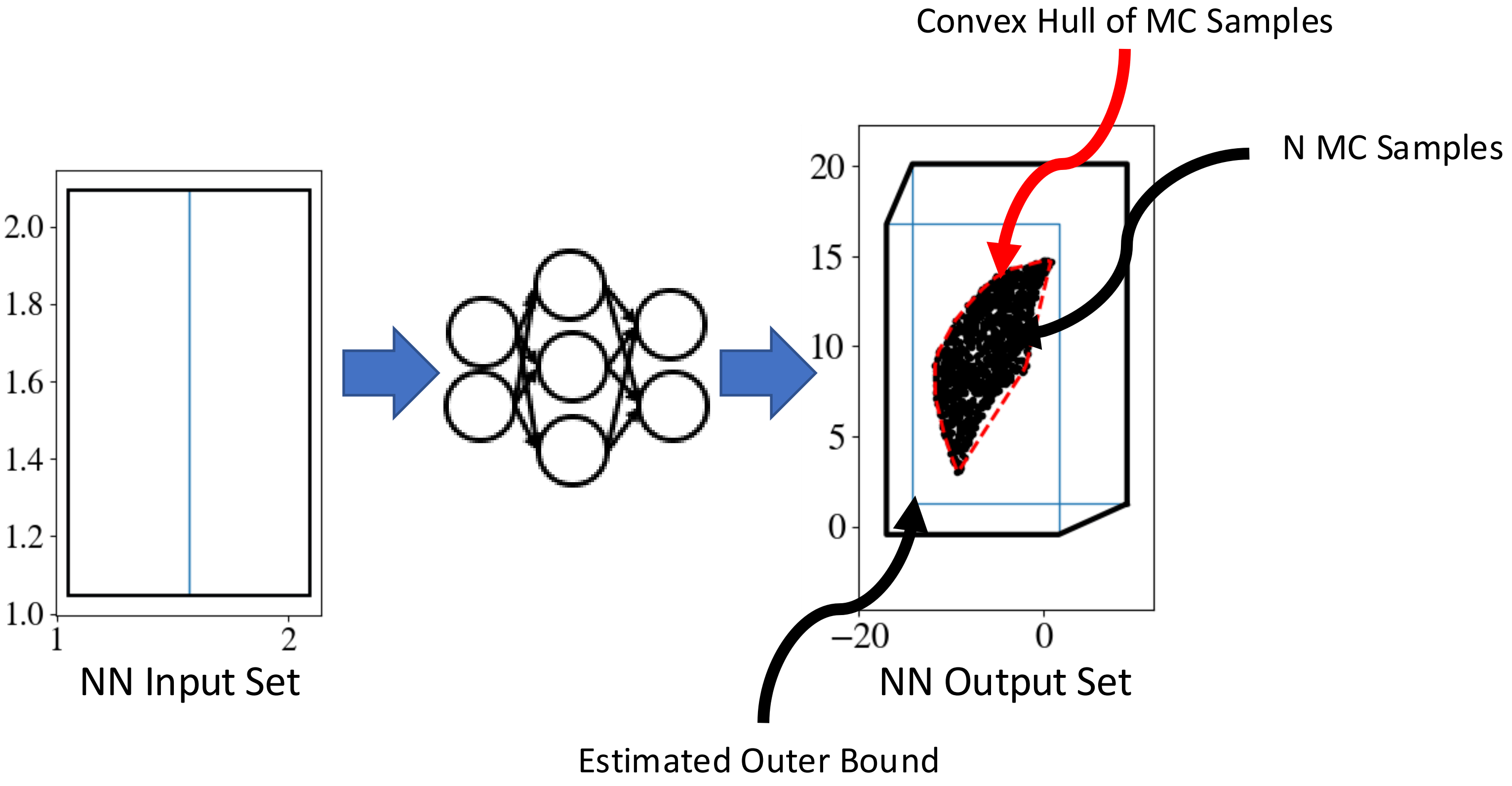
Rudy Bunel * RUDY@ROBOTS.OX.AC.UK
Ilker Turkaslan ILKER.TURKASLAN@LMH.OX.AC.UK
Philip H.S. Torr PHILIP.TORR@ENG.OX.AC.UK
M. Pawan Kumar PAWAN@ROBOTS.OX.AC.UK
Department of Engineering Science
University of Oxford
Oxford OX1 3PJ

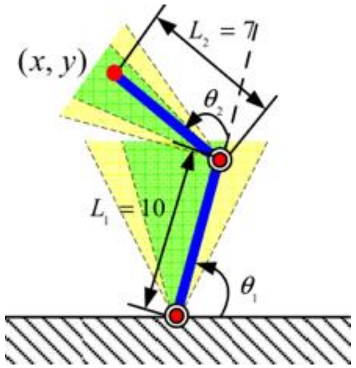
Jingyue Lu * JINGYUE.LU@SPC.OX.AC.UK
Department of Statistics
University of Oxford
Oxford OX1 3LB

Pushmeet Kohli PUSHMEET@GOOGLE.COM
Deepmind
London N1C 4AG

Tight Analysis via Partitioning

Refine outer bounds on output set to approach inner bounds from MC sampling

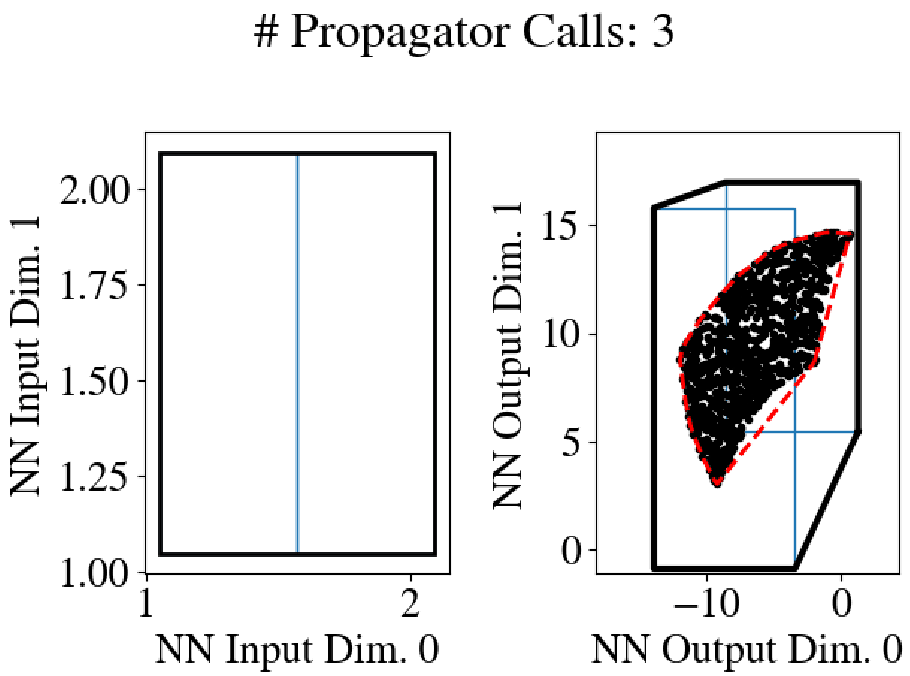


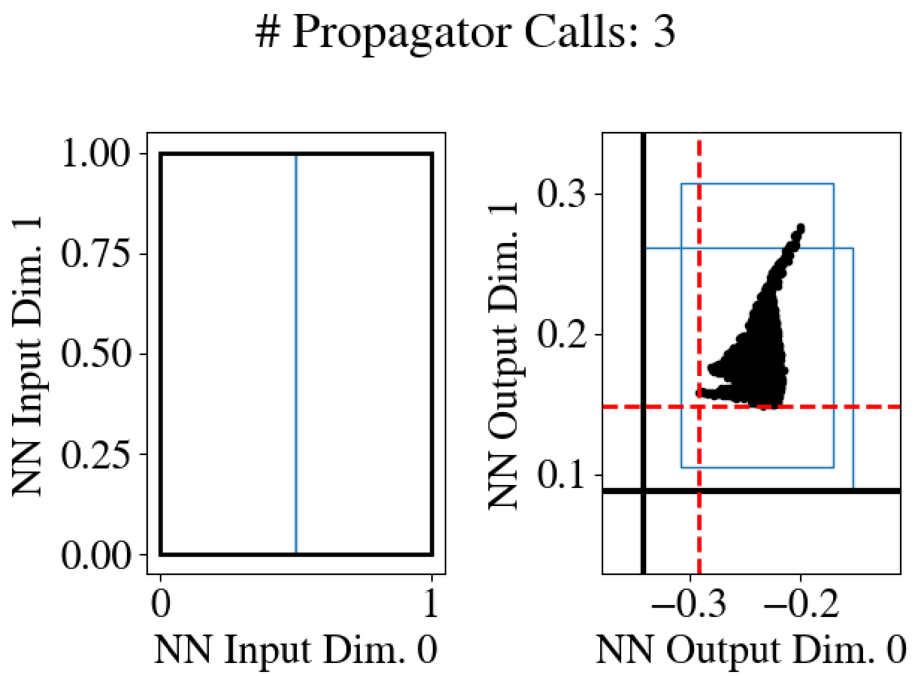


Everett, Habibi, How, “Robustness analysis of neural networks via efficient partitioning with applications in control systems” IEEE Control Systems Letters 2021 and ACC 2021

Branch and Bound: Illustration

- Branches may depend on what shape we want to produce



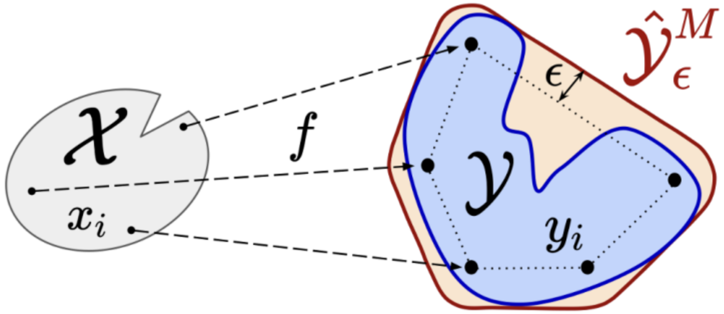


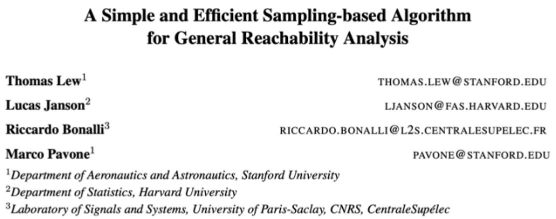
Everett, Habibi, How, "Robustness analysis of neural networks via efficient partitioning with applications in control systems" IEEE Control Systems Letters 2021 and ACC 2021

Sampling-Based Methods

Sampling-Based Methods

- So far, we have avoided sampling-based methods
- However, sampling-based methods can be much simpler, faster, tighter!
- For example, [Lew21] samples from X, propagates those through NN f, then inflates/pads appropriately
  - Proves convergence with increasing samples
  - Assuming NN is Lipschitz continuous, proves that estimate from finite # of samples is ϵ-accurate (with probability above some threshold)
- Still difficult to scale to large / high-dim spaces (covering number)

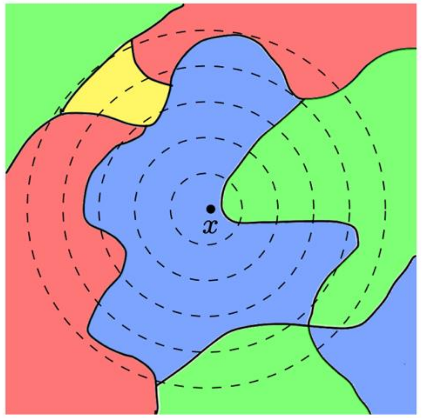



A Simple and Efficient Sampling-based Algorithm
for General Reachability Analysis

Thomas Lew[1] THOMAS.LEW@STANFORD.EDU
Lucas Janson[2] LJANSON@FAS.HARVARD.EDU
Riccardo Bonalli[3] RICCARDO.BONALLI@L2S.CENTRALESUPELEC.FR
Marco Pavone[1] PAVONE@STANFORD.EDU
[1]*Department of Aeronautics and Astronautics, Stanford University*
[2]*Department of Statistics, Harvard University*
[3]*Laboratory of Signals and Systems, University of Paris-Saclay, CNRS, CentraleSupélec*

Randomized Smoothing

- So far, haven't modified classifier – just verified if robust
- **Idea:** Let's "smooth" our classifier to reduce input sensitivity
  - Imagine adv. examples as "spikes" in decision boundary
- Given a base classifier f, generate new, smoothed classifier g:



$$g(x) = \underset{c \in \mathcal{Y}}{\arg\max}\ \mathbb{P}(f(x+\varepsilon) = c) \quad \text{where } \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

i.e., g outputs class that f is most likely to output under Gaussian noise ptb

- [Cohen21] proves that g is robust around x within $R = \frac{\sigma}{2}(\Phi^{-1}(p_A) - \Phi^{-1}(p_B))$
- Propose to estimate probabilities by MC sampling of perturbed inputs

Certified Adversarial Robustness via Randomized Smoothing

Jeremy Cohen [1] Elan Rosenfeld [1] J. Zico Kolter [1 2]

K-Neuron Constraints

K-Neuron Constraints

- One contributor to the convex barrier is *individual* neuron relaxations
- Instead, [Singh19] considers relaxations of k ReLUs together
- Leads to smaller (i.e., less relaxed) feasible sets → tighter proofs
- Shown to scale to NNs with 100k neurons (e.g., ResNet for CIFAR-10)

Beyond the Single Neuron Convex Barrier for Neural Network Certification

Gagandeep Singh[1], Rupanshu Ganvir[2], Markus Püschel[1], Martin Vechev[1]
Department of Computer Science
ETH Zurich, Switzerland
[1]{gsingh,pueschel,martin.vechev}@inf.ethz.ch
[2]rganvir@student.ethz.ch

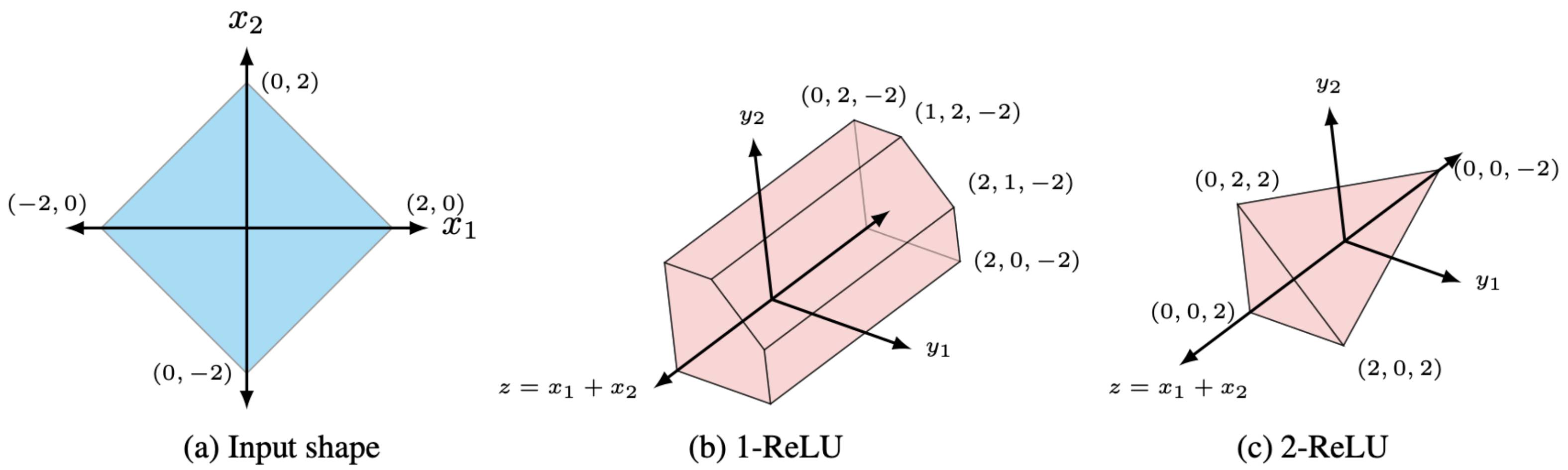


Figure 1: The input space for the ReLU assignments $y_1 := ReLU(x_1)$, $y_2 := ReLU(x_2)$ is shown on the left in blue. Shapes of the relaxations projected to 3D are shown on the right in red.

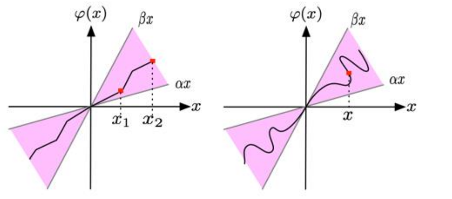
Semidefinite Programming (SDP) for NN Verification

SDP Verification

Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming

Mahyar Fazlyab[†], Manfred Morari, George J. Pappas

- So far, have used linear inequalities to relax nonlinearities
- **Alternative:** can often use Quadratic Constraints (QCs)



$\alpha \le \frac{\varphi(x_2)-\varphi(x_1)}{x_2-x_1} \le \beta$ $\qquad$ $\alpha \le \frac{\varphi(x)}{x} \le \beta$

Fig. 2: A slope-restricted nonlinearity (left) and a sector-bounded nonlinearity (right).

Definition 1 *Let $\mathcal{X} \subset \mathbb{R}^{n_x}$ be a nonempty set. Suppose $\mathcal{P}_{\mathcal{X}}$ is the set of all symmetric indefinite matrices P such that*

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^\top P \begin{bmatrix} x \\ 1 \end{bmatrix} \ge 0 \quad \textit{for all } x \in \mathcal{X}. \tag{4}$$

We then say that $\mathcal{X}$ satisfies the QC defined by $\mathcal{P}_{\mathcal{X}}$.

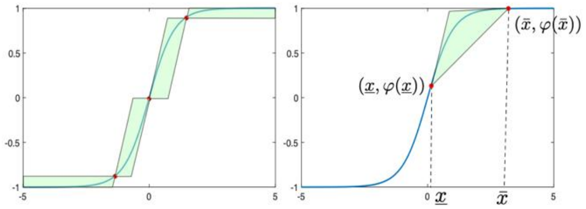
Lemma 3 (Global QC for ReLU **function)** *The function $\phi(x) = \max(\alpha x, \beta x)$ satisfies the QC*

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^\top \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{12}^\top & Q_{22} & Q_{23} \\ Q_{13}^\top & Q_{23}^\top & Q_{33} \end{bmatrix} \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \ge 0, \tag{22}$$

for all $x \in \mathbb{R}^n$, where

$Q_{11} = -2\alpha\beta(\mathrm{diag}(\lambda) + T)$, $Q_{12} = (\alpha+\beta)(\mathrm{diag}(\lambda) + T)$,
$Q_{13} = -\beta\nu - \alpha\eta$, $Q_{22} = -2(\mathrm{diag}(\lambda) + T)$,
$Q_{23} = \nu + \eta$, $Q_{33} = 0$,

$\nu, \eta \in \mathbb{R}_+^n$, and T is given by (17).



- Then, finding minimum-volume ellipsoid enclosing f(X) is an SDP[1]:

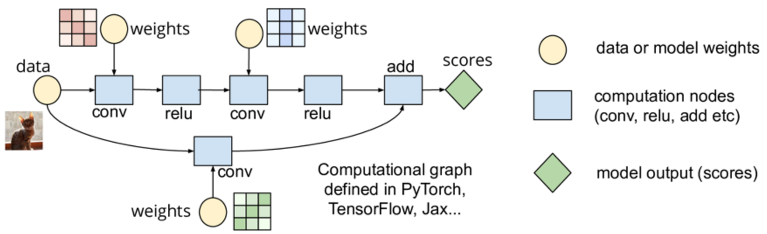
$$\begin{array}{ll} \text{minimize} & \log\det(A_y^{-1}) \\ \text{subject to} & M_{\text{in}}(P) + M_{\text{mid}}(Q) + M_{\text{out}}(S(A_y, b_y)) \preceq 0 \\ & (P, Q, A_y, b_y) \in \mathcal{P}_{\mathcal{X}} \times \mathcal{Q} \times \mathbb{S}^{n_f} \times \mathbb{R}^{n_f}. \end{array} \tag{39}$$

- Scalability / computation time can be challenging, but lots of sparsity to potentially leverage

[1]Need to use Schur complements to make this convex

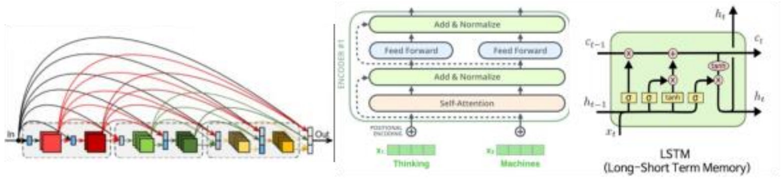
Software Libraries for NN Verification

auto_LiRPA



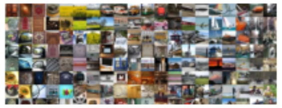
- PyTorch-based verification software
- You provide your computation graph, property to verify (e.g., specification vector, perturbation set), and choose verification algorithm
- Library knows how to propagate through and/or relax many of the commonly used PyTorch operations
- Includes graph traversal algorithm to handle arbitrary computation graphs
- You get to experiment with this library in HW2 ☺

General Neural Network Architecture



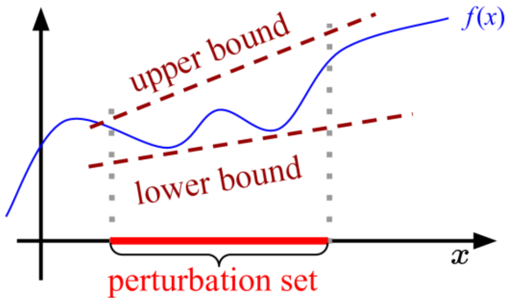
Scalable Certified Defense (up to 1000x faster)

IMAGENET



Automatic Provable Bounds (like auto diff)

`loss.backward()` ⟹ `loss.compute_bounds()`



Lower and upper bounds computed by `auto_LiRPA` given $f(x)$ defined in PyTorch

jax_verify

- Similar principle to auto-LiRPA, but based on JAX (instead of PyTorch)

```
output_bounds = jax_verify.verification_technique(network_fn, input_bounds)
```

- In my experience, the JIT-compilation of jax_verify works extremely well (>100x speedup), leading to very fast verification results
  - Can also easily deploy algorithms on CPU/GPU/TPU
  - (I haven't experimented with this on auto-LiRPA but think it's possible too)
- See HW2 ☺

Verification Techniques

The methods currently provided by `jax_verify` include:

- Functional Lagrangian Berrada et al 2021
- SDP-FO (first-order SDP verification, Dathathri et al 2020)
- Non-convex (Bunel et al 2020)
- Interval Bound Propagation (Gowal et al 2018, Mirman et al 2018)
- Backward Lirpa bounds such as CAP (Wong and Kolter 2017), FastLin(Weng et al 2018) or CROWN (Zhang et al 2018)
- Forward Lirpa bounds (Xu et al 2020)
- CROWN-IBP (Zhang et al 2019)
- Planet (also known as the "LP" or "triangle" relaxation, Ehlers 2017), currently using CVXPY as the LP solver
- MIP encoding (Cheng et al 2017, Tjeng et al 2019)

Recap

- Convex Barrier proposed theoretical framework & suggested reasons why many existing NN verification methods run into same limits
- More recently, many proposed algorithms aim to reduce gap
  - Branch and Bound (BaB)
  - Sampling-Based Methods
  - K-Neuron Constraints
  - SDP
- Discussed 2 excellent libraries that implement various algorithms
  - Can use these in your own research and stay up-to-date as field advances

Plan for Wednesday: Paper Discussion

- **Individual presentations:** Pick your paper and add it to the spreadsheet
- **Group discussion:** β-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification
  - https://arxiv.org/pdf/2103.06624.pdf
- **Next Lecture (9/27):** Training Robust Neural Networks