Geometric Path Enumeration Methods for Verifying ReLU Neural Networks

Stanley Bak

HCSS 2020

Stony Brook University

First international competition on neural network verification (VNN-COMP) happened this summer

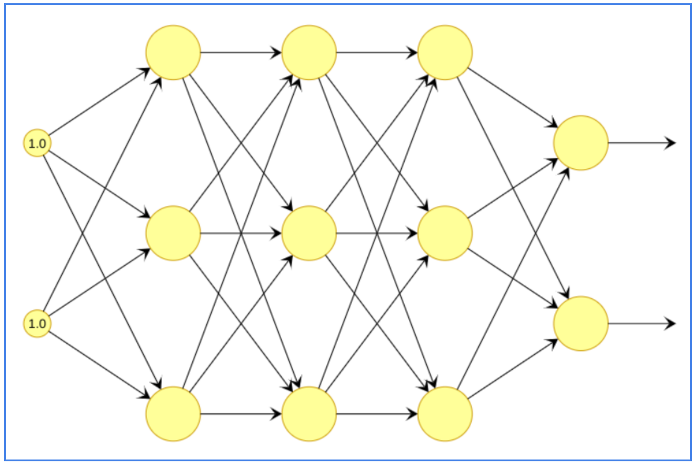
Pre-COVID, 18 participants intended to participate

1. NNV (Vanderbilt)
2. DNNV (U of Virginia)
3. **nnenum (Stony Brook)**
4. Neurify (Columbia)
5. ReluVal (Columbia)
6. CROWN-IBP (UCLA / MIT / Michigan / DeepMind / UIUC)
7. auto_LiRPA (Northeastern / Tsinghua / UCLA / Lawrence Livermore National Laboratory)
8. Sherlock (U of Colorado Boulder)
9. NPAQ (U of Singapore / Berkeley)
10. Branch-and-Bound (Oxford / DeepMind)
11. PaRoT from FiveAI (Cambridge / FiveAi)
12. MIPVerify.jl (MIT / Cruise Automation)
13. ARFramework (Utah State)
14. Marabou (Stanford / Hebrew U of Jerusalem)
15. Venus (Imperial College London)
16. Verinet (Imperial College London)
17. ERAN (ETH Zurich / UIUC)
18. PeregriNN (UC Irvine)

https://sites.google.com/view/vnn20/vnncomp

What is Meant by Neural Network Verification?

Input Set



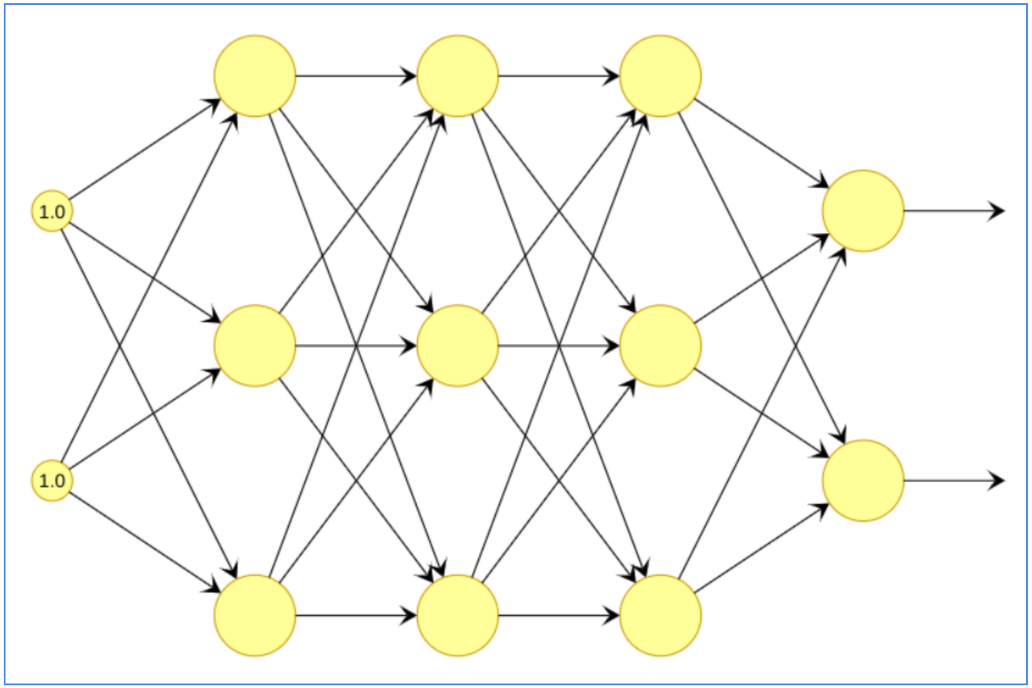
Output Set

$$i_1 \in [0, 1]$$
$$i_2 \in [0, 1]$$
$$\dots$$
$$i_n \in [0, 1]$$

$$o_1 \geq o_2$$
$$o_1 \geq o_3$$
$$\dots$$
$$o_1 \geq o_m$$

Two Operations Needed

Verification needs to reason over two types of operations: (1) affine transformations, and (2) activation functions.



Input Set → → → → → → → → Output Set

Affine Transform

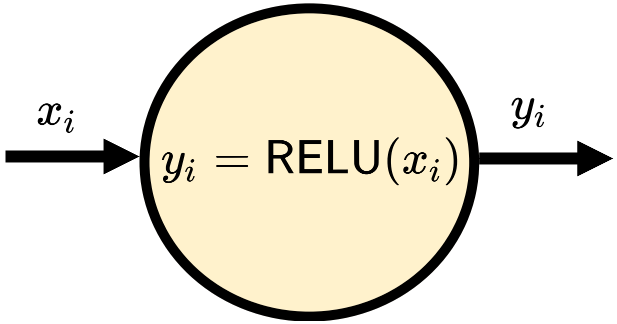
An **affine transformation** f is a function that transforms an n-dimensional point x to a q-dimensional point defined using a matrix A and vector b.

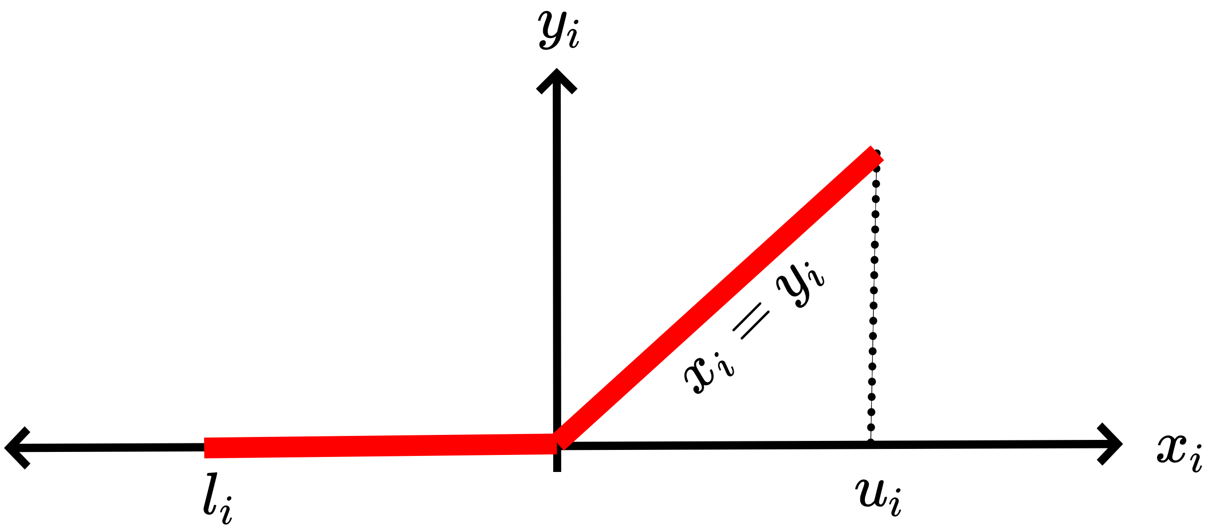
$$f(x) : \mathbb{R}^n \to \mathbb{R}^q$$
$$x \mapsto Ax + b$$

If x is a vector of n outputs of some layer, then the q inputs to the next layer are $Ax + b$, where A is the weights matrix and b is the bias vector.

ReLU Activation Functions

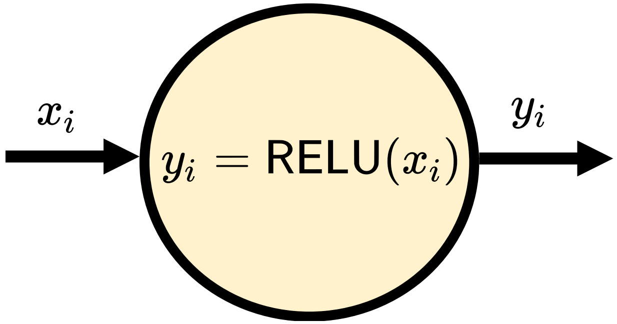
$$\mathsf{RELU}(x) = \max(x, 0)$$

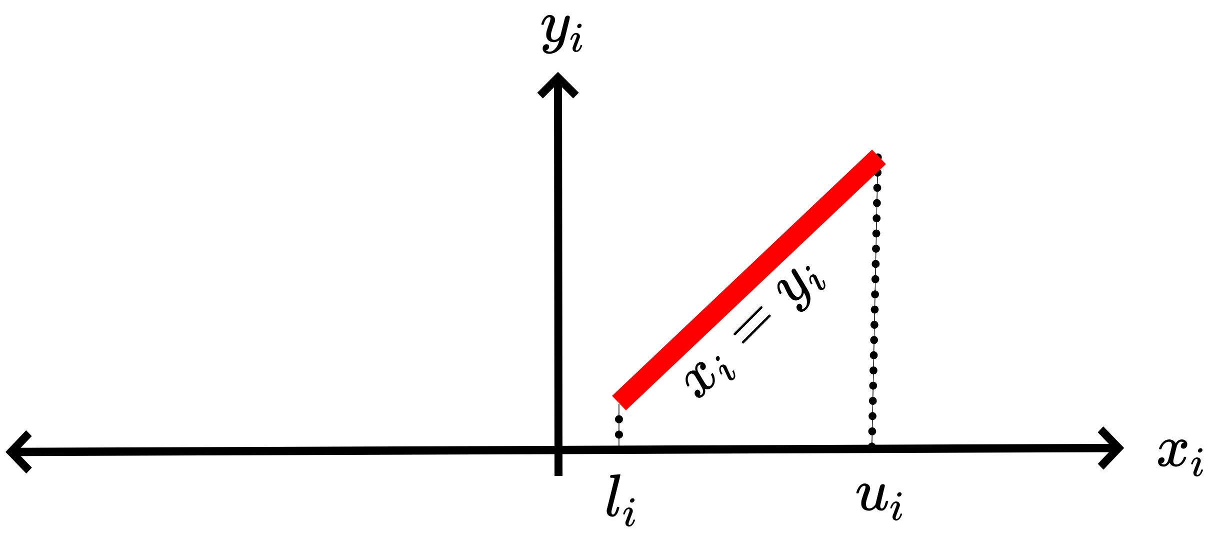




ReLU Activation Functions

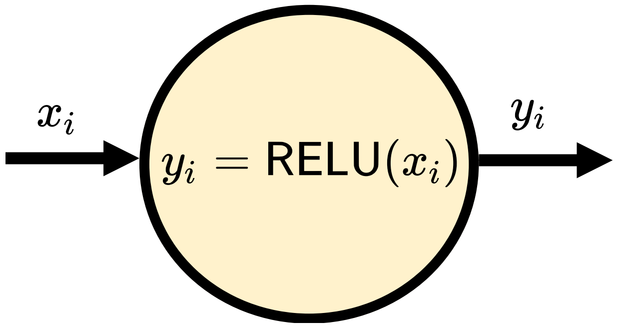
$$\mathsf{RELU}(x) = \max(x, 0)$$

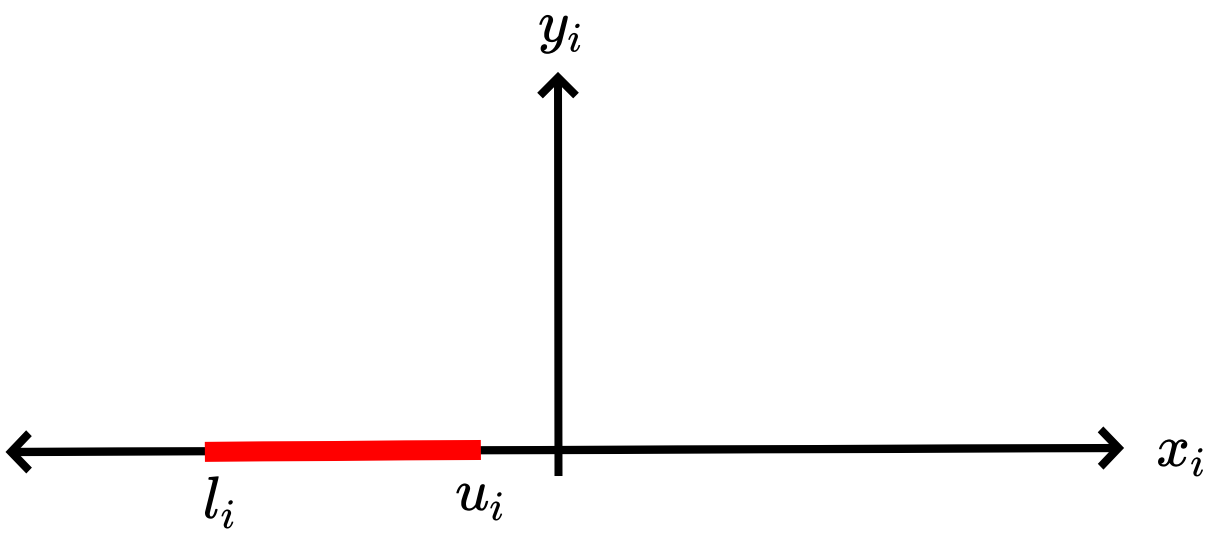




ReLU Activation Functions

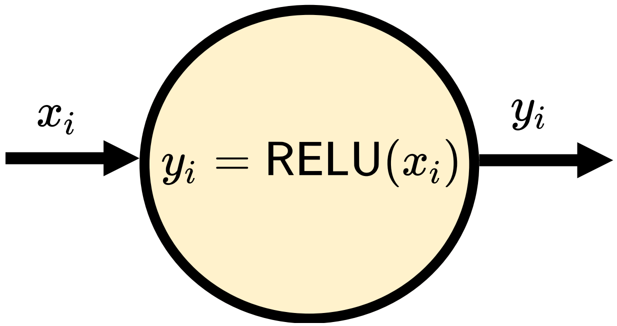
$$\text{RELU}(x) = \max(x, 0)$$

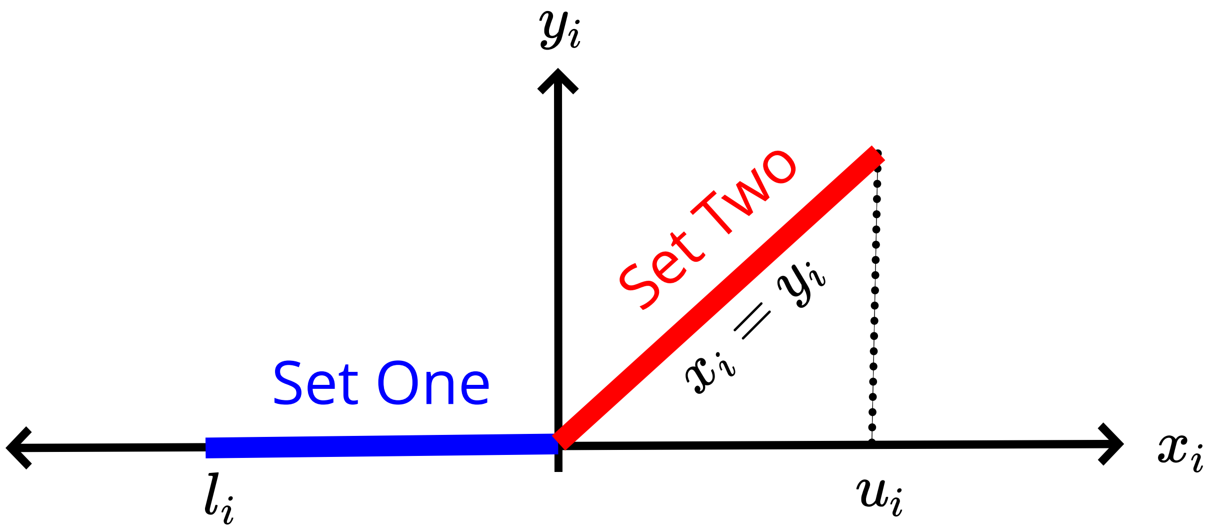




ReLU Activation Functions

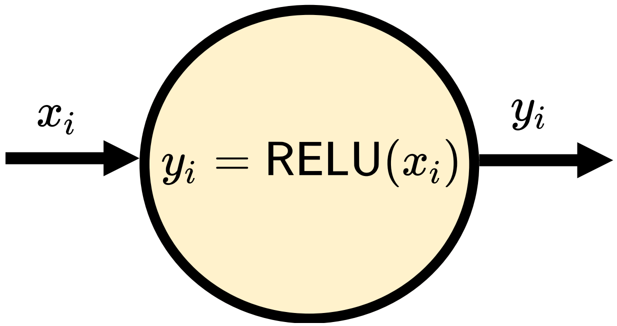
$$\text{RELU}(x) = \max(x, 0)$$

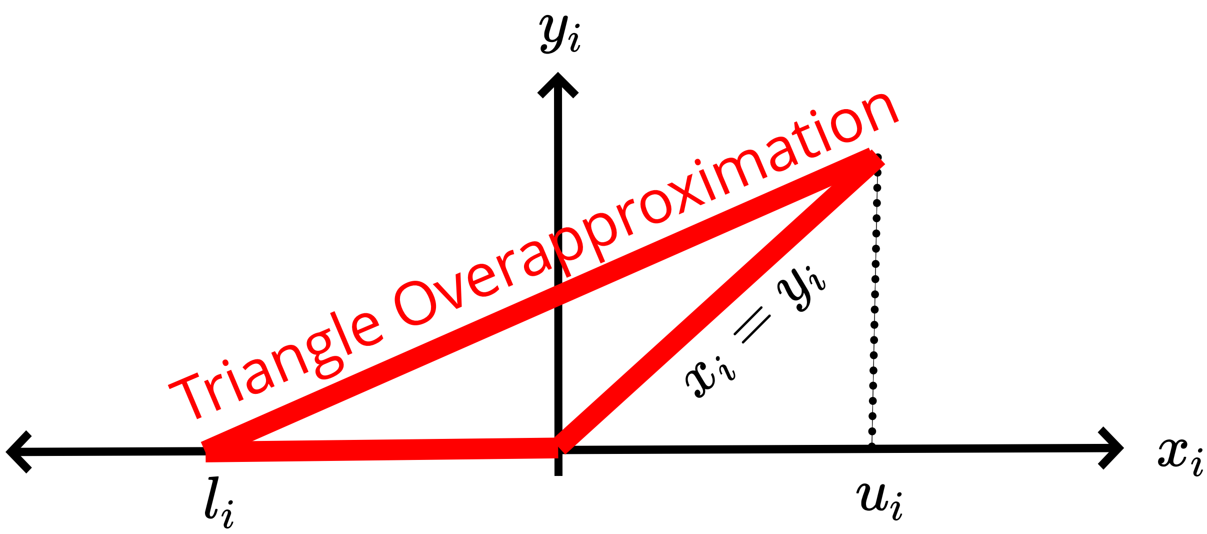




ReLU Activation Functions

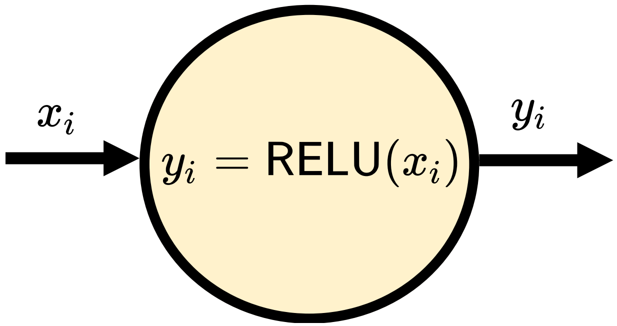
$$\text{RELU}(x) = \max(x, 0)$$

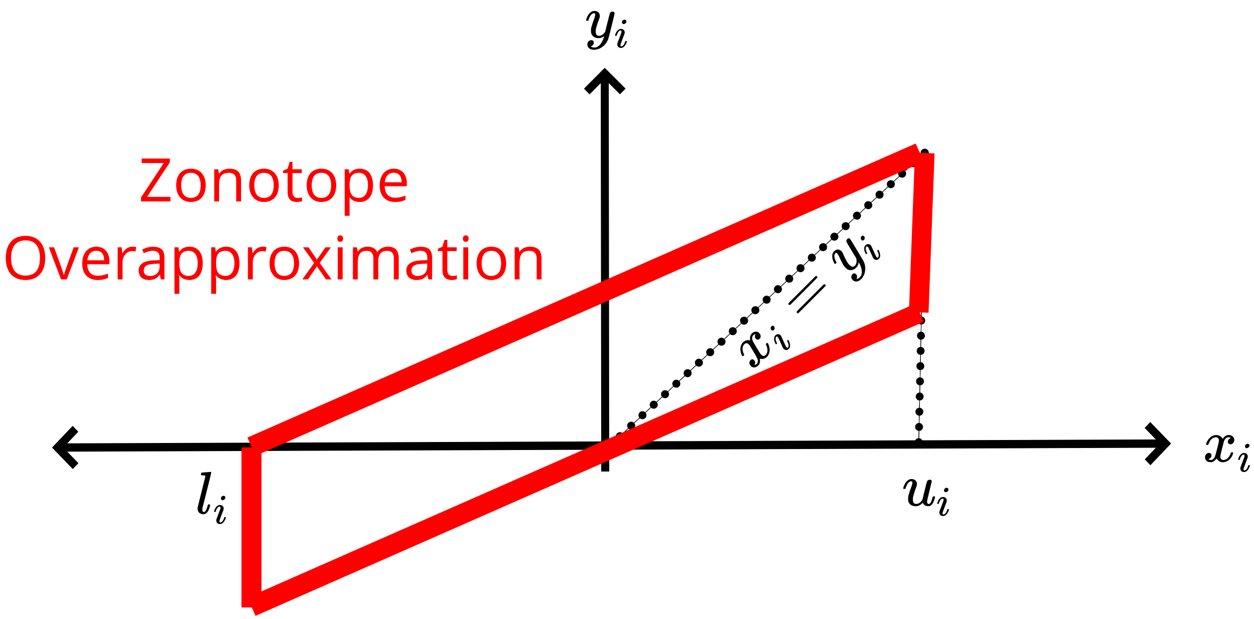




ReLU Activation Functions

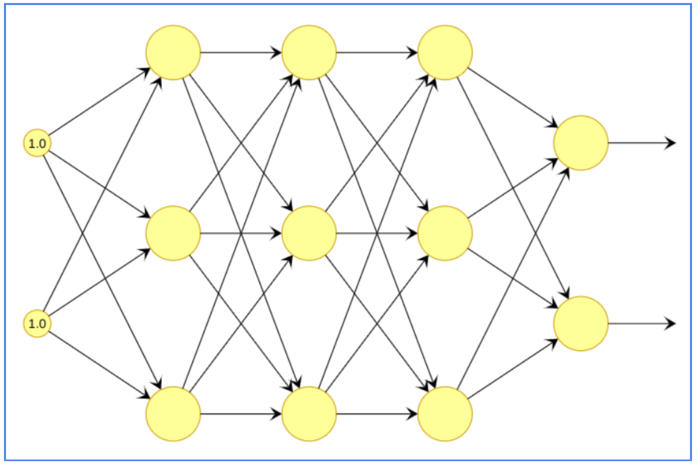
$$\text{RELU}(x) = \max(x, 0)$$





Set Operations are Needed for Verification

Input Set



Output Set

We need to be able to efficiently perform operations on <u>sets</u>:

- **Affine Transformation**
- **Optimization**
- **Intersection**

Representations for Subsets of $\mathbb{R}^n$

The set representation determines what operations are possible and efficient.

Some options:

- Boxes
- $\mathcal{V}$-Polytopes
- $\mathcal{H}$-Polytopes
- Zonotopes
- Linear Star Sets ($\mathcal{AH}$-Polytopes)

Operations on Star Set $\langle c, V, P \rangle$

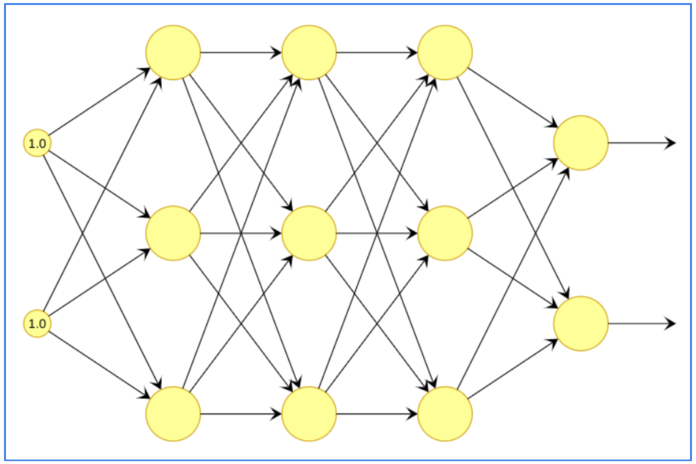
Affine Transform: matrix-matrix multiplication to compute c' and V'. Result is $\langle c', V', P \rangle$.

Optimization: put star set definition into a linear program (LP) and minimize.

Intersection: given a halfspace $H = \{x \mid Gx \leq g\}$, let $P_H(\alpha) = GV\alpha \leq g - Gc$. Result is $\langle c, V, P \wedge P_H \rangle$.

Star Sets for Verification

Input Set



Output Set

Star Sets exactly and efficiently encode linear transformation, optimizations and intersections.

This means **exact analysis is possible** for NNs with ReLUs, fully connected layers, convolutional layers, avg / max pooling layers.

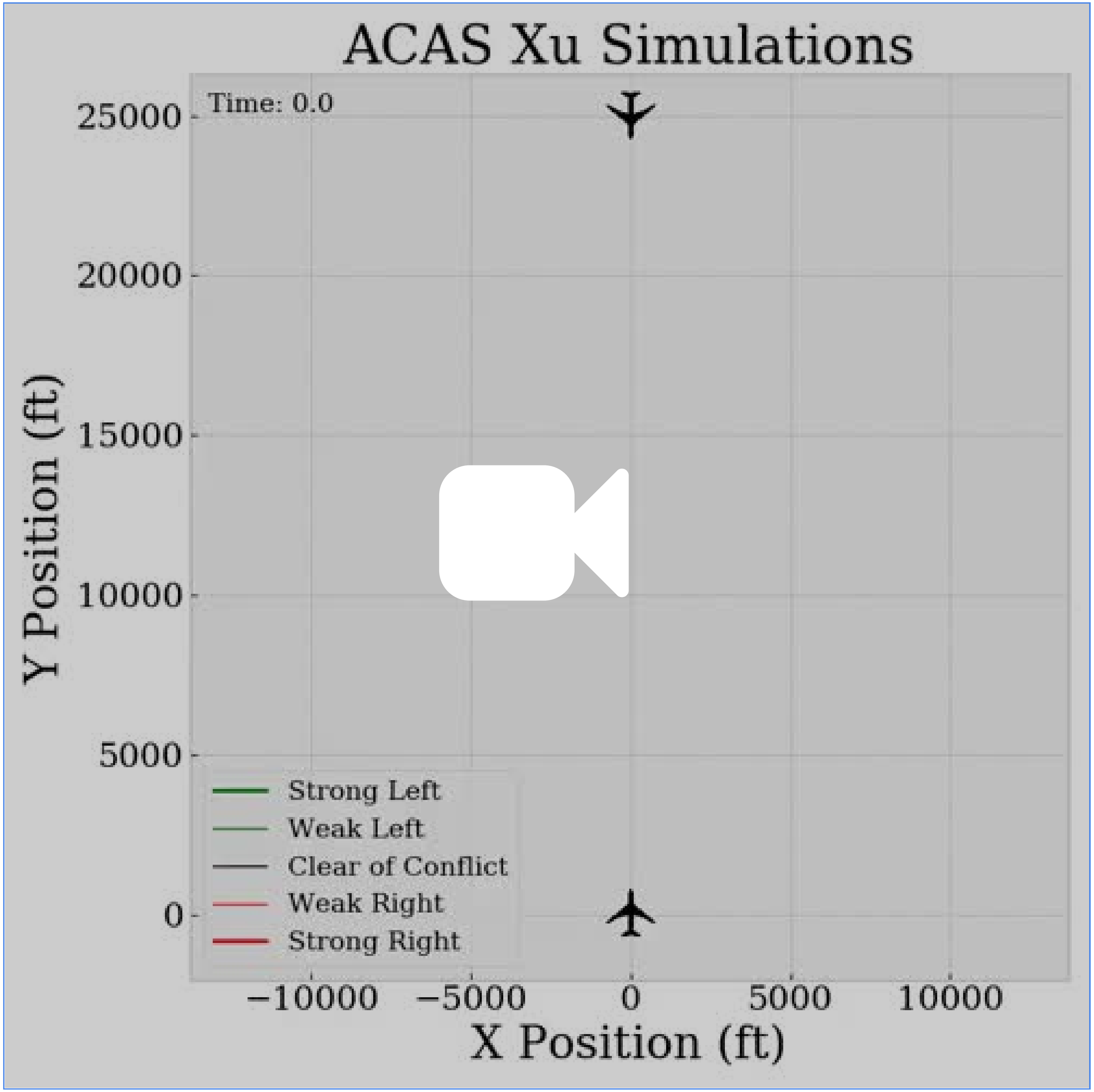
NN Verification is NP-Complete

Every ReLU neuron can in theory **double** the number of sets.

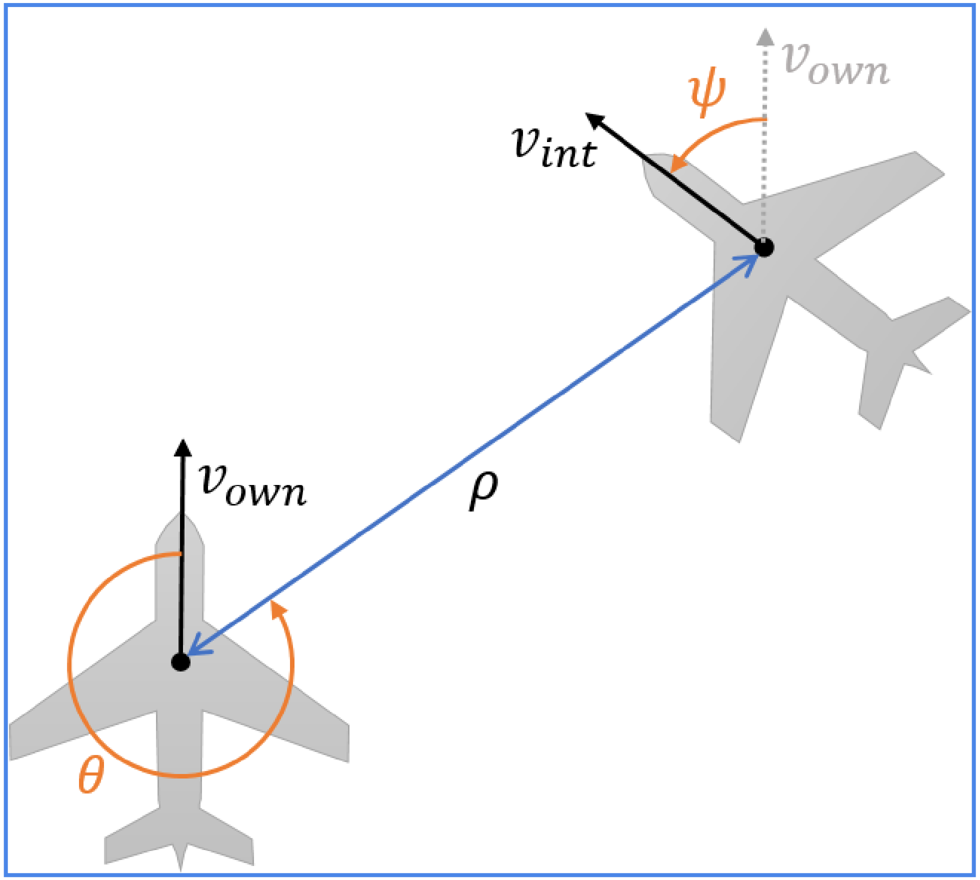
Example: A 300 neuron network could require 2^{300} sets.

Does this happen in practice?

Need to define "in practice".


ACAS Xu Simulations
Time: 0.0
25000
20000
15000
10000
5000
0
Y Position (ft)
Strong Left
Weak Left
Clear of Conflict
Weak Right
Strong Right
−10000
−5000
0
5000
10000
X Position (ft)

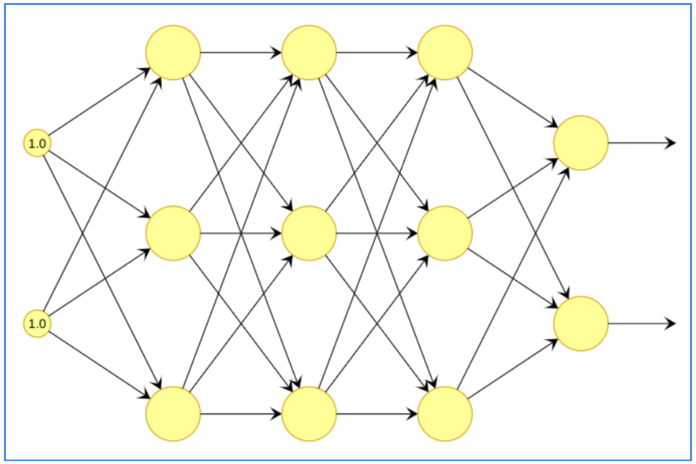
ACAS Xu Collision Avoidance System [Katz '17]



Why NN?: Replace a several GB lookup table with 45 neural networks (compression)

ACAS Xu Collision Avoidance System [Katz '17]

Inputs:
1. v_{int}
2. v_{own}
3. ρ
4. ψ
5. θ



300 neurons in 6 layers

Outputs:
1. Clear
2. Weak-Left
3. Weak-Right
4. Strong-Left
5. Strong-Right

Property φ_3: If the intruder is directly ahead and is moving towards the ownship, a turn will be commanded.

Input: $1500 \leq \rho \leq 1800, |\theta| \leq 0.06, \psi \geq 3.1, v_{own} \geq 980, v_{int} \geq 960$

Unsafe Output: Clear $\leq$ Weak-Left $\wedge$ Clear $\leq$ Weak-Right $\wedge$ Clear $\leq$ Strong-Left $\wedge$ Clear $\leq$ Strong-Right

Formal Methods "at Scale"

"Engineering matters: you can't properly evaluate a technique without an efficient implementation."

-Ken McMillan

Optimizations

To improve performance, you must first find the bottleneck of the algorithm.

The majority of the runtime is spent making unnecessary copies.

Optimizations

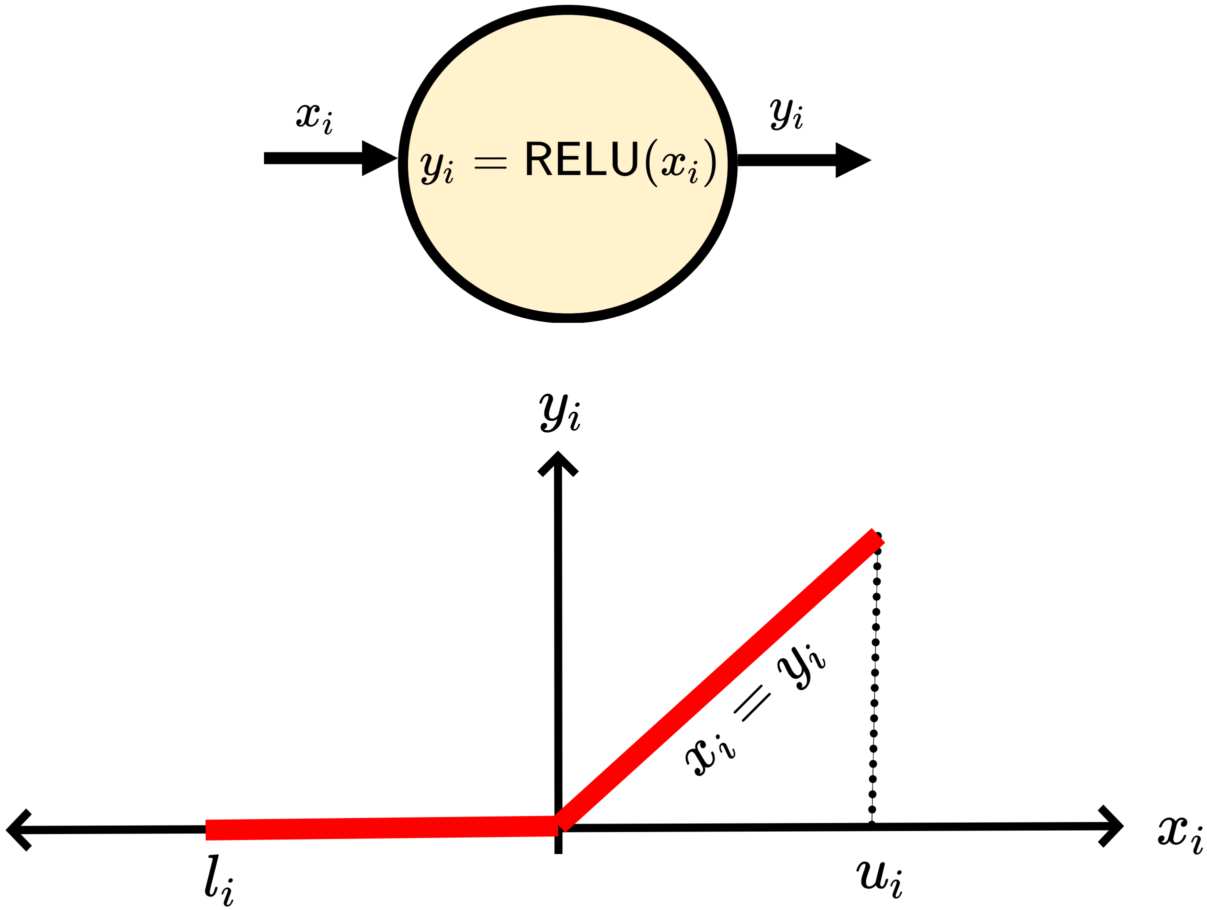
To improve performance, you must first find the bottleneck of the algorithm.

~~The majority of the runtime is spent making unnecessary copies.~~

The majority of the runtime is spent optimizing (solving LPs), to find the input bounds for each neuron.

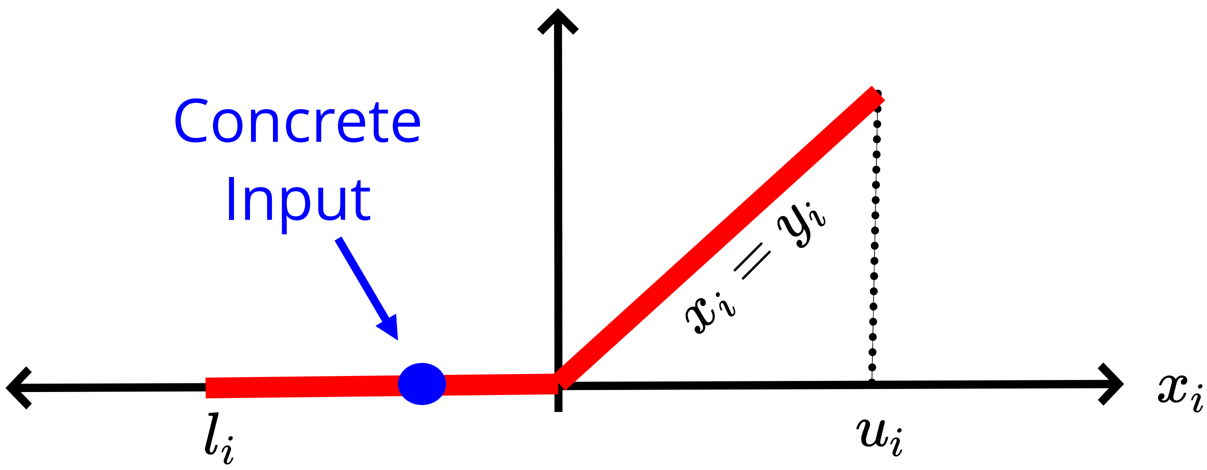
ReLU Activation Functions

$$\mathsf{RELU}(x) = \max(x, 0)$$



Two LPs are solved to find l_i and u_i for each neuron.

Observations



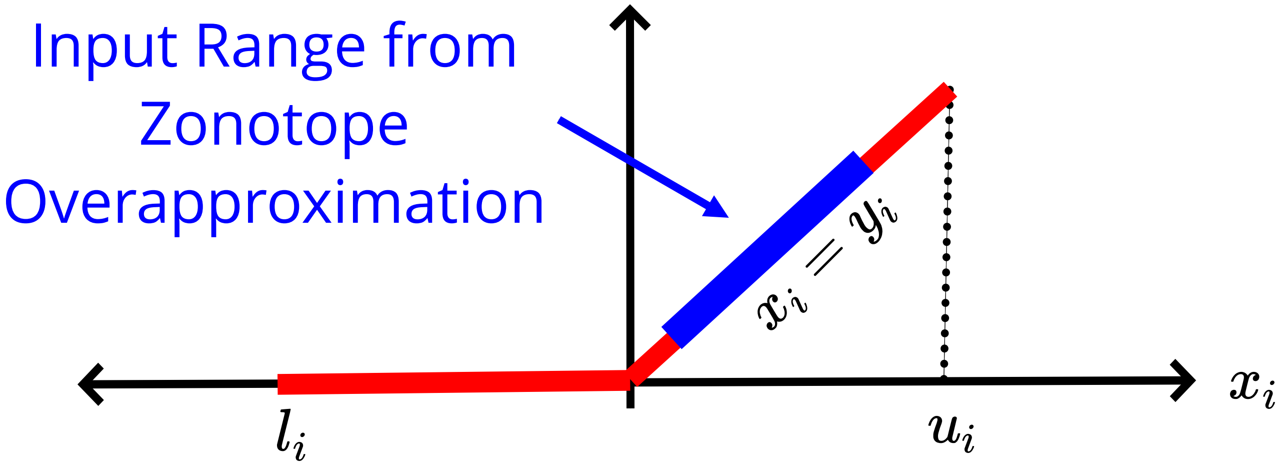
Actually, we don't usually need to compute l_i and u_i, just to check if $l_i < 0 < u_i$.

If l_i > 0, we're done (single LP)!

Also, if u_i < 0, we're done... how to choose direction?

Idea #1: use a concrete execution of the NN

Furter LP Reductions



LP solving is still the bottleneck, can we do better than a single LP per neuron?

In formal verification, achieving high performance means using the appropriate level of abstraction

Idea #2: Use Zonotope overapproximations to prove branching is possible without LP solving

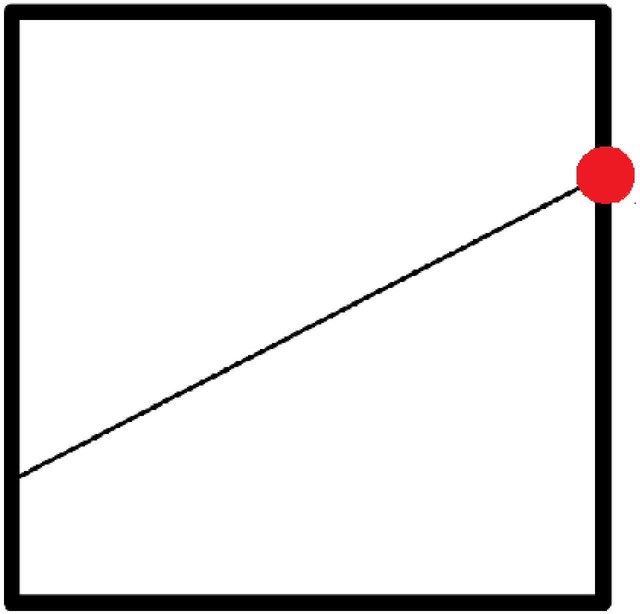
Zonotope Accuracy

LP solving is still the bottleneck, how can we do better?

The zonotope prefilter works better if it's more accurate. How can we increase it's accuracy?

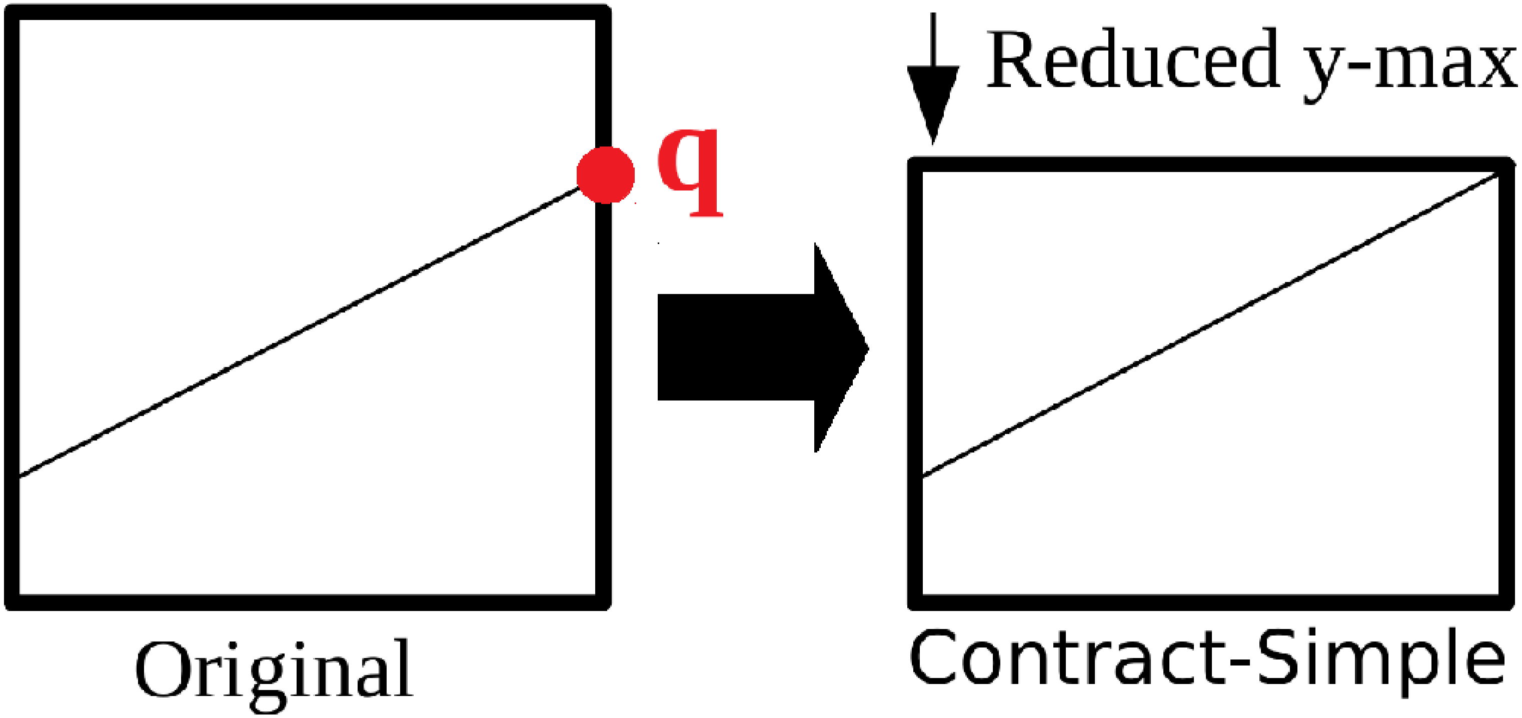
Idea #3: <u>Contract</u> the domain of the zonotope overapproximation when splitting.

Zonotope Domain Contraction

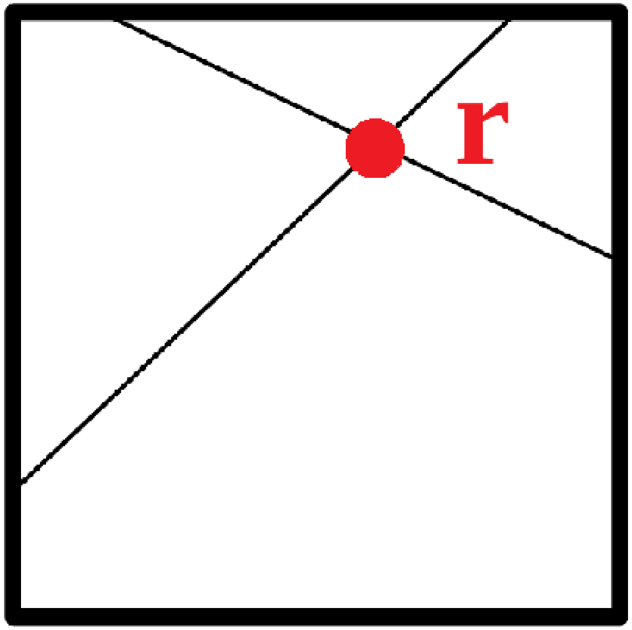


Original

Zonotope Domain Contraction

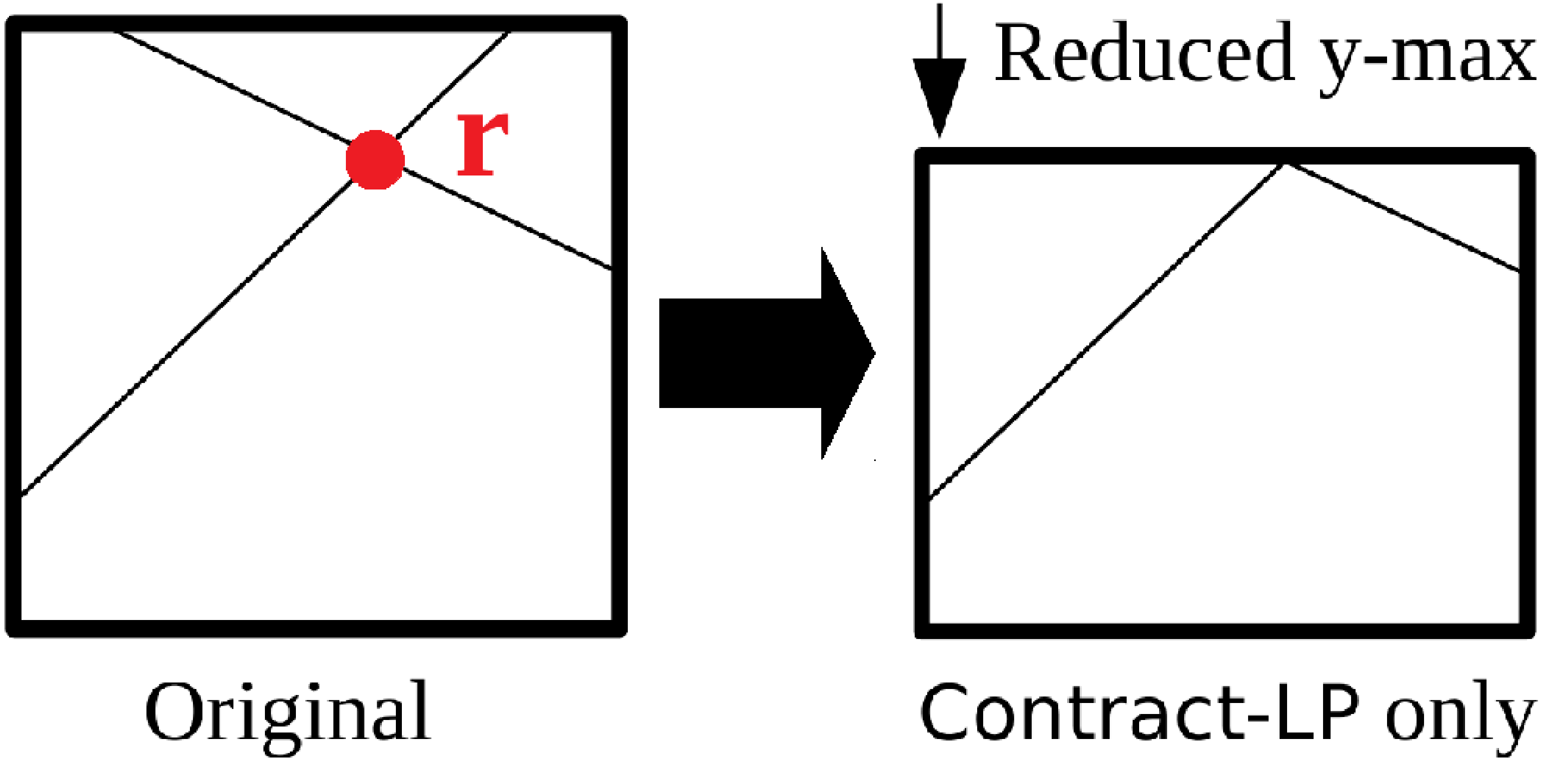


Zonotope Domain Contraction 2



Original

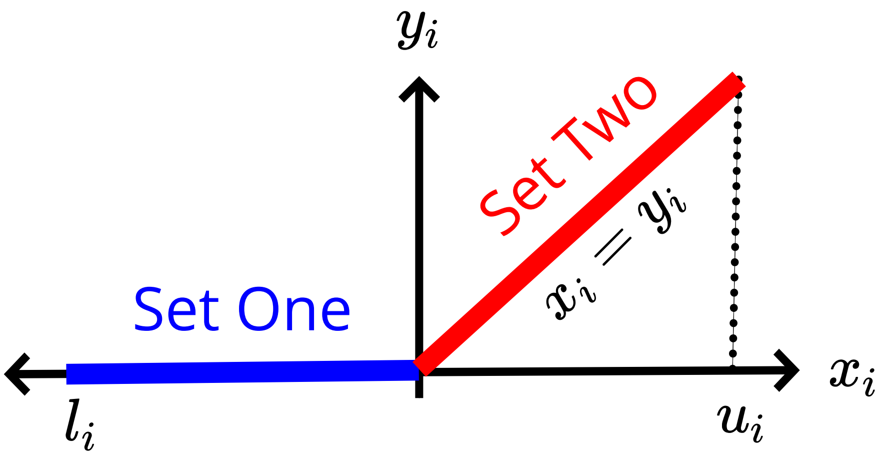
Zonotope Domain Contraction 2

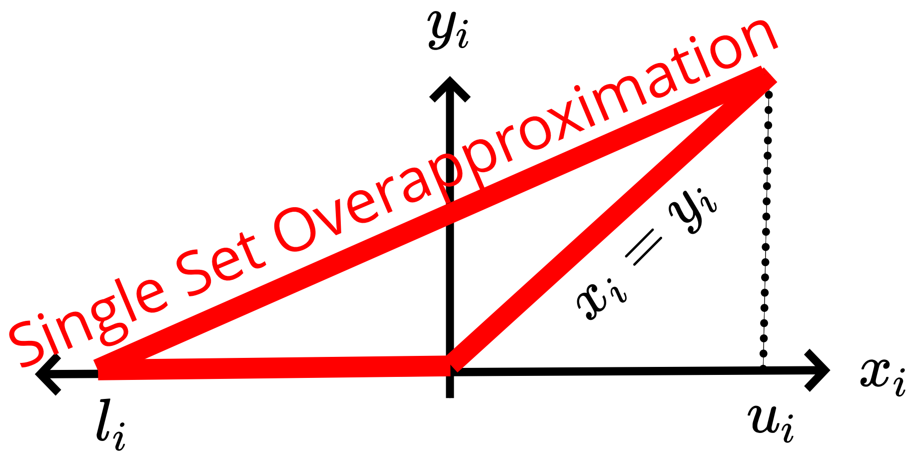


Exact vs Overapproximation

For each ReLU with $l_i < 0$ and $u_i > 0$, you can choose between splitting (exact) or single-set triangle overapproximation.

Neither is always best.





In formal verification, achieving high performance means using the appropriate level of abstraction

Idea #4: Combine splitting and overapproximation. Challenge: how to choose?

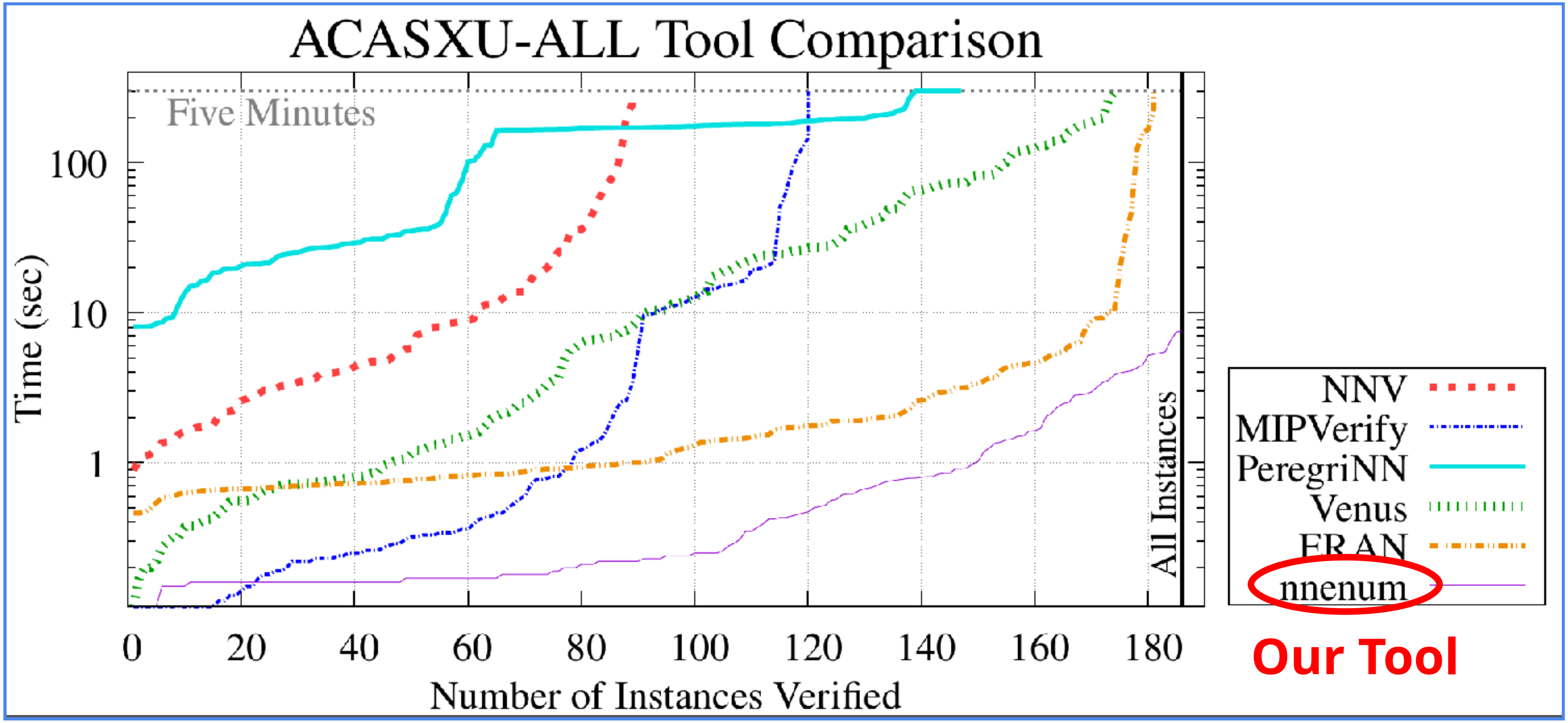
Results from VNN-COMP 2020

186 Benchmarks from ACASXu System

Original runtimes in 2017 paper were seconds to days, with some unsolved instances

Six tools submitted results

Results from VNN-COMP 2020



Results from VNN-COMP 2020

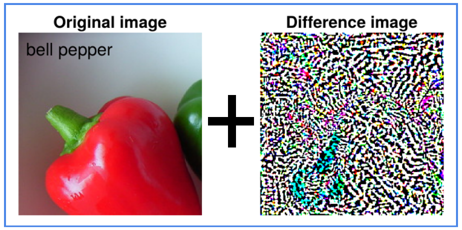
Table 2: Tool Runtime (sec) for ACASXU-HARD.

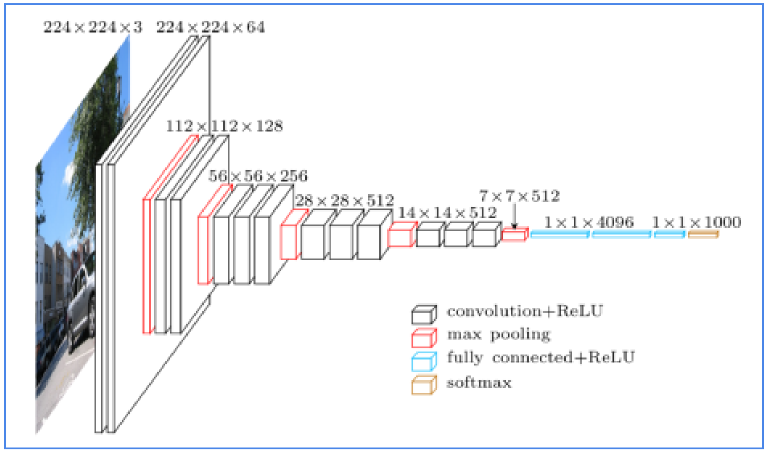
Our Tool

Prop	Net	Result	nnenum	NNV	PeregriNN	MIPVerify	Venus	ERAN
1	4-6	UNSAT	5.30	-	3191.34	-	179.98	5.38
1	4-8	UNSAT	3.96	-	2568.02	-	372.11	3.69
2	3-3	UNSAT	7.46	-	-	-	294.53	167
2	4-2	UNSAT	7.59	-	-	-	648.57	230
2	4-9	SAT	0.17	-	-	37.42	446.13	6.0
2	5-3	SAT	0.85	9302	-	5390.41	9.63	9.7
3	3-6	UNSAT	0.22	7.38	178.48	0.64	3.36	0.72
3	5-1	UNSAT	0.43	35.07	181.43	1.30	27.13	1.78
7	1-9	SAT	1.50	-	-	-	8010.49	91.4
9	3-3	UNSAT	2.52	13326.19	1121.38	-	1795.17	9.21

Other Verification Problems

Larger Perception NNs



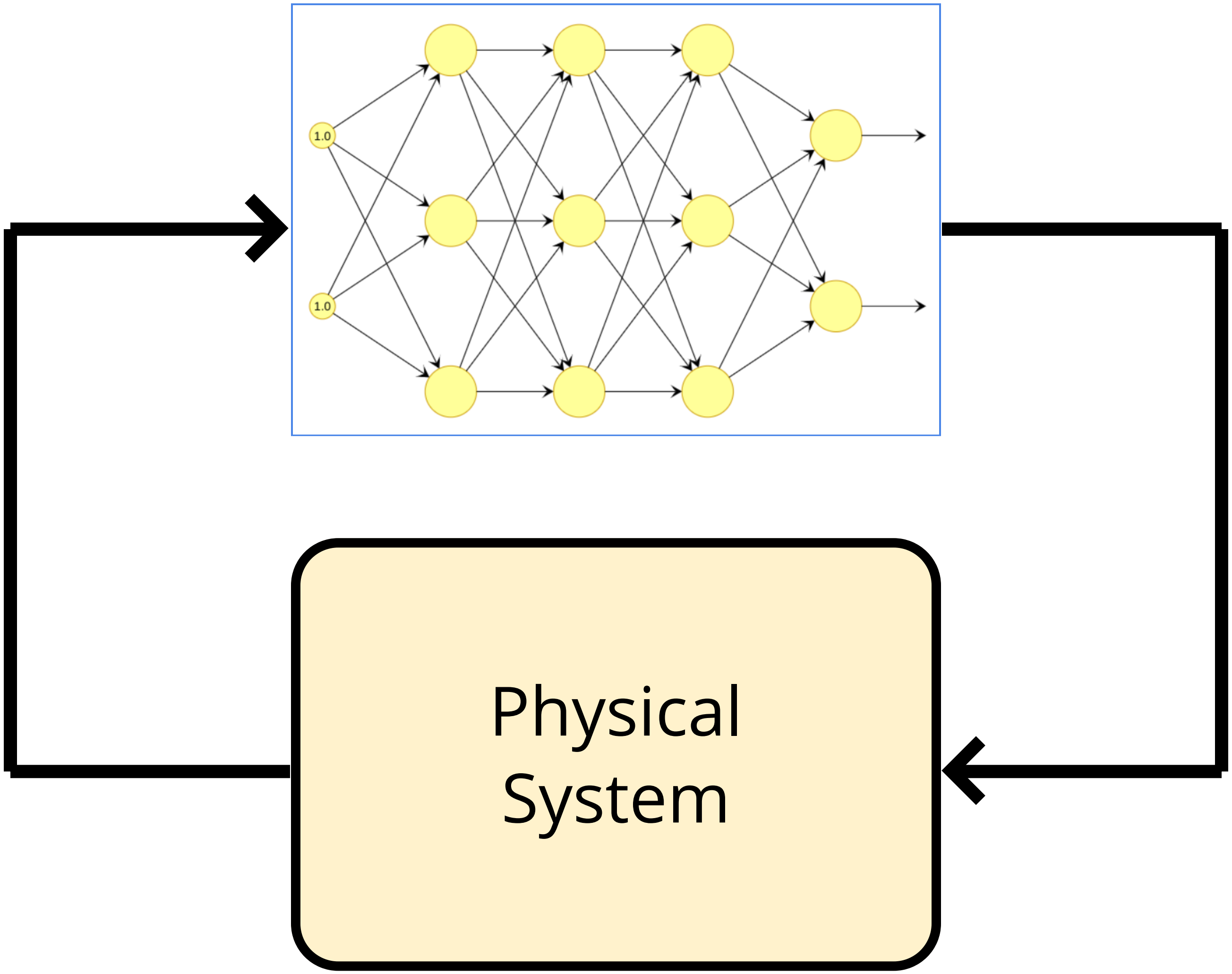


Bell Pepper?

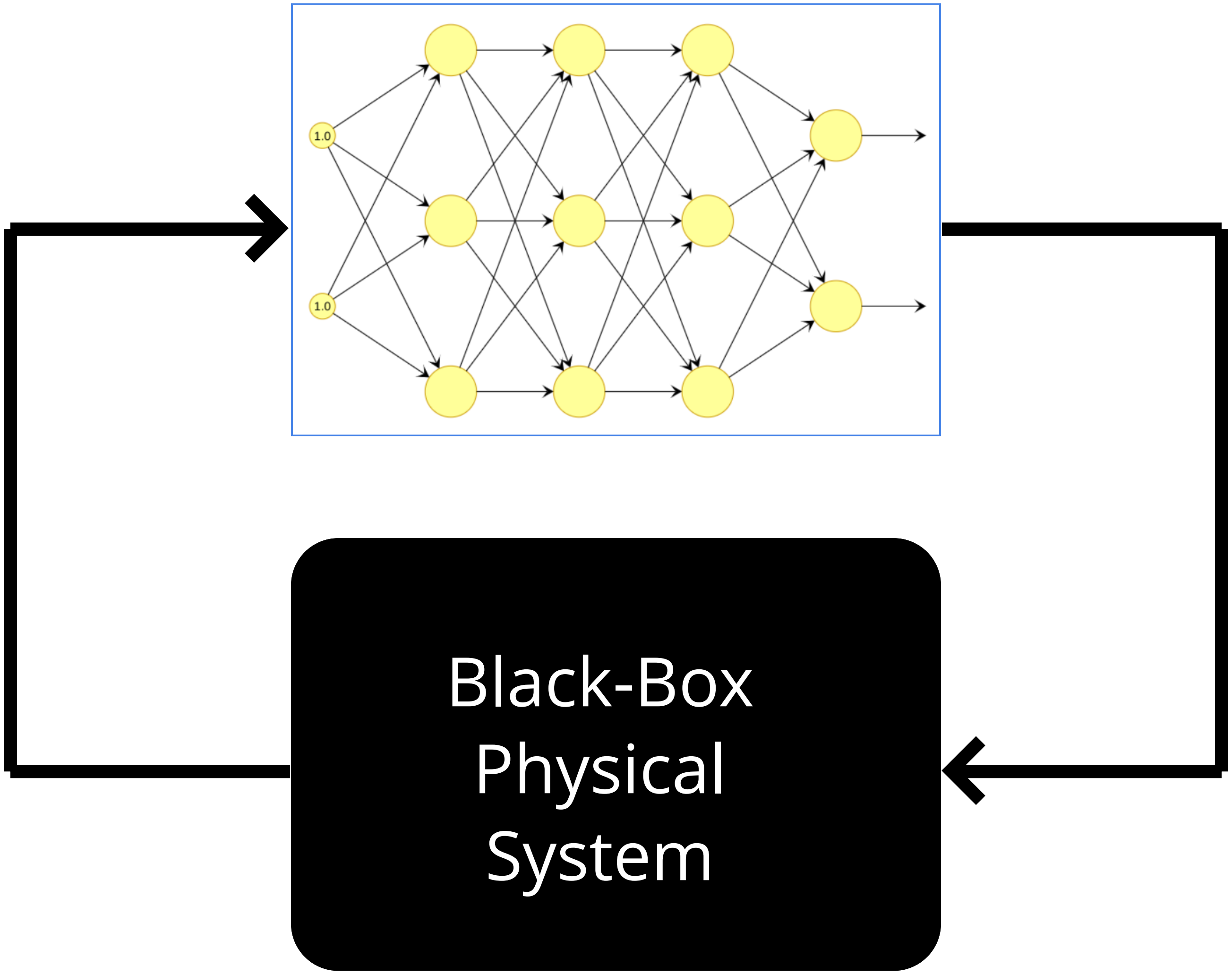
VGG-16
(>10 million neurons)

See the CAV 2020 paper:
"Verification of Deep Convolutional Neural Networks Using ImageStars"
H.D Tran, S. Bak, W. Xiang and T. T. Johnson

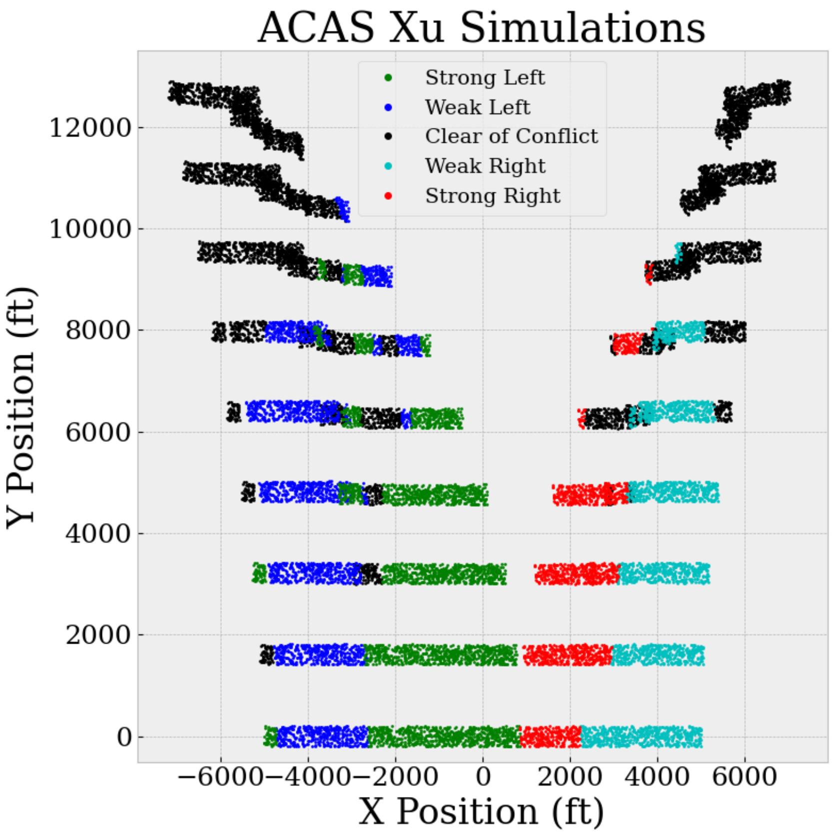
Closed-Loop Analysis



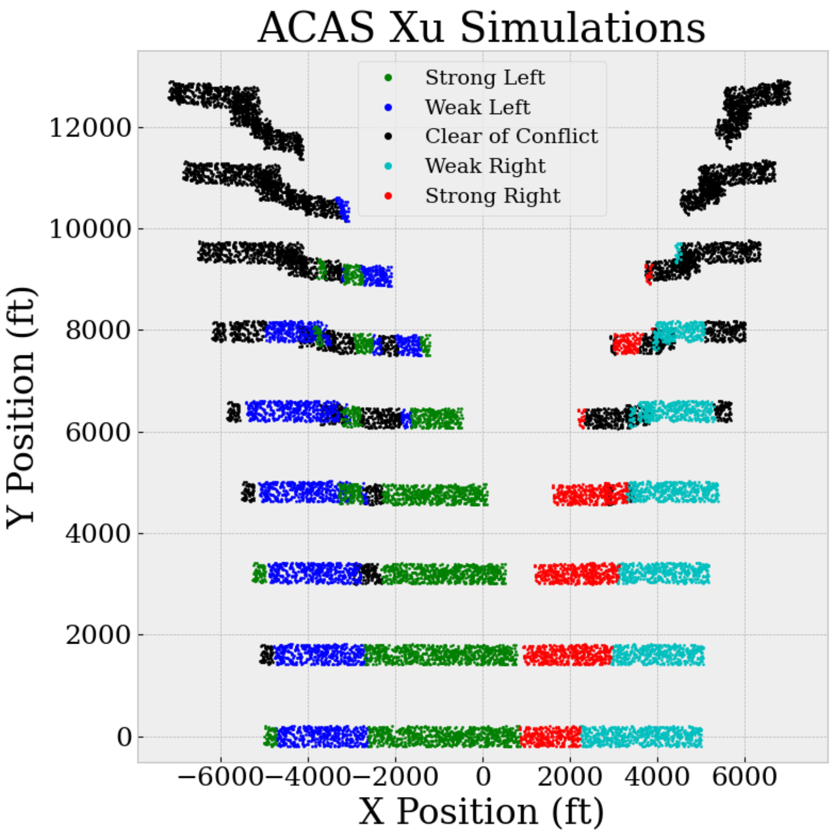
Closed-Loop Analysis

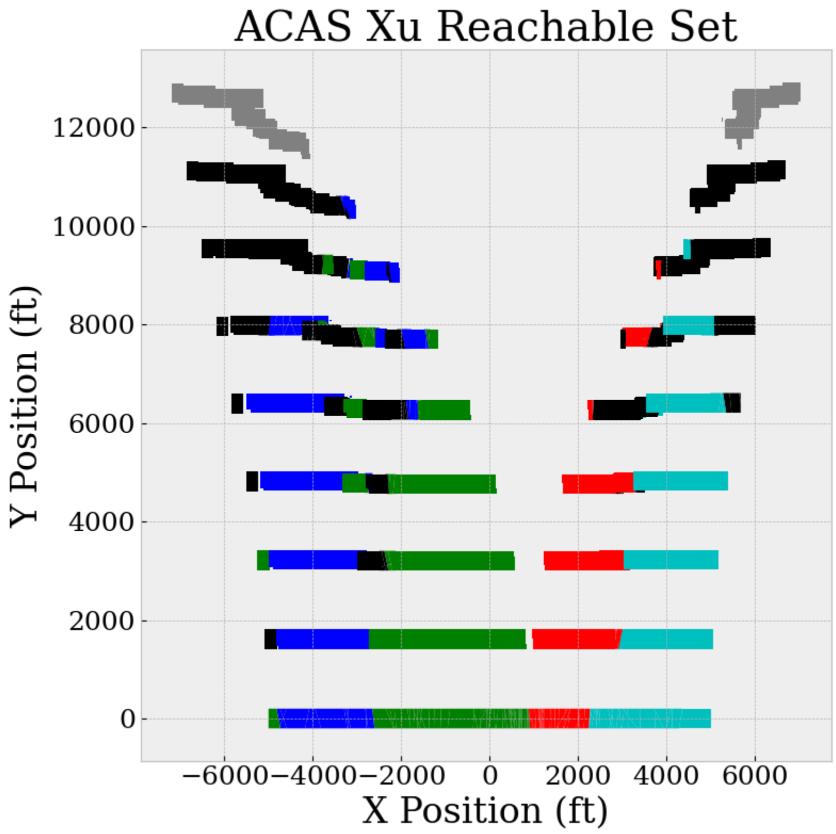


Decision Points



Decision Points





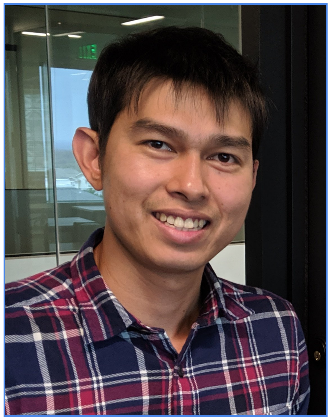
From black-box analysis with local numerical linearization

Summary

Verification of neural networks is becoming increasingly feasible.

Now is a good opportunity for collaboration: stanley.bak@stonybrook.edu

Stanley Bak



Hoang-Dung Tran

Kerianne Hobbs

Taylor Johnson