Certification in Deep Neural Networks

Rerun of:

Verification of Deep Neural Networks

Guy Katz

The Hebrew University of Jerusalem

ForMaL Spring School
June 5, 2019

RELUPLEX

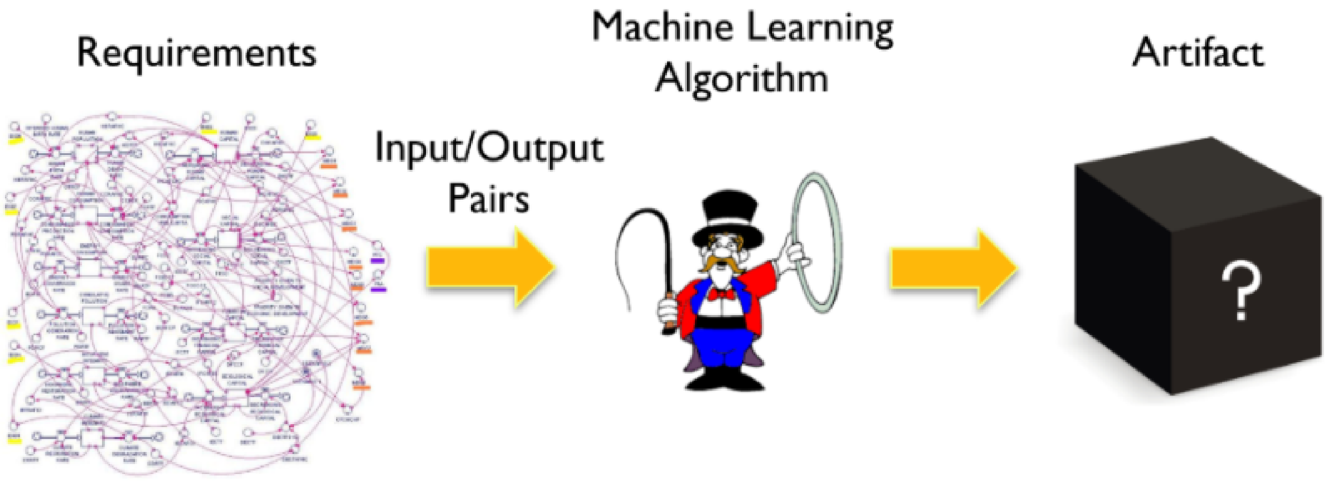
Safe and Robust Deep Learning

Gagandeep Singh
PhD Student
Department of Computer Science

ETH Zürich

AI^2: Abstract Interpretation for AI

See slides and more talks at:
https://formal-paris-saclay.fr/



Guarantees?

Supervised learning
Needs a lot of annotated data

Adversarial Inputs

- In 2014, an intriguing property was observed:

Goodfellow et al., 2015

- *Small perturbations* of inputs lead to misclassification
- Can usually find such inputs *very* easily

Attacks on Deep Learning

The self-driving car incorrectly decides to turn right on Input 2 and crashes into the guardrail

(a) Input 1 (b) Input 2 (darker version of 1)

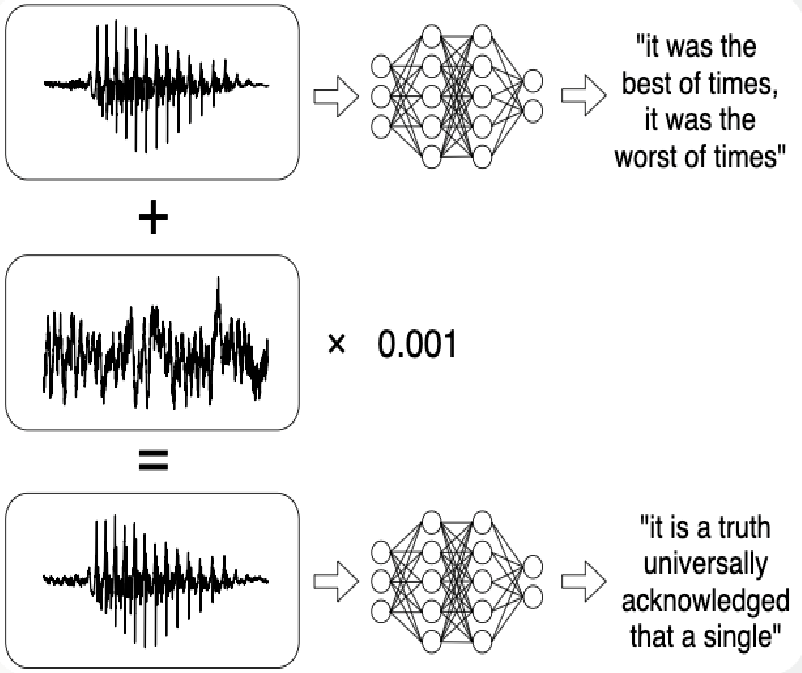
DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP'17

The Ensemble model is fooled by the addition of an adversarial distracting sentence in blue.

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Adversarial Examples for Evaluating Reading Comprehension Systems, EMNLP'17

Adding small noise to the input audio makes the network transcribe any arbitrary phrase

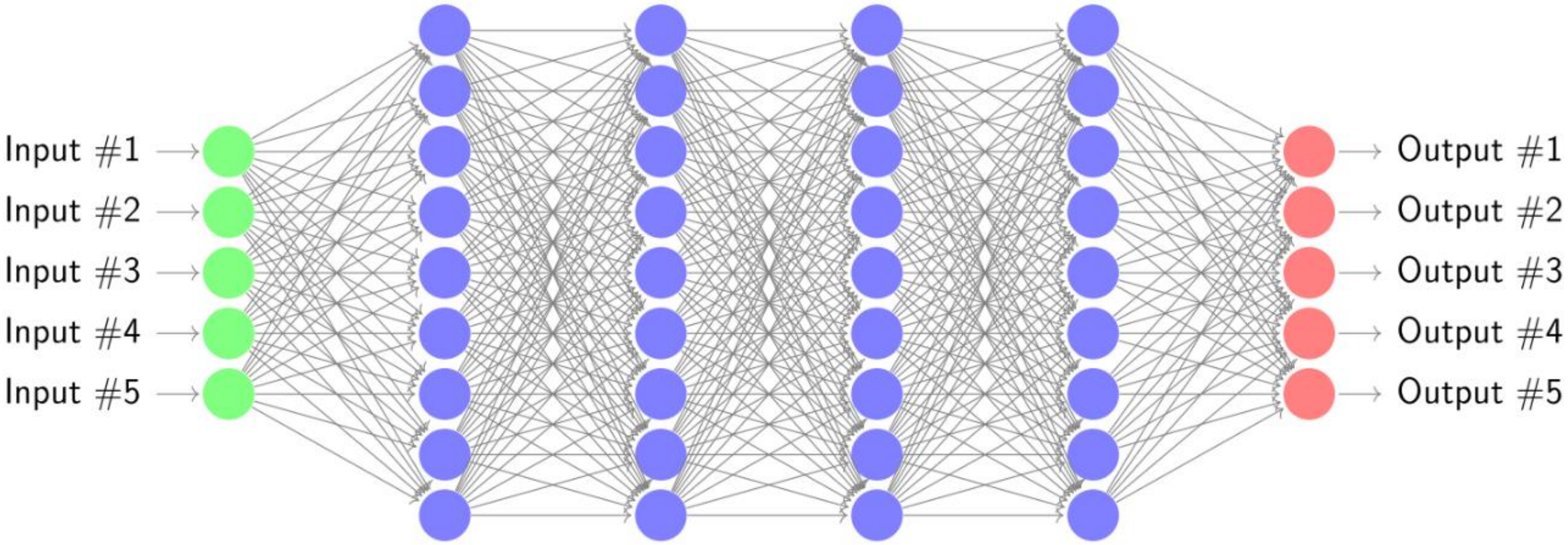


Audio Adversarial Examples: Targeted Attacks on Speech-to-Text, ICML 2018

Adversarial Robustness

- A network's resilience to adversarial attacks is called *adversarial robustness*
- There exist hardening techniques for increasing robustness
- But...
  - These usually defend against *existing* attacks
  - And then a *new* attack breaks them
- Verification can be used to establish robustness *guarantees*

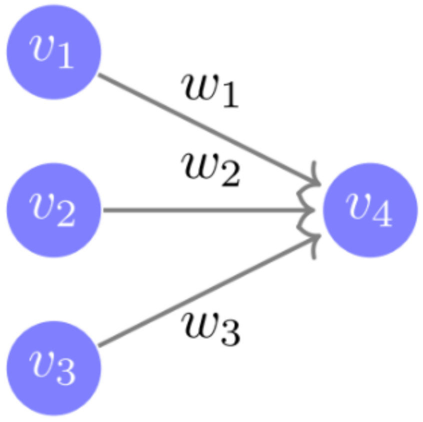
Neural Networks



- Typical sizes (number of neurons): between few hundreds and millions

Evaluating Neural Networks

- Nodes evaluated layer by layer:
  - Input layer is given
  - Every layer computed from its predecessor, according to *weights* and *activation functions*

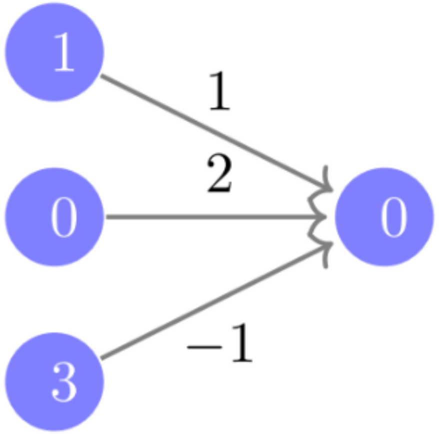


$$v_4 = f\left(\sum_{i=1}^{3} w_i \cdot v_i\right)$$

can be Non-Linear

Linear part

Activation Functions



$$1 \cdot 1 + 0 \cdot 2 + 3 \cdot (-1) = -2$$

- Rectified Linear Unit (ReLU): $f(x) = \max(x, 0)$
  - *Active* phase: $x \geq 0$, output is x
  - *Inactive* phase: $x < 0$, output is 0.

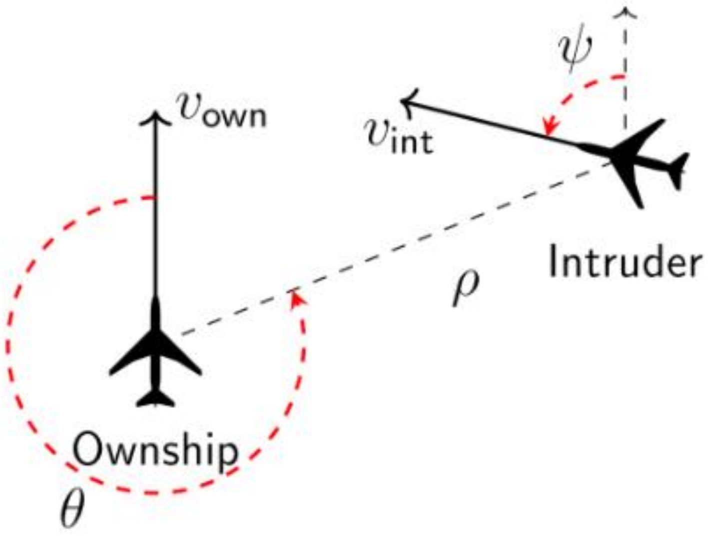
Mostly Non-Linear functions

- $\text{ReLU}(x) = \max(x, 0)$
- $\max(x, y) = \text{ReLU}(x - y) + y$

- Pooling layers:
  - Max pooling: $f(x_1, \ldots, x_n) = \max(x_1, \ldots, x_n)$
  - Average pooling: $f(x_1, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} x_i$ Linear
- Sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$
- Hyperbolic tangent function: $f(x) = \tanh(x)$

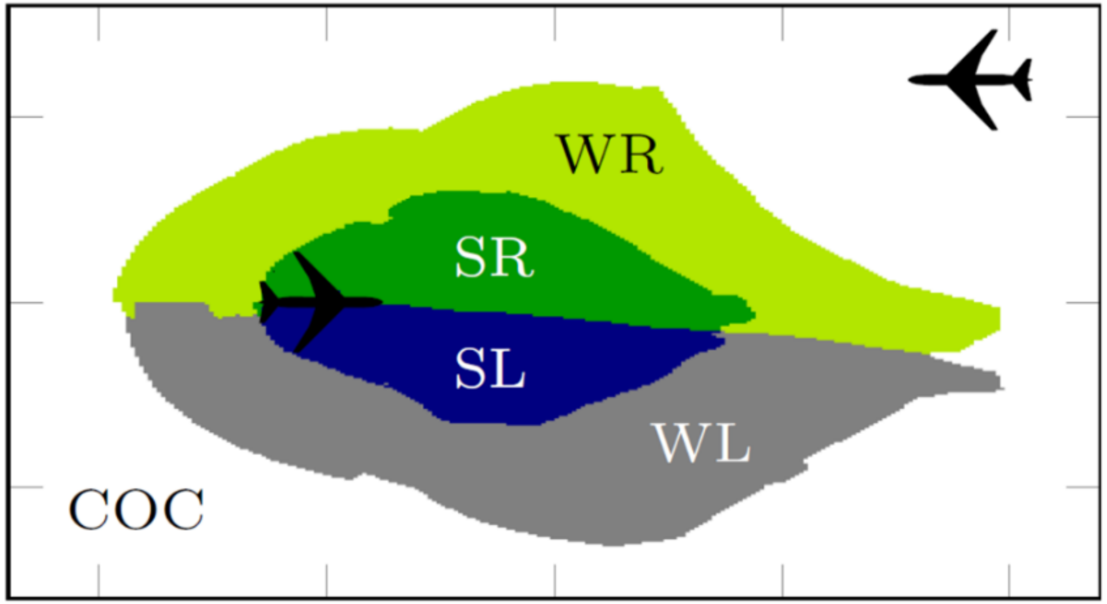
The ACAS Xu System

- An *Airborne Collision-Avoidance System*, for drones
- Being developed by the US Federal Aviation Administration (FAA)
- Produce an advisory:
  - *Clear-of-conflict (COC)*
  - *Strong left*
  - *Weak left*
  - *Strong right*
  - *Weak right*



The ACAS Xu System (cnt'd)

- Certification via testing and simulation
- *Encounter plots*



- But these only cover a finite set of inputs
  - Verification can help

The ACAS Xu System (cnt'd)

- ACAS Xu logic *too complex* for manual implementation
- Previous approach: large lookup table (size: 2GB)
  - Interpolate if needed
- Switched to neural networks for *compression* (size: 3MB)
  - Also smoother than interpolation
- But this requires a new *certification* procedure
  - Especially because this is a new approach

Neural Network Verification

Definition (The Neural Network Verification Problem)

For a neural network $N : \bar{x} \rightarrow \bar{y}$, an input property $P(\bar{x})$ and an output property $Q(\bar{y})$, does there exist an input $\bar{x}_0$ with output $\bar{y}_0 = N(\bar{x}_0)$, such that $\bar{x}_0$ satisfies P and $\bar{y}_0$ satisfies Q?

- $P(\bar{x})$ characterizes the inputs we are checking
- $Q(\bar{y})$ characterizes *undesired* behavior for those inputs
- Negative answer (`UNSAT`) means property *holds*
- Positive answer (`SAT`) includes a *counterexample*

Example: ACAS Xu

- Want to ensure: whenever intruder is distant, network always answers *clear-of-conflict*
- $P(\bar{x})$:
  - $\bar{x}[0] \geq 40000$
- $Q(\bar{y})$:
  - $(\bar{y}[0] \leq \bar{y}[1]) \vee (\bar{y}[0] \leq \bar{y}[2]) \vee (\bar{y}[0] \leq \bar{y}[3]) \vee (\bar{y}[0] \leq \bar{y}[4])$
- `UNSAT` means the system behaves as expected

Verification Complexity

Theorem (Neural Network Verification Complexity)

For a neural network with ReLU activation functions, and for properties $P()$ and $Q()$ that are conjunctions of linear constraints, the verification problem is NP-complete in the number of ReLU nodes

- Membership in NP: can check in polynomial time that a given x satisfies $P(x)$ and $Q(N(x))$

  Continuous variables => use LP after guessing phases (de/activatation) of ReLU

- NP-Hardness: by reduction from 3-SAT

Techniques and Challenges

- Main challenge is *scalability*
  - Usually the case in verification
- Two kinds of techniques:
  - *Sound* and *complete*:
    - limited scalability
    - always succeed
  - *Sound* and *incomplete*:
    - better scalability
    - can return "don't know"
- Orthogonal: *abstraction* techniques
- Related: testing techniques (e.g., *coverage criteria*, *concolic testing*). Not covered here

Provide exact bounds.

Ex: RELUPLEX

Provide certified upper/lower bound.
No refinement if not enough

Ex: AI^2, next part

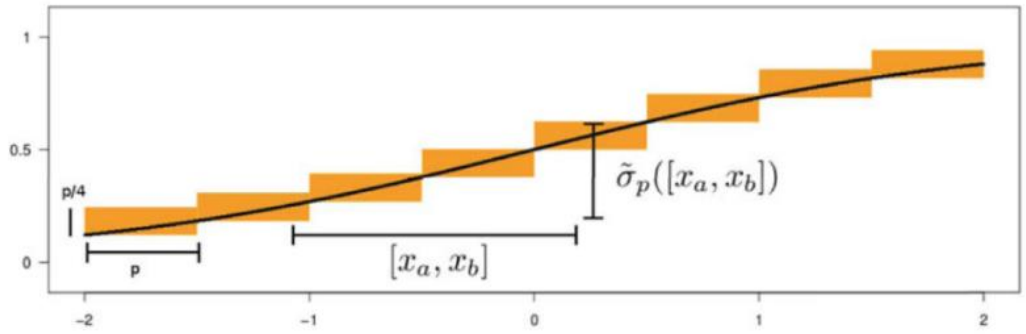
So, How Big a Network can you Verify?

- Very difficult to compare!
  - Different *properties* make a huge difference
  - Compare *complete* and *incomplete* techniques
  - Different underlying *engines*
  - Different *benchmarks*
    - Comparative study: Bunel et al, 2017 [BTT+17]
- Still, as a rule of thumb...
  - *Complete* techniques: hundreds to *thousands*
  - *Incomplete* techniques: thousands to *tens of thousands*

Incomplete techniques (abstractions…):

First: NEVER (Pulina et al. 2010).

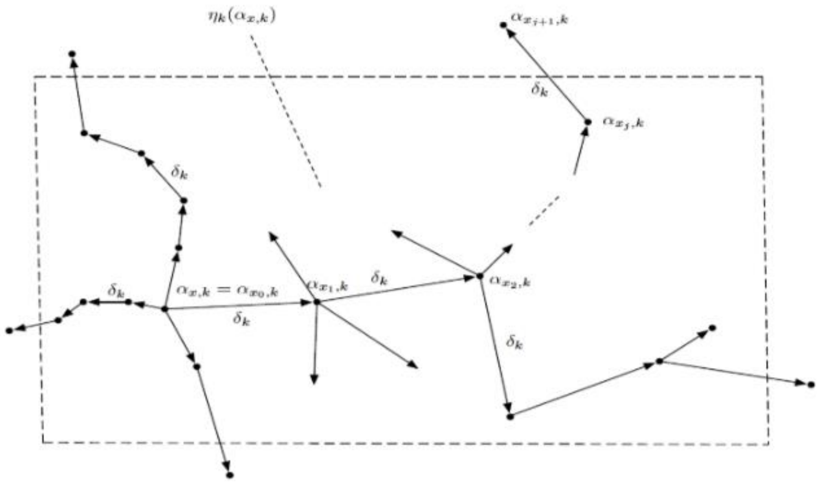
- Among first attempts to verify neural networks
- Focused on networks with Sigmoid activation functions
- Main idea: *over-approximate* Sigmoids using *interval arithmetic*
- … and then apply the interval arithmetic solver HySAT



Abstraction used (piecewise constant)
Can tackle only ~10 neurons
Later: AI²

DLV (Huang et al, 2017) [HKWW17]

- Apply a *discretization* of the input space
  - Discretization via *manipulations*
  - These can represent camera scratches, rotations, etc
  - *Sound* but *incomplete*



- Then do an *exhaustive* search, layer-by-layer
- Tool: the *DLV* solver, evaluated on image recognition networks

Complete techniques:

First: Bastani et al. 2016.

Use LP solvers (linear programming, PTIME).

Problem: ReLU is not linear => it is a OR of 2 linear function.

Heuristic to fix the phase of each ReLU.

$\Rightarrow$ Sound but incomplete techniques.

To make it complete:

Search exhaustively every possible choice for RELU.
Set a choice. Backtrack if no counterexample found.

Heuristic to search in a good direction, like SAT solvers.
Many varations in 2017:
Planet Solver (Elhers), Tjeng and Tedrake, Katz et al, BAB Solver (BTT), Lomuscio and Magnenti….
Sherlock Solver (Dutta et al. 2018).

Or use quadratic Solvers Cheng et al. 2017.

Later: RELUPLEX

Additional Techniques at a Glance

- Networks as continuous functions, *Lipschitz continuity*
  - Ruan et al [RHK18], Hull et al [HWZ02], Hein and Andriushchenko [HA17], Weng at al [WZC+18]
- Verification of *Binarized* Neural Networks
  - Cheng et al [CNR17b], Narodytska et al [NKR+18]

Reluplex

- Joint work with Clark Barrett, David Dill, Kyle Julian and Mykel Kochenderfer (CAV 2017 [KBD+17a]), supported by the FAA and Intel

- A *sound* and *complete* verification procedure
- Applied to the ACAS Xu case study
  - Networks an order of magnitude larger than previously possible
- Project still ongoing (*Marabou* [KHI+19])

Reluplex (cnt'd)

- SMT-solver for quantifier-free linear real arithmetic + ReLUs
- Based on the *Simplex* method for linear programming
  - Simplex + ReLUs = Reluplex
  - Applicable to other piece-wise linear functions
- Key SMT idea: handle ReLUs *lazily*
  - As opposed to eager case splitting
  - *Defer* splitting for as long as possible
  - May not have to split at all!
- But first, an introduction to Simplex

Simplex

Aim: find optimal solution satisfying some constraints.
First phase: Find Feasible solution

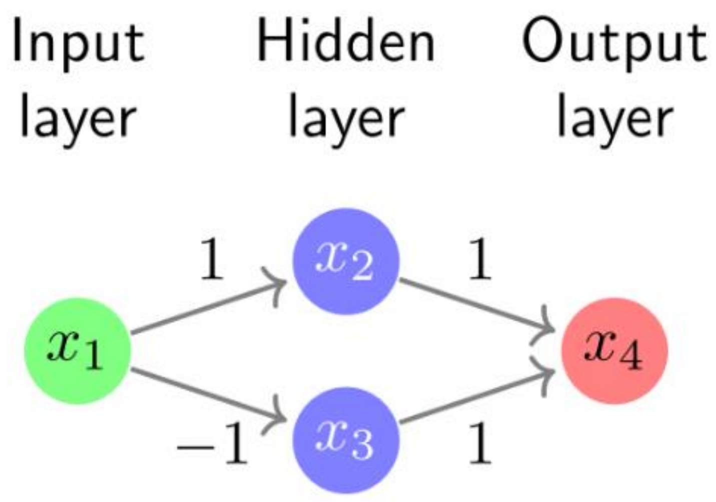
- Iterative algorithm
- Always maintain a *variable assignment*
- Assignment always *satisfies equations*
  - But may *violate bounds*
- In every iteration, attempt to reduce the overall *infeasibility*

Secund phase: Optimize

Simplex: Basics and Non-Basics

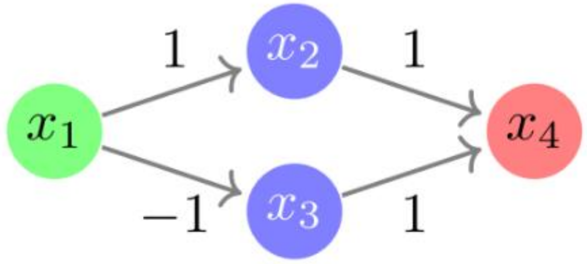
- Variables partitioned into *basic* and *non-basic* variables
  - Non-basics are "free"
  - Basics are "bounded"
- Non-basic assignment dictates basic assignment
  - This is how the equations are maintained
- In every iteration, we can perform
  1. an *update*: change the assignment of a non-basic variable
     - and any affected basics
  2. a *pivot*: switch a basic and a non-basic variable

Simplex: Example



- No activation functions
- Property being checked: for $x_1 \in [0, 1]$, always $x_4 \notin [0.5, 1]$ True
  - Negated output property: $x_1 \in [0, 1]$ and $x_4 \in [0.5, 1]$ False

Simplex: Example (cnt'd)



- Equations for weighted sums:

$$x_2 - x_1 = x_5$$
$$x_3 + x_1 = x_6$$
$$x_4 - x_3 - x_2 = x_7$$

- Bounds:

$$x_1 \in [0, 1]$$
$$x_4 \in [0.5, 1]$$
x_2, x_3 unbounded
$$x_5, x_6, x_7 \in [0, 0]$$

Hypothesis to check

- Technicality: replace constants by *auxiliary* variables

Simplex: Example (cnt'd)

$$x_5 = x_2 - x_1$$

$$x_6 = x_3 + x_1$$

$$x_7 = x_4 - x_3 - x_2$$

Lower B.	Var	Value	Upper B.	
0	x_1	0	1	Hypothesis to check
	x_2	0		
	x_3	0		
0.5	x_4	0.5	1	Hypothesis to check
0	x_5	0	0	
0	x_6	0	0	
0	x_7	0.5	0	

Update:

x4 := x4 + 0.5

Now, need to change x7.

But x7 on the left (Var. are either right or left)

Can change only variable on right, so need to:

pivot x7,x2

Simplex: Example (cnt'd)

$$x_5 = x_2 - x_1 \qquad \leftarrow \qquad x_5 = x_4 - x_3 - x_7 - x_1$$

$$x_6 = x_3 + x_1$$

$$x_7 = x_4 - x_3 - x_2 \qquad \leftarrow \qquad x_2 = x_4 - x_3 - x_7$$

Lower B.	Var	Value	Upper B.
0	x_1	0	1
	x_2	0.5	
	x_3	0	
0.5	x_4	0.5	1
0	x_5	0.5	0
0	x_6	0	0
0	x_7	0	0

x5 is incorrect.

Pivot: x_7, x_2

Update:
x7 := x7 - 0.5

Simplex: Example (cnt'd)

$x_5 = x_4 - x_3 - x_7 - x_1 \quad \leftarrow \quad x_1 = x_4 - x_3 - x_7 - x_5$

$x_6 = x_3 + x_1 \quad \leftarrow \quad x_6 = x_4 - x_7 - x_5$

$x_2 = x_4 - x_3 - x_7$

x4 ➔ x7 ➔ x5 ➔ x6

➔ need to change x5 (through x1 or x3)

➔ Failure.

no $x_1 \in [0,1]$ with $x_4 \in [0.5,1]$

Pivot: x_5, x_1

Update:
x5 := x5 - 0.5

Lower B.	Var	Value	Upper B.
0	x_1	0.5	1
	x_2	0.5	
	x_3	0	
0.5	x_4	0.5	1
0	x_5	0	0
0	x_6	0.5	0
0	x_7	0	0

x6 is incorrect.

Properties of Simplex

Theorem (Soundness and Completeness of Simplex)

*The simplex algorithm is sound and complete**

- Soundness:
  - `SAT` $\Rightarrow$ assignment is correct
  - `UNSAT` $\Rightarrow$ no assignment exists
- Completeness: depends on variable selection strategy
- *Bland's rule*: guarantees termination
  - Always pick variables with smallest index
  - Prevents cycling
  - But unfortunately quite slow
- Better selection strategies exist (e.g., *steepest edge*)
- Problem is in $\mathbf{P}$, unknown whether simplex is in $\mathbf{P}$

Simplex to simple RELUPLEX

Fix every ReLU activation phase first: activated (input >0) or deactivated (input <0).

We have linear constraints!

We can use Simplex to solve it

If couterexample found => return SAT

Else: fix another activation of ReLU and loop till all activation have been tested.

Return UNSAT

Properties of Reluplex

Theorem (Soundness and Completeness of Reluplex)

*The Reluplex algorithm is sound and complete**

- Soundness:
  - `SAT` $\Rightarrow$ assignment is correct
  - `UNSAT` $\Rightarrow$ no assignment exists
- Completeness: depends on *variable selection strategy* and *splitting strategy*
- Naive approach: split on all variables immediately, apply Bland's rule
  - This is the case-splitting approach from before
  - Ensures termination

More Efficient Reluplex (Lazy)

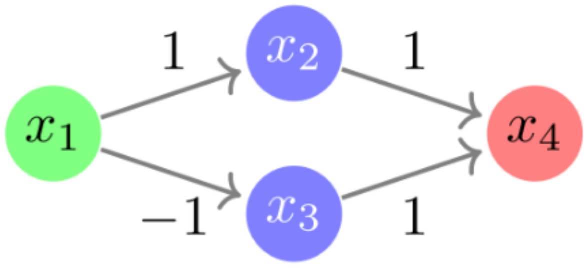
- Better approach: *lazy splitting*
  - Start fixing bound violations
  - Once all variables within bounds, address broken ReLUs
    - If a ReLU is repeatedly broken, split on it
    - Otherwise, fix it without splitting
  - And repeat as needed
- Usually end up splitting on a fraction of the ReLUs (20%)
- Can reduce splitting further with some additional work

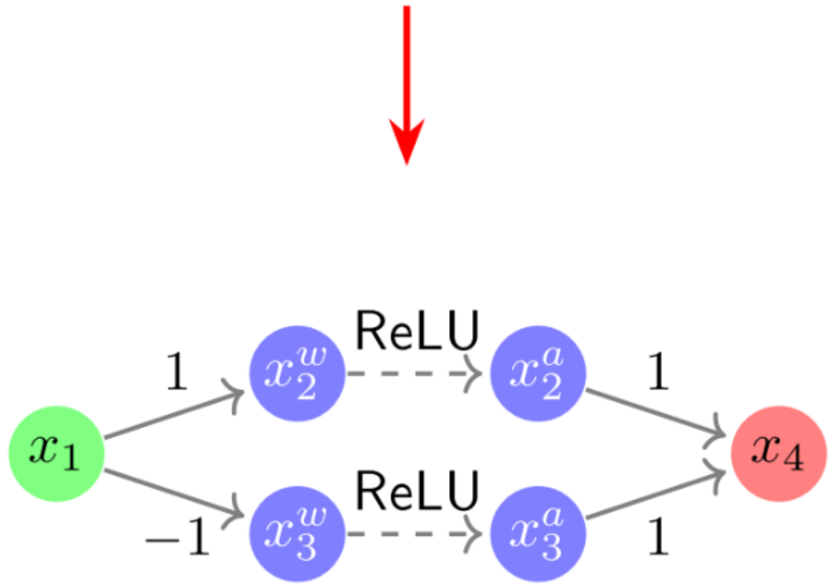
From Simplex to Reluplex (Lazy)

- Each ReLU node x represented as two variables:
  - x^w to represent the (input) *weighted sum*
  - x^a to represent the (output) *activation result*

Decoupled variables
x^w and x^a have no
relation at first
=> Only linear op.
Can run simplex

Reluplex: Example



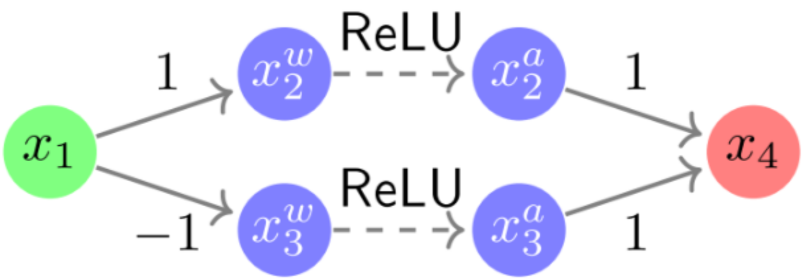


Can I find

$x_1 \in [0,1]$ with $x_4 \in [0.5,1]$?

Yes. $x_1=0.5 \Rightarrow x_4=0.5$

Reluplex: Example (cnt'd)



plus the ReLU properties:
$x_i^a = x_i^w$ if $x_i^w \geq 0$ and $x_i^a = 0$ otherwise
to solve after the rest is solved (lazy)

- Equations for weighted sums:

$$x_5 = x_2^w - x_1$$
$$x_6 = x_3^w + x_1$$
$$x_7 = x_4 - x_3^a - x_2^a$$

- Bounds:

$$x_1 \in [0, 1]$$
$$x_4 \in [0.5, 1]$$
$$x_2^w, x_3^w \text{ unbounded}$$
$$x_2^a, x_3^a \in [0, \infty)$$
$$x_5, x_6, x_7 \in [0, 0]$$

Linear Constraints
=> usual
Simplex algorithm

Reluplex: Example (cnt'd)

$$x_5 = x_2^w - x_1$$
$$x_6 = x_3^w + x_1$$
$$x_2^a = x_4 - x_3^a - x_7$$

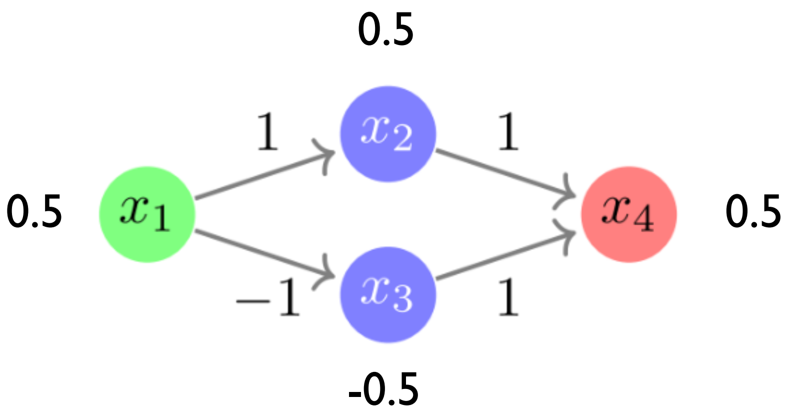
Lower B.	Var	Value	Upper B.
0	x_1	0.5	1
	x_2^w	0.5	
0	x_2^a	0.5	
	x_3^w	-0.5	
0	x_3^a	0	
0.5	x_4	0.5	1
0	x_5	0	0
0	x_6	0.5	0
0	x_7	0	0

Normal simplex algorithm finds a solution

But not true with additional $x_2^a = x_2^w$ if $x_2^w \geq 0$

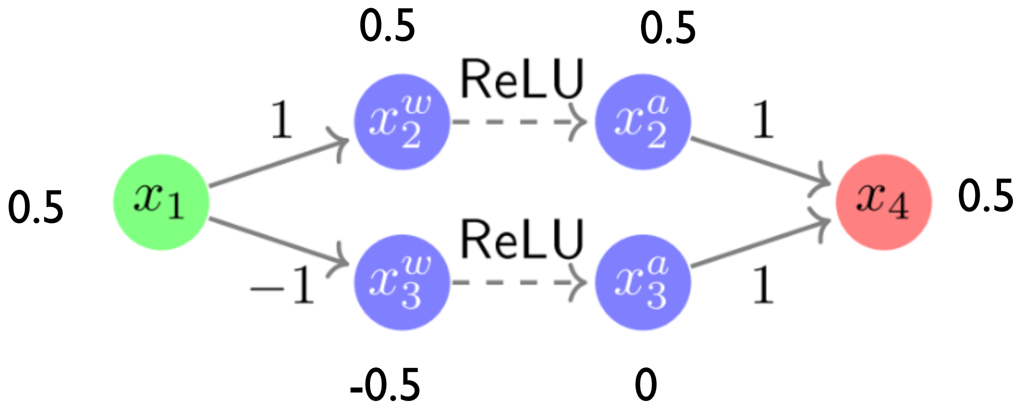
SOLUTION found: $x_1=0.5$ => $x_4=0.5$

Reluplex: Example



SOLUTION found

$x_1=0.5$ => $x_4=0.5$



More Efficient Reluplex: Bound Tightening

- During execution we encounter many equations
- Can use them for *bound tightening*
- Example:

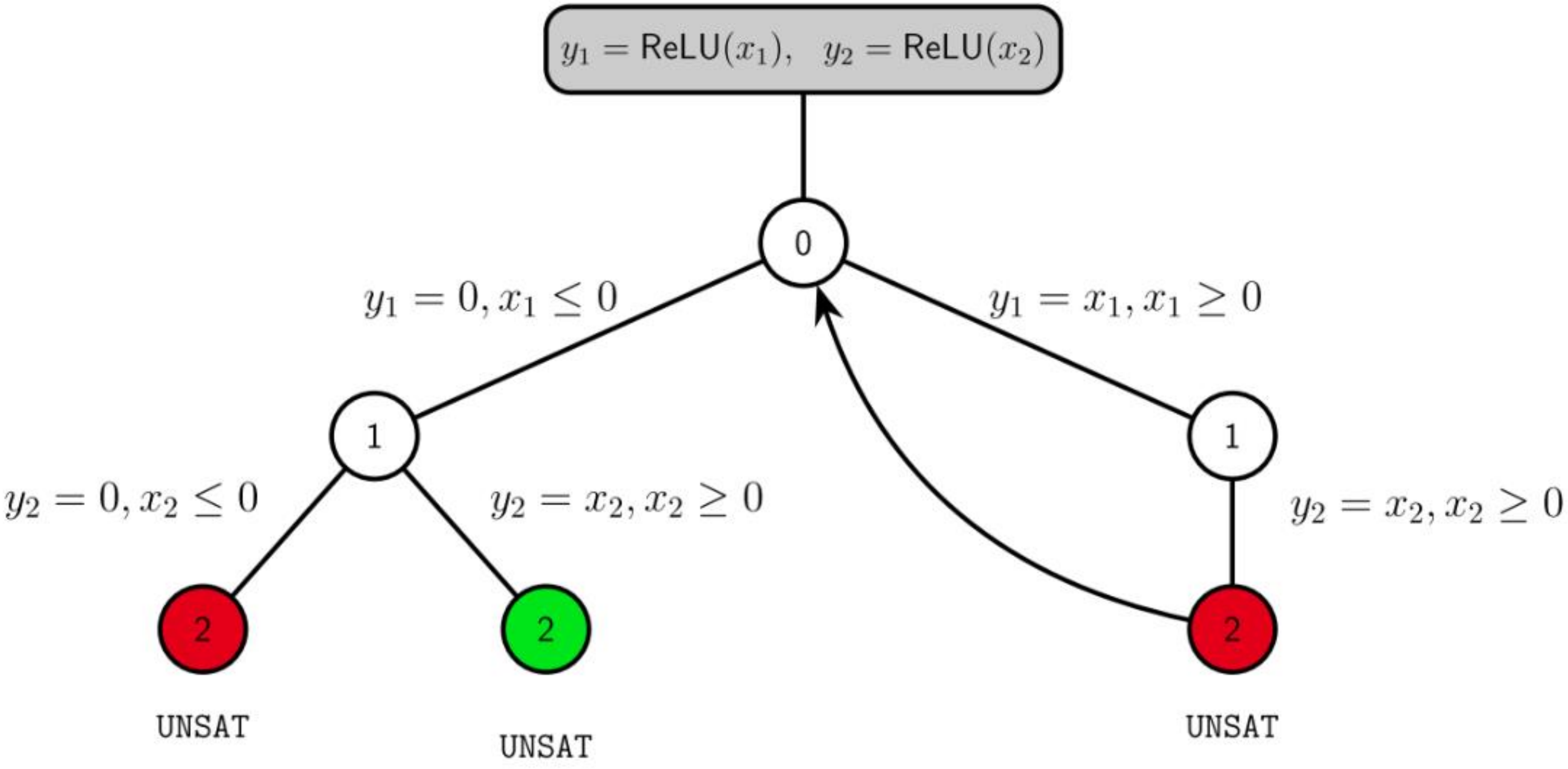
$$x = y + z \qquad\qquad x \geq -2, \quad y \geq 1, \quad z \geq 1$$

- Can derive a *tighter* bound: $x \geq 2$
- If x is part of a ReLU pair, we say that ReLU's phase is *fixed*
  - And we replace it by a linear equation
  - Same as in case splitting, only no back-tracking required

Non-Chronological Backtracking (Backjumping)

- A useful technique in SAT and SMT solving
- Backtracking: change *last* guess
- Backjumping: change an *earlier* guess
- Need to keep track of the discovery of new bounds

Non-Chronological Backtracking (Backjumping) (cnt'd)



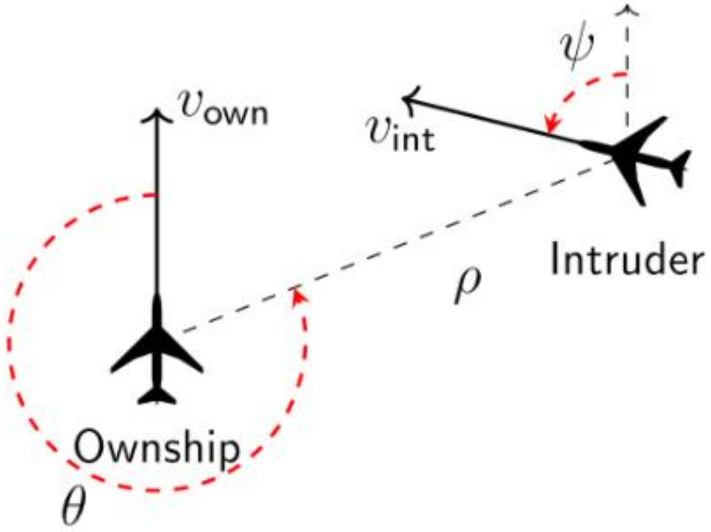
Enhancements (Marabou [KHI$^+$19])

- Engineering improvements: multiple input formats
  - E.g., TensorFlow
- Parallelism: divide and conquer
- Network level reasoning
- New simplex solver

RELUPLEX for:

The ACAS Xu System

- An *Airborne Collision-Avoidance System*, for drones
- Being developed by the US Federal Aviation Administration (FAA)
- Produce an advisory:
  - *Clear-of-conflict (COC)*
  - *Strong left*
  - *Weak left*
  - *Strong right*
  - *Weak right*



RELUPLEX for

Certifying ACAS Xu (cnt'd)

- We worked on a list of 10 properties
- Example 1:
  - If the intruder is near and approaching from the left, the network advises strong right
    - Distance: $12000 \leq \rho \leq 62000$
    - Angle to intruder: $0.2 \leq \theta \leq 0.4$
    - Etc.
  - Proved in under 1.5 hours

RELUPLEX for

Certifying ACAS Xu (cnt'd)

- Example 2:
  - If vertical separation is large and the previous advisory is weak left, the network advises clear-of-conflict or weak left
    - Distance: $0 \leq \rho \leq 60760$
    - Time to loss of vertical separation: $\tau = 100$
    - Etc.
  - Found a counter-example in 11 hours

Certifying ACAS Xu (cnt'd)

	Networks	Result	Time	Stack	Splits
ϕ_1	41	UNSAT	394517	47	1522384
	4	TIMEOUT			
ϕ_2	1	UNSAT	463	55	88388
	35	SAT	82419	44	284515
ϕ_3	42	UNSAT	28156	22	52080
ϕ_4	42	UNSAT	12475	21	23940
ϕ_5	1	UNSAT	19355	46	58914
ϕ_6	1	UNSAT	180288	50	548496
ϕ_7	1	TIMEOUT			
ϕ_8	1	SAT	40102	69	116697
ϕ_9	1	UNSAT	99634	48	227002
ϕ_{10}	1	UNSAT	19944	49	88520

Adversarial Robustness

Goodfellow et al., 2015

- Slight perturbations of inputs lead to misclassification
- Verification can prove that this cannot occur
- Allows us to assess attacks and defenses

Local Adversarial Robustness

- Verification query: for a given panda $\bar{x}_0$ and a given amount of noise δ, does classification remain the same?
  - If $\|\bar{x} - \bar{x}_0\|_L \leq \delta$ then $\bigwedge_i (\bar{y}[i_0] \geq \bar{y}[i])$, where $\bar{y}[i_0]$ is the desired label
- Easiest norm to handle: L_∞, the infinity norm
  - $\|\bar{x} - \bar{x}_0\|_{L_\infty} \leq \delta \quad \Leftrightarrow \quad \forall i. -\delta \leq \bar{x}[i] - \bar{x}_0[i] \leq \delta$
- Can also handle L_1

Local Adversarial Robustness (cnt'd)

- Can find the *optimal* δ for which robustness holds
  - Using binary search
- Example: an ACAS Xu network

	$\delta = 0.1$		$\delta = 0.075$		$\delta = 0.05$		$\delta = 0.025$		$\delta = 0.01$	
	Result	Time	Result	Time	Result	Time	Result	Time	Result	Time
Point 1	SAT	135	SAT	239	SAT	24	UNSAT	609	UNSAT	57
Point 2	UNSAT	5880	UNSAT	1167	UNSAT	285	UNSAT	57	UNSAT	5
Point 3	UNSAT	863	UNSAT	436	UNSAT	99	UNSAT	53	UNSAT	1
Point 4	SAT	2	SAT	977	SAT	1168	UNSAT	656	UNSAT	7
Point 5	UNSAT	14560	UNSAT	4344	UNSAT	1331	UNSAT	221	UNSAT	6

Assessing Attacks and Defenses [CKBD18]

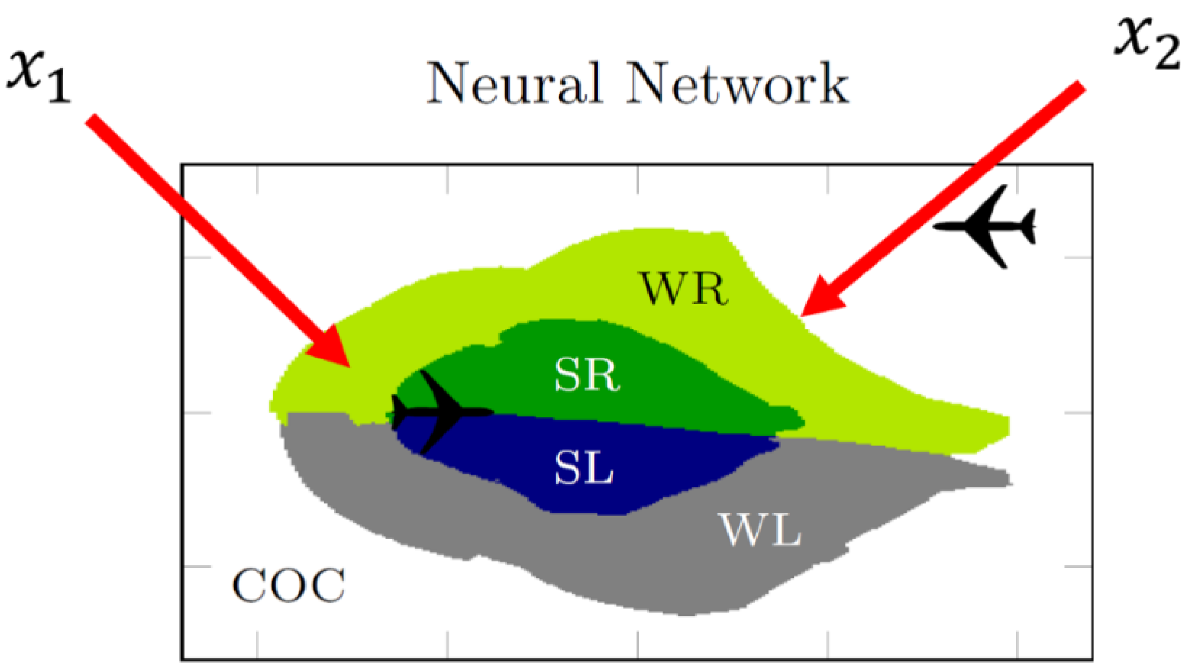
- Assessing attacks:
  - Pick point $\bar{x}$
  - Use *verification* to find optimal δ
  - Use *attack* to find δ'
  - See how close δ' is to δ
- Example: Carlini-Wagner attack [CW17] on a small MNIST network
- On average, δ about 6% smaller than δ'

Assessing Attacks and Defenses [CKBD18] (cnt'd)

- Assessing defenses:
  - Start with network N
  - Train *hardened* network $\bar{N}$
  - Pick point $\bar{x}$
  - Compare optimal δ *before* and *after* hardening
- Example: Madry defense [MMS+18] on a small MNIST network
- On average, hardened δ about 423% larger
- However, smaller in some cases

Global Robustness?

- Previous definition: for a particular input $\bar{x}_0$
  - What's an acceptable δ?
  - How do you pick $\bar{x}_0$?



Global Robustness Queries

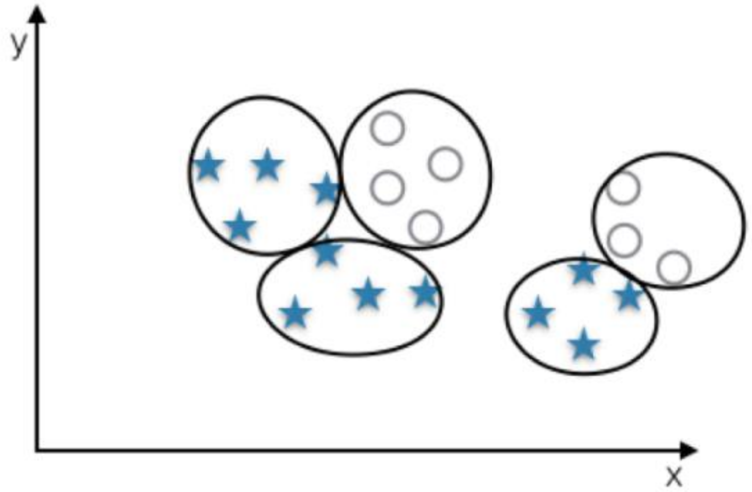
- Region boundaries: look at *confidence* instead of label
- Let p_1, p_2 be confidence levels for certain label:

$$\forall \bar{x}_1, \bar{x}_2. \quad \|\bar{x}_1 - \bar{x}_2\| \leq \delta \Rightarrow |p_1 - p_2| \leq \epsilon$$

  - Small changes to input do not change output by much
- *Significantly* slower to compute
  - Double the network size
  - Large input regions
- And also still need to choose δ, ϵ
- A compromise: a *clustering* based approach

DeepSafe: A Clustering-Based Approach [GKPB18]

- Use *clustering* to identify regions on which the network should be consistent
  - Clustering applied to known points (e.g., training set)
  - Identify centroid $\bar{x}_0$ and radius δ for each cluster



- Higher degree of automation
- Discovered an adversarial example in ACAS Xu

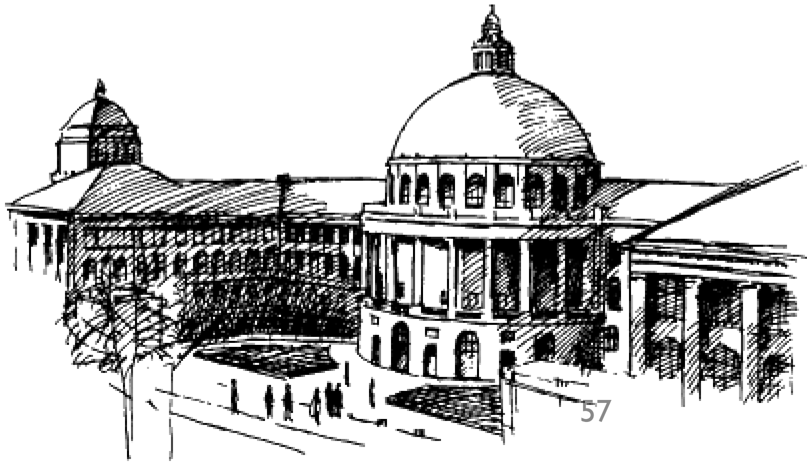
Safe and Robust Deep Learning: Using Abstraction (AI² to ERAN)

Gagandeep Singh

PhD Student

Department of Computer Science

ETH Zürich



SafeAI @ ETH Zurich

Joint work with

Martin Vechev

Markus Püschel

Timon Gehr

Matthew Mirman

Mislav Balunovic

Maximilian Baader

Petar Tsankov

Dana Drachsler

Publications:

[1] AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation, S&P'18

[2] Differentiable Abstract Interpretation for Provably Robust Neural Networks, ICML'18

[3] Fast and Effective Robustness Certification, NeurIPS'19

[4] An Abstract Domain for Certifying Neural Networks, POPL'19

[5] Boosting Robustness Certification of Neural Networks, ICLR'19

safeai.ethz.ch

Abstrations for Adversarial Robustness

- Exact solvers often do not scale to large networks => Use abstraction.
- Not always complete, but can prove both SAT (problem) and UNSAT (safe)

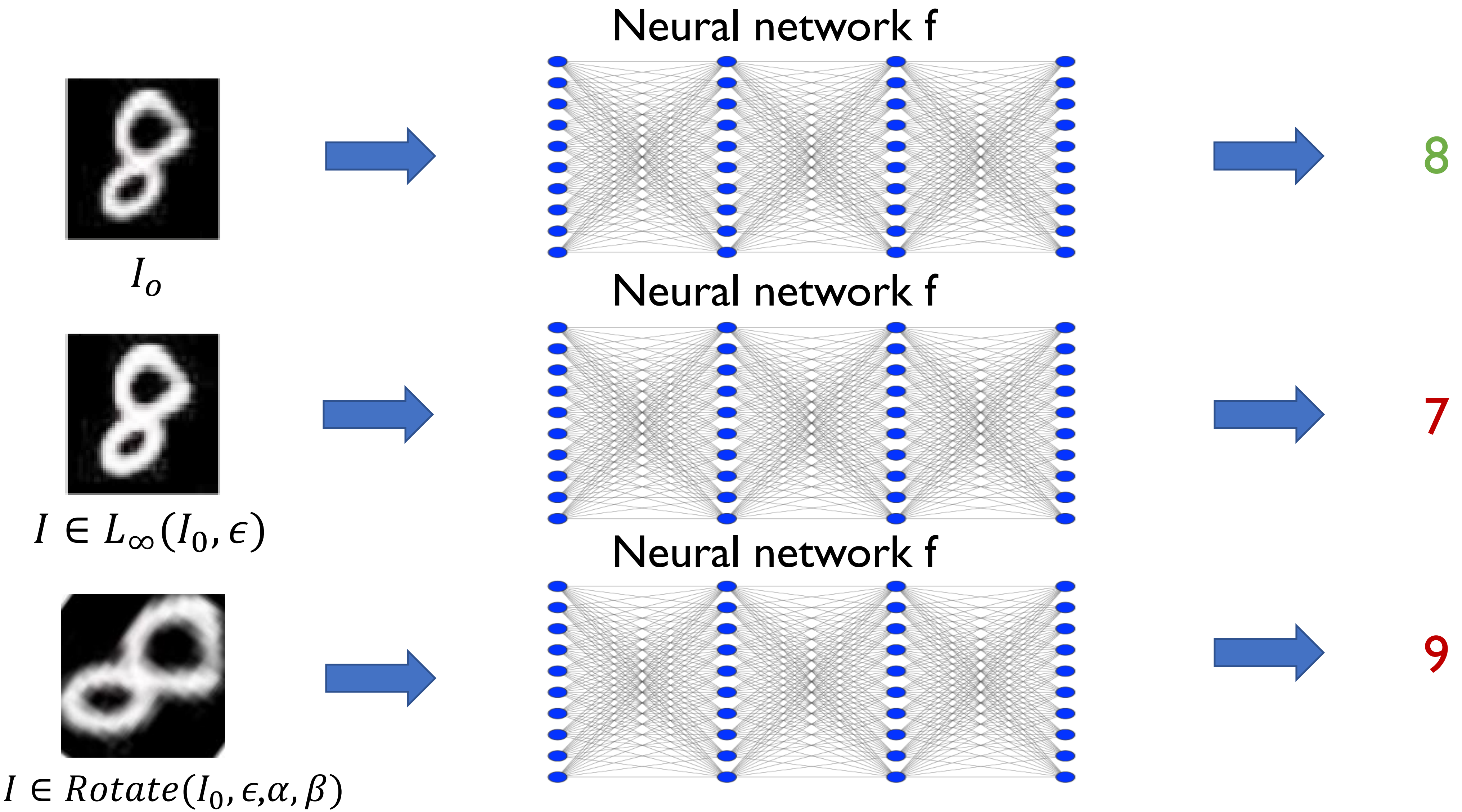
Experimental robustness

- **generate** adversarial examples
- **under-**approximation of network behavior in the adversarial region
- Madry et al. 2017

Certified robustness

- **prove** absence of adversarial examples
- **over-**approximation of network behavior in the adversarial region
- Gehr et al. 2018

Adversarial regions



Adversarial region $L_\infty(I_0, \epsilon)$

All images I where the intensity at each pixel differs from the intensity at the corresponding pixel in I_0 by $\leq \epsilon$

I_0 | $I_0 + 0.1$ | $I_0 + 0.2$ | $I_0 + 0.3$ | $I_0 + 0.4$ | $I_0 + 0.5$ | $I_0 + 0.6$ | $I_0 + 0.7$ | $I_0 + 0.8$

Adversarial region $Rotate(I_0, \epsilon, \alpha, \beta)$

All images I which are obtained by rotation each image in $L_\infty(I_0, \epsilon)$ by an angle between α and β using bilinear interpolation

Adversarial region

Intensity changes

Geometric transformation

Noise+Audio preprocessing

Aircraft sensors → Possible sensor values

Neural Network

Fully connected	ReLU
Convolutional	Sigmoid
Residual	Tanh
LSTM	Maxpool

Safety Property

ERAN analyzer

https://github.com/eth-sri/eran

- Box
- DeepZ
- DeepPoly
- RefineZono
- K-Poly

Based on ELINA
https://github.com/eth-sri/ELINA

Tensorflow graph as input

Sound with respect to floating point arithmetic

Both complete and incomplete verification

State-of-the-art precision and performance

Used by SBB CFF FFS

Yes

No

Results with ERAN

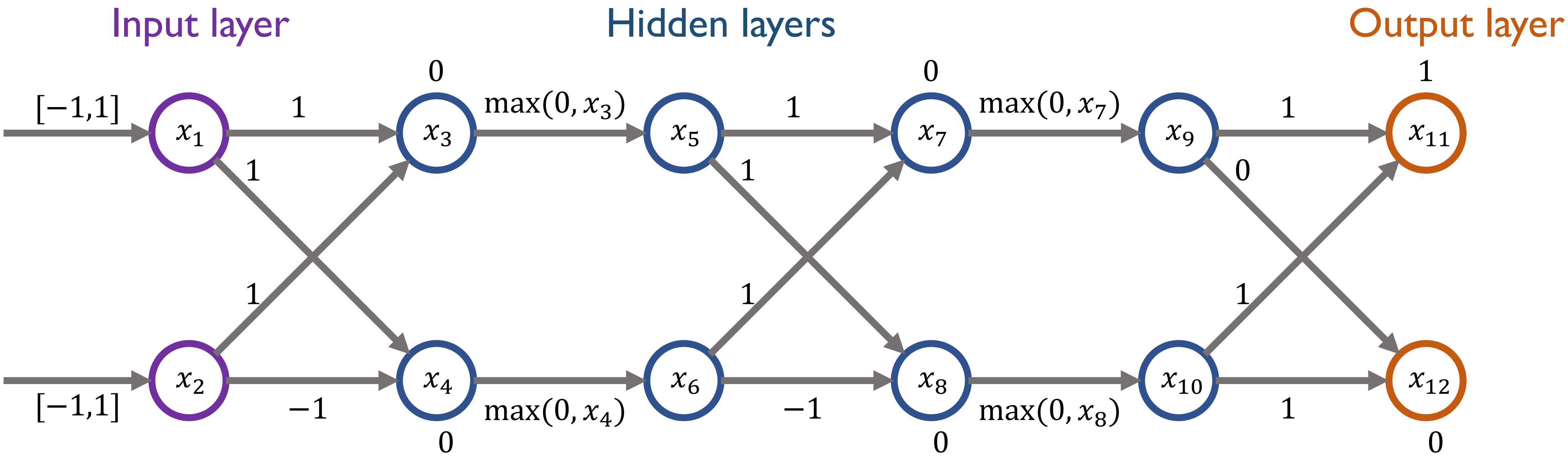
Aircraft collision avoidance system		
Reluplex	**Neurify**	**ERAN**
> 32 hours	921 sec	227 sec

MNIST CNN with > 88K neurons		
ϵ	**%verified**	**Time (s)**
0.1	97%	133 sec

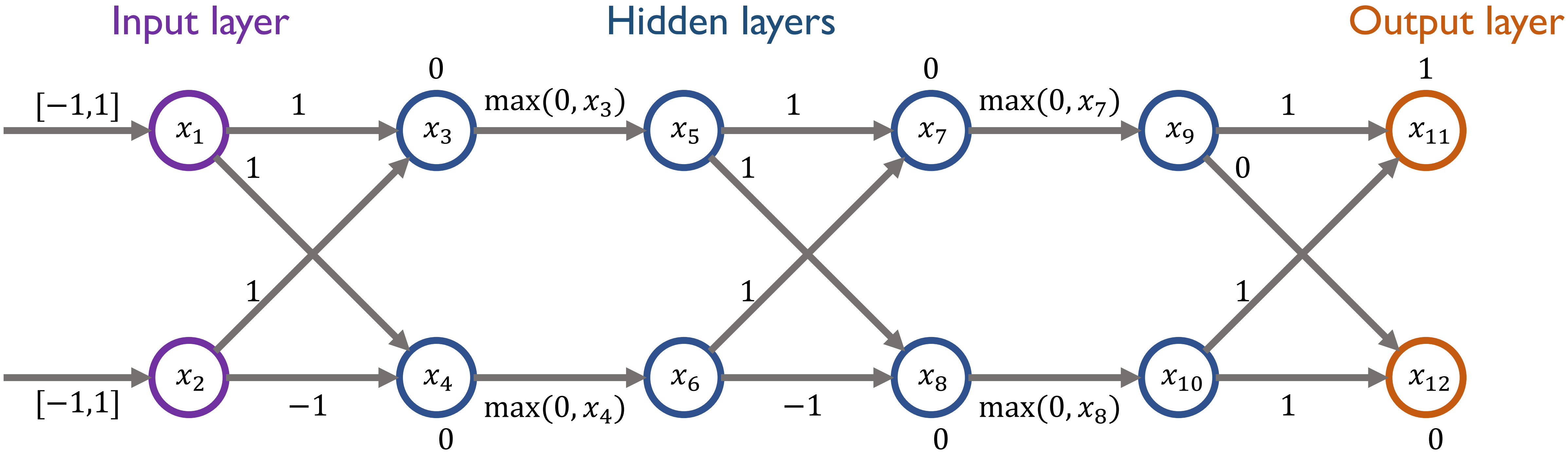
Rotation between -30° and 30° on MNIST CNN with 4,804 neurons		
ϵ	**%verified**	**Time(s)**
0.001	86	10 sec

LSTM with 64 hidden neurons		
ϵ	**%verified**	**Time (s)**
-110 dB	90%	9 sec

Example: Analysis of a Toy Neural Network



We want to prove that $x_{11} > x_{12}$ for all values of x_1, x_2 in the input set



$$min\ x_{11} - x_{12}$$

$$s.t.:\ x_{11} = x_9 + x_{10} + 1,\ x_{12} = x_{10},$$
$$x_9 = \mathbf{max}(0, x_7),\ x_{10} = \mathbf{max}(0, x_8),$$
$$x_7 = x_5 + x_6,\ x_8 = x_5 - x_6,$$
$$x_5 = \mathbf{max}(0, x_3),\ x_6 = \mathbf{max}(0, x_4),$$
$$x_3 = x_1 + x_2,\ x_4 = x_1 - x_2,$$
$$-1 \leq x_1 \leq 1,\ -1 \leq x_2 \leq 1.$$

Each $x_j = \mathbf{max}(0, x_i)$ corresponds to
$(x_i \leq 0$ and $x_j = 0)$ or
$(x_i > 0$ and $x_j = x_i)$

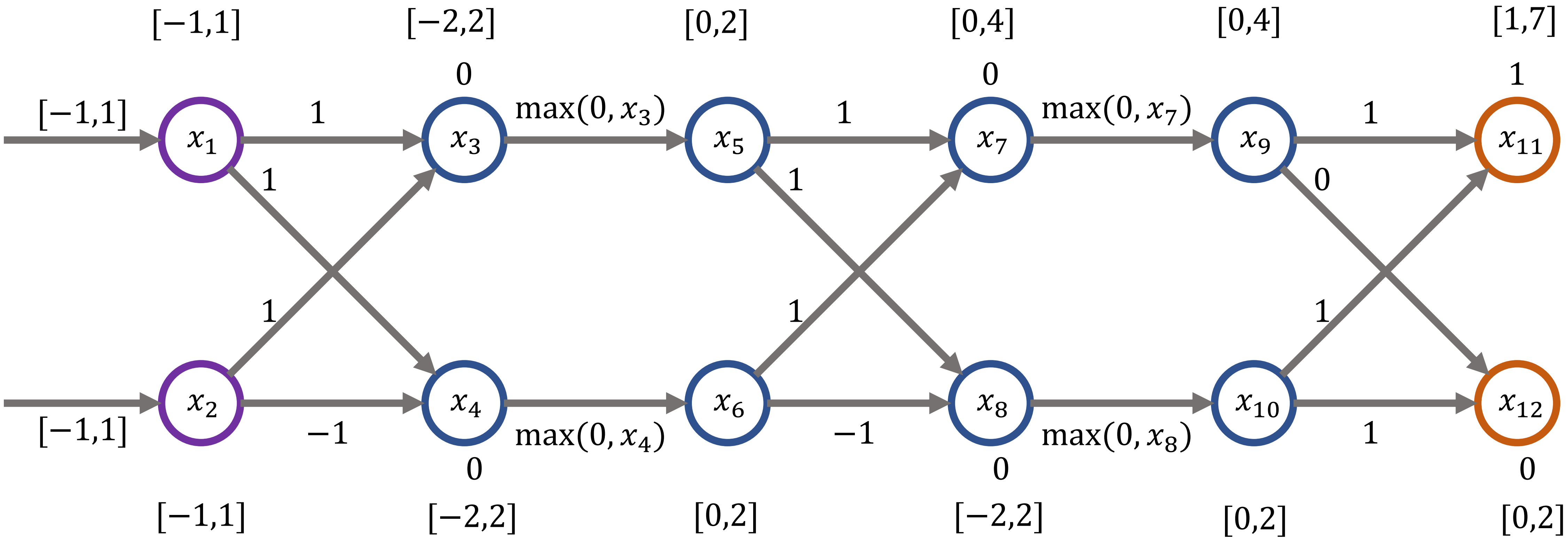
Solver has to explore two paths per ReLU resulting in exponential number of paths

Complete verification with solvers often does not scale

Analysis Trade-offs: Precision vs. Scalability

Publication	Description	
AI^2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation, Security & Privacy, 2018 (Gehr, Mirman, Drachsler-Cohen, Tsankov, Chaudhuri, Vechev)	AI2: Generic conceptual framework for analyzing neural networks with AI.	
Fast and Effective Robustness Certification NeurIPS 2018 (with Gehr, Mirman, Vechev, Püschel)	DeepZ: Zonotope domain with new custom abstract transformers tailored to neural networks	More scalable Less precise
An Abstract Domain for Certifying Neural Networks POPL 2019 (with Gehr, Vechev, Püschel)	DeepPoly: New, restricted polyhedra domain with abstract transformers specifically tailored to neural networks	More scalable Less precise
Boosting Robustness Certification of Neural Networks ICLR 2019 (with Gehr, Vechev, Püschel)	RefineZono: Best of both: AI + solvers. More scalable than pure MILP solutions and more precise than pure AI (but less scalable)	More precise Less scalable

Box Abstract Domain



Verification with the Box domain fails as it cannot capture relational information

DeepPoly Abstract Domain [POPL'19]

Shape: associate a lower polyhedral $a_i^{\leq}$ and an upper polyhedral $a_i^{\geq}$ constraint with each x_i

$$a_i^{\leq}, a_i^{\geq} \in \{x \mapsto v + \sum_{j \in [i-1]} w_j \cdot x_j \mid v \in \mathbb{R} \cup \{-\infty, +\infty\}, w \in \mathbb{R}^{i-1}\} \text{ for } i \in [n]$$

Concretization of abstract element a:

$$\gamma_n(a) = \{x \in \mathbb{R}^n \mid \forall i \in [n].\, a_i^{\leq}(x) \leq x_i \wedge a_i^{\geq}(x) \geq x_i\}$$

Domain invariant: store auxiliary concrete lower and upper bounds l_i, u_i for each x_i

$$\gamma_n(a) \subseteq \times_{i \in [n]} [l_i, u_i]$$

- less precise than Polyhedra, restriction needed to ensure scalability
- captures affine transformation precisely unlike Octagon, TVPI
- custom transformers for ReLU, sigmoid, tanh, and maxpool activations

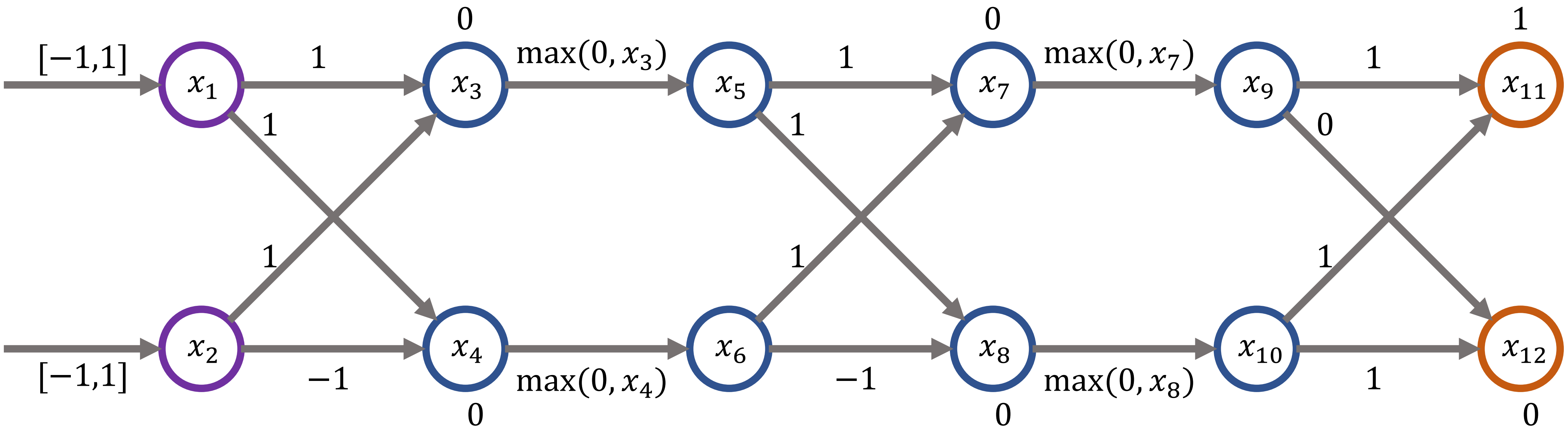
n: #neurons, m: #constraints

w_{max}: max #neurons in a layer, L: # layers

Transformer	Polyhedra	Our domain
Affine	$O(nm^2)$	$O(w_{max}^2 L)$
ReLU	$O(\exp(n, m))$	$O(1)$

$$\langle x_1 \geq -1, \quad x_1 \leq 1, \quad l_1 = -1, \quad u_1 = 1 \rangle$$

$$\langle x_3 \geq x_1 + x_2, \quad x_3 \leq x_1 + x_2, \quad l_3 = -2, \quad u_3 = 2 \rangle$$



$$\langle x_2 \geq -1, \quad x_2 \leq 1, \quad l_2 = -1, \quad u_2 = 1 \rangle$$

$$\langle x_4 \geq x_1 - x_2, \quad x_4 \leq x_1 - x_2, \quad l_4 = -2, \quad u_4 = 2 \rangle$$

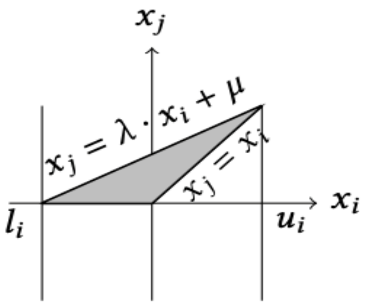
ReLU activation

Pointwise transformer for $x_j := max(0, x_i)$ that uses l_i, u_i

$if\ u_i \le 0, a_j^{\le} = a_j^{\ge} = 0, l_j = u_j = 0,$

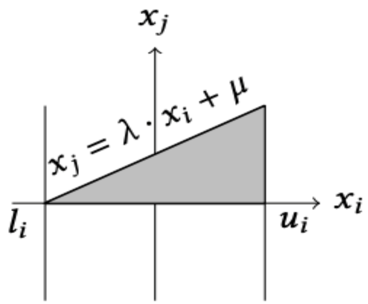
$if\ l_i \ge 0, a_j^{\le} = a_j^{\ge} = x_i, l_j = l_i, u_j = u_i,$

$if\ l_i < 0\ and\ u_i > 0$



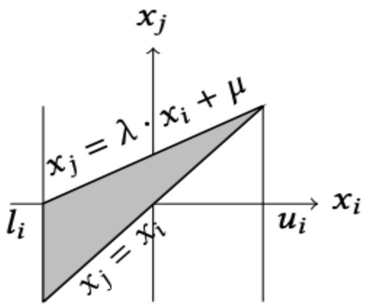
(a)

$x_i \le x_j, 0 \le x_j,$
$x_j \le u_i(x_i - l_i)/(u_i - l_i).$
$l_j = 0, u_j = u_i$



(b)

$0 \le x_j,$
$x_j \le u_i(x_i - l_i)/(u_i - l_i),$
$l_j = 0,\ u_j = u_i$



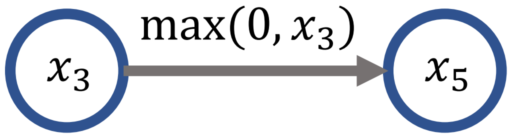
(c)

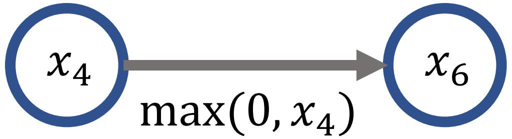
$x_i \le x_j,$
$x_j \le u_i(x_i - l_i)/(u_i - l_i),$
$l_j = l_i,\ u_j = u_i$

choose (b) or (c) depending on the area. Here use (b)

$\langle x_3 \ge x_1 + x_2,$
$x_3 \le x_1 + x_2,$
$l_3 = -2,$
$u_3 = 2\rangle$

$\langle x_5 \ge 0,$
$x_5 \le 0.5 \cdot x_3 + 1,$
$l_5 = 0,$
$u_5 = 2\rangle$





$\langle x_4 \ge x_1 - x_2,$
$x_4 \le x_1 - x_2,$
$l_4 = -2,$
$u_4 = 2\rangle$

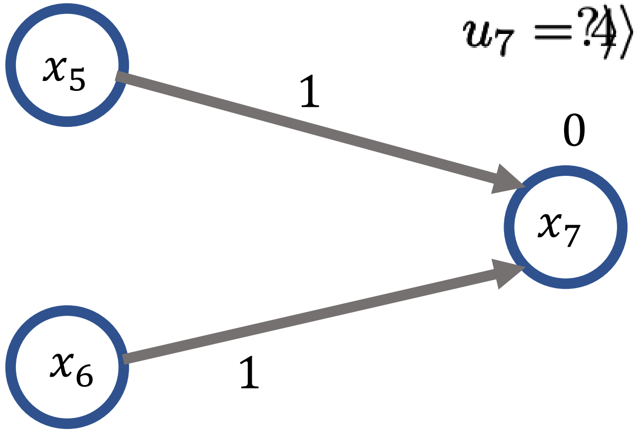
$\langle x_6 \ge 0,$
$x_6 \le 0.5 \cdot x_4 + 1,$
$l_6 = 0,$
$u_6 = 2\rangle$

Constant runtime

Affine transformation after ReLU

$$\langle x_5 \geq 0,$$
$$x_5 \leq 0.5 \cdot x_3 + 1,$$
$$l_5 = 0,$$
$$u_5 = 2\rangle$$

$$\langle x_7 \geq x_5 + x_6,$$
$$x_7 \leq x_5 + x_6,$$
$$l_7 = 0,$$
$$u_7 = 4\rangle$$



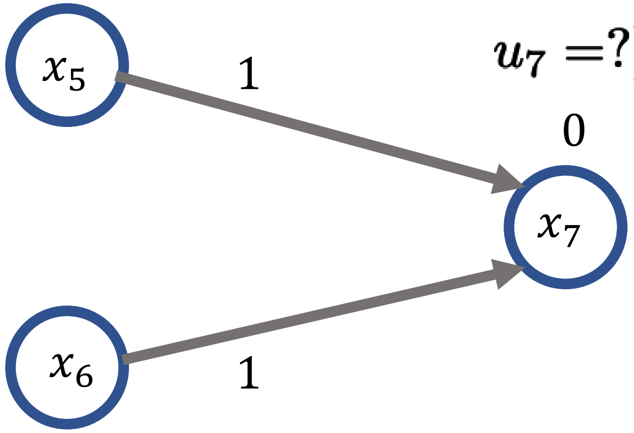
$$\langle x_6 \geq 0,$$
$$x_6 \leq 0.5 \cdot x_4 + 1,$$
$$l_6 = 0,$$
$$u_6 = 2\rangle$$

Imprecise upper bound u_7 by substituting u_5, u_6 for x_5 and x_6 in $a_7^{\geq}$

Backsubstitution

$$\langle x_5 \geq 0,$$
$$x_5 \leq 0.5 \cdot x_3 + 1,$$
$$l_5 = 0,$$
$$u_5 = 2\rangle$$

$$\langle x_7 \geq x_5 + x_6,$$
$$x_7 \leq 0.5 \cdot x_3 + 0.5 \cdot x_4 + 2,$$
$$l_7 = ?,$$
$$u_7 = ?\rangle$$

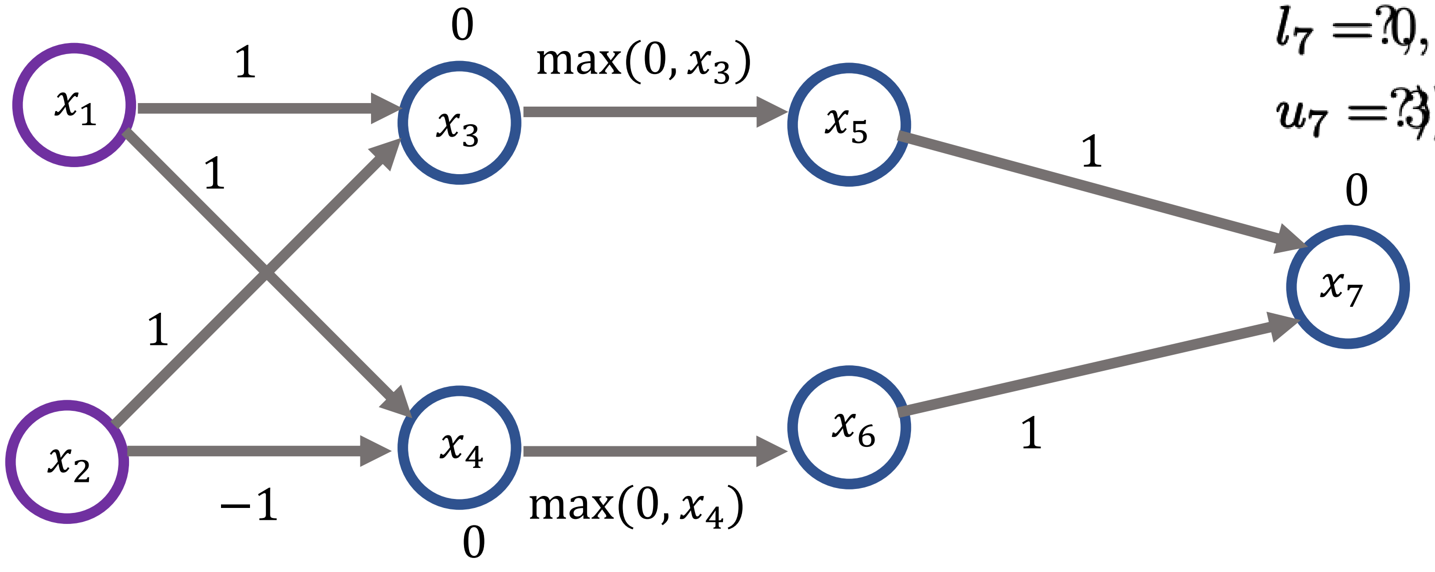


$$\langle x_6 \geq 0,$$
$$x_6 \leq 0.5 \cdot x_4 + 1,$$
$$l_6 = 0,$$
$$u_6 = 2\rangle$$

$$\langle x_1 \geq -1,\; x_1 \leq 1,\; l_1 = -1,\; u_1 = 1 \rangle$$

$$\langle x_5 \geq 0,\; x_5 \leq 0.5 \cdot x_3 + 1,\; l_5 = 0,\; u_5 = 2 \rangle$$

$$\langle x_7 \geq 0,\; x_7 \leq 0.5 \cdot x_3 + 0.5 \cdot x_4 + 2,\; l_7 = 0,\; u_7 = 3 \rangle$$



$$\langle x_2 \geq -1,\; x_2 \leq 1,\; l_2 = -1,\; u_2 = 1 \rangle$$

$$\langle x_6 \geq 0,\; x_6 \leq 0.5 \cdot x_4 + 1,\; l_6 = 0,\; u_6 = 2 \rangle$$

Affine transformation with backsubstitution is pointwise, complexity: $O(w_{max}^2 L)$

$$\langle x_1 \geq -1,\ x_1 \leq 1,\ l_1 = -1,\ u_1 = 1\rangle$$

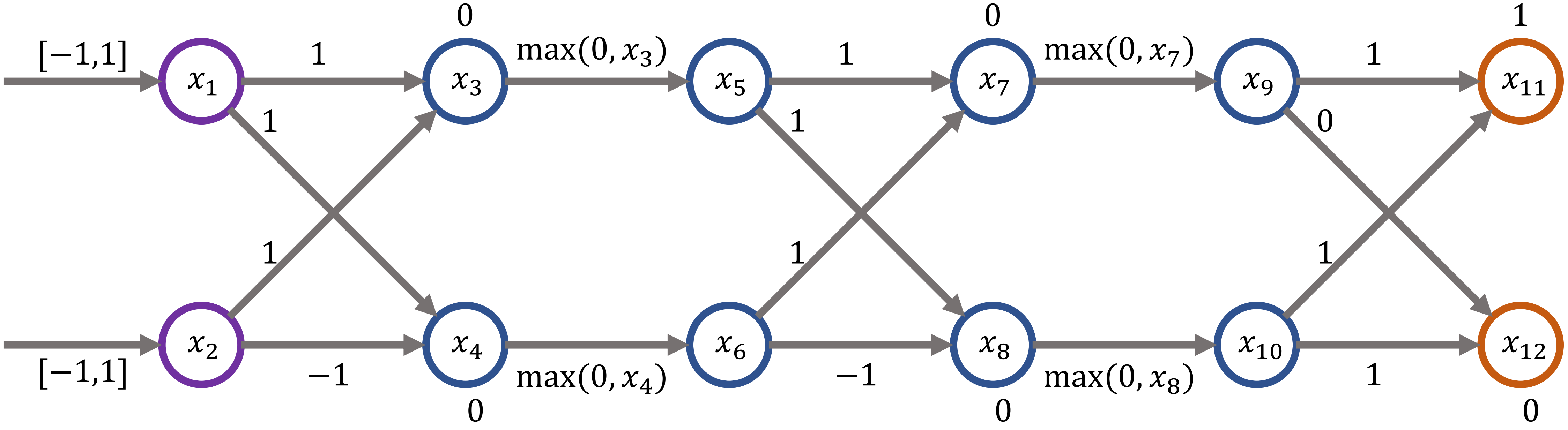
$$\langle x_3 \geq x_1 + x_2,\ x_3 \leq x_1 + x_2,\ l_3 = -2,\ u_3 = 2\rangle$$

$$\langle x_5 \geq 0,\ x_5 \leq 0.5 \cdot x_3 + 1,\ l_5 = 0,\ u_5 = 2\rangle$$

$$\langle x_7 \geq x_5 + x_6,\ x_7 \leq x_5 + x_6,\ l_7 = 0,\ u_7 = 3\rangle$$

$$\langle x_9 \geq x_7,\ x_9 \leq x_7,\ l_9 = 0,\ u_9 = 3\rangle$$

$$\langle x_{11} \geq x_9 + x_{10} + 1,\ x_{11} \leq x_9 + x_{10} + 1,\ l_{11} = 1,\ u_{11} = 5.5\rangle$$



$$\langle x_2 \geq -1,\ x_2 \leq 1,\ l_2 = -1,\ u_2 = 1\rangle$$

$$\langle x_4 \geq x_1 - x_2,\ x_4 \leq x_1 - x_2,\ l_4 = -2,\ u_4 = 2\rangle$$

$$\langle x_6 \geq 0,\ x_6 \leq 0.5 \cdot x_4 + 1,\ l_6 = 0,\ u_6 = 2\rangle$$

$$\langle x_8 \geq x_5 - x_6,\ x_8 \leq x_5 - x_6,\ l_8 = -2,\ u_8 = 2\rangle$$

$$\langle x_{10} \geq 0,\ x_{10} \leq 0.5 \cdot x_8 + 1,\ l_{10} = 0,\ u_{10} = 2\rangle$$

$$\langle x_{12} \geq x_{10},\ x_{11} \leq x_{10},\ l_{12} = 0,\ u_{12} = 2\rangle$$

Checking for robustness

Prove $x_{11} - x_{12} > 0$ for all inputs in $[-1,1] \times [-1,1]$

$$\langle x_{11} \geq x_9 + x_{10} + 1, \quad x_{11} \leq x_9 + x_{10} + 1, \quad l_{11} = 1, \quad u_{11} = 5.5\rangle$$

$$\langle x_{12} \geq x_{10}, \quad x_{11} \leq x_{10}, \quad l_{12} = 0, \quad u_{12} = 2\rangle$$

Computing lower bound for $x_{11} - x_{12}$ using l_{11}, u_{12} gives -1 which is an imprecise result

With backsubstitution, one gets 1 as the lower bound for $x_{11} - x_{12}$, proving robustness

Benchmarks

Dataset	Model	Type	#Neurons	#Layers	Defense
MNIST	6×100	feedforward	610	6	None
	6×200	feedforward	1,210	6	None
	9×200	feedforward	1,810	9	None
	ConvSmall	convolutional	3,604	3	DiffAI
	ConvBig	convolutional	34,688	6	DiffAI
	ConvSuper	convolutional	88,500	6	DiffAI
CIFAR10	ConvSmall	convolutional	4,852	3	DiffAI

DiffAI: trained to be more robust

Robusteness around input

% => % of input images such that robustness around them can be certified

Dataset	Model	ϵ	DeepZono		DeepPoly		RefineZono	
			% ✓	time(s)	% ✓	time(s)	% ✓	time(s)
MNIST	6 × 100	0.02	31	0.6	47	0.2	67	194
	6 × 200	0.015	13	1.8	32	0.5	39	567
	9 × 200	0.015	12	3.7	30	0.9	38	826
	ConvSmall	0.12	7	1.4	13	6.0	21	748
	ConvBig	0.2	79	7	78	61	80	193
	ConvSuper	0.1	97	133	97	400	97	665
CIFAR10	ConvSmall	0.03	17	5.8	21	20	21	550

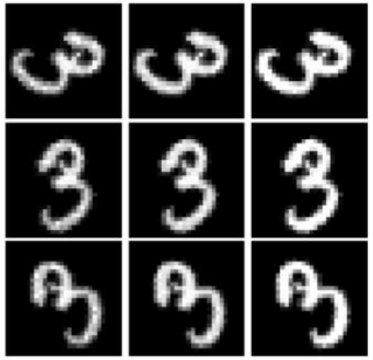
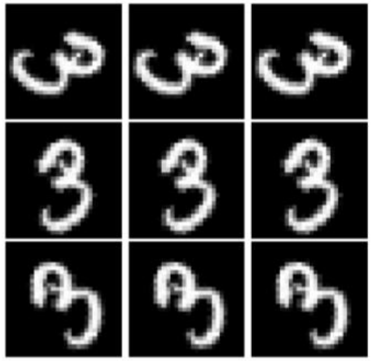
Partitioning the space with Batches is importants.

Test Robustness for:

Rotation of -45,+65°
Intensity of each pixel +/- 1%

=>

220 batches for different rotation (65°,64.5°)…
300 regions encoding different intensity for pixels

#Batches	Batch Size	Region(s) $(l, \frac{1}{2}(l+u), u)$	Analysis time	Verified?
1	1		0.5s + 1.9s	No
1	10000		22.2s + 1.8s	No
220	1		1.2s + 5m51s	No
220	300		2m29s + 5m30s	Yes

Conclusion

Attacks on Deep Learning

The self-driving car incorrectly decides to turn right on Input 2 and crashes into the guardrail

(a) Input 1 (b) Input 2 (darker version of 1)

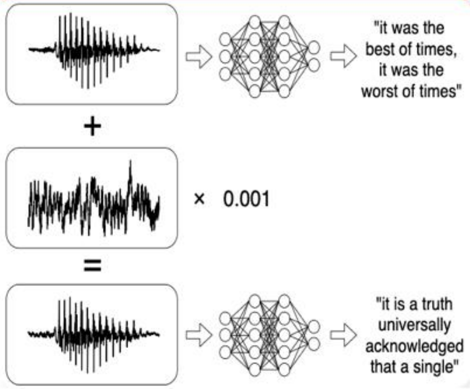
DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP'17

The Ensemble model is fooled by the addition of an adversarial distracting sentence in blue.

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Adversarial Examples for Evaluating Reading Comprehension Systems, EMNLP'17

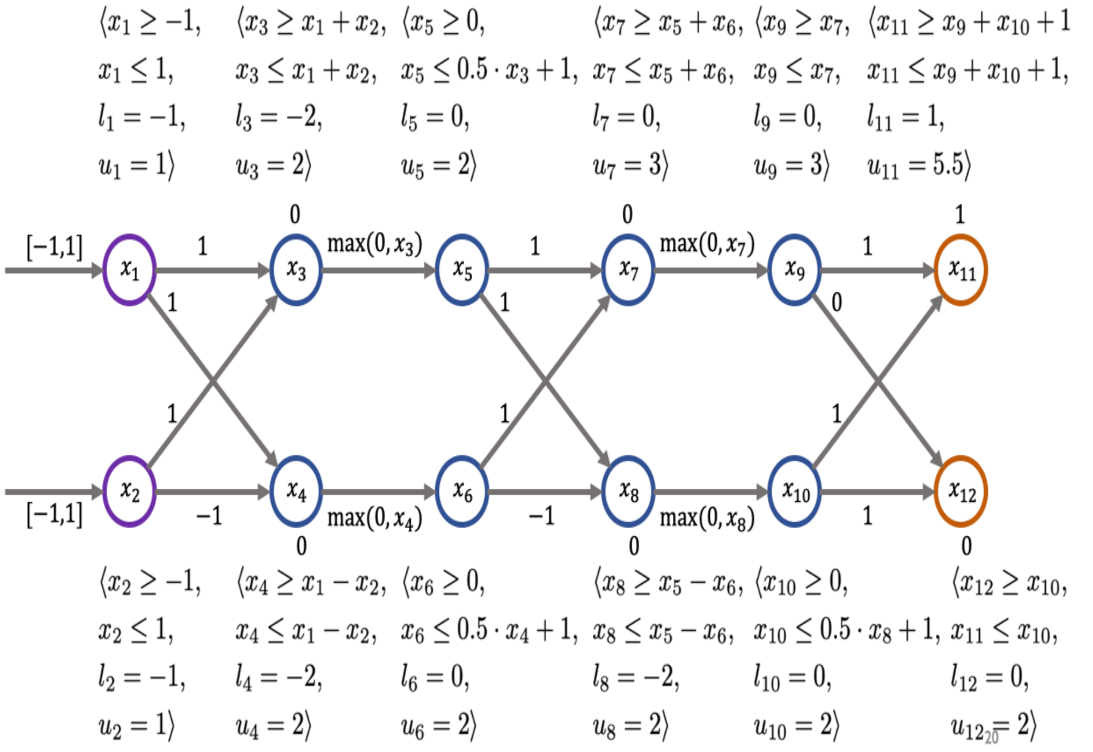
Adding small noise to the input audio makes the network transcribe any arbitrary phrase



Audio Adversarial Examples: Targeted Attacks on Speech-to-Text, ICML 2018

Our analyzer ERAN is publicly available at https://github.com/eth-sri/eran

Feedforward (FNN), convolutional (CNN), and residual networks
ReLU, sigmoid, and tanh activations
Sound with respect to floating point arithmetic
State-of-the-art precision and performance



Aircraft collision avoidance system

Reluplex	Neurify	ERAN
> 32 hours	921 sec	227 sec