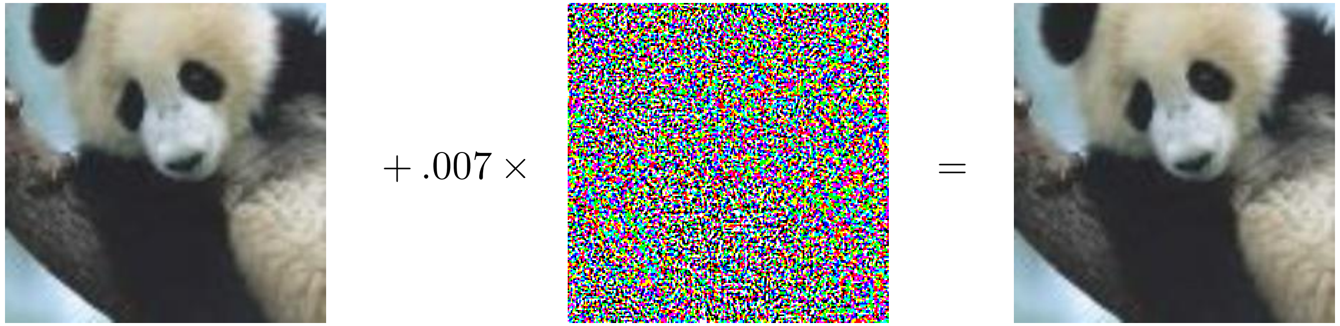
Provable defenses against adversarial attacks

Eric Wong
Joint work with Zico Kolter

AI Seminar

02/26/2019

Adversarial examples can fool deep networks



$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

[Goodfellow et. Al., 2014]

Defending networks is harder than attacking networks

Defenses	Attacks
Distillation [Papernot et al. 2015]	Distillation not safe [Carlini & Wagner 2016]
	Real world objects are attackable [Sharif et al. 2016, Kurakin et al. 2016]
Can detect adversarial examples [10+ methods]	Detection methods fail [Metzen et al. 2017, Carlini & Wagner 2017]
Realistic rotations and translations are safe [Lu et al. 2017]	Realistic adversarial examples exist [Athalye & Sutskever, 2017]
	Black-box attacks are effective [Papernot et al. 2017]
9 defenses at ICLR 2018 Adversarial training with PGD [Madry et al. 2018]	Most heuristics broken before review cycle ends [Athalye et al. 2018]

Newer defenses combine with [Madry et al. 2018]

Non-heuristic defenses

Formal methods (SMT, integer programming, SAT solving, etc.)

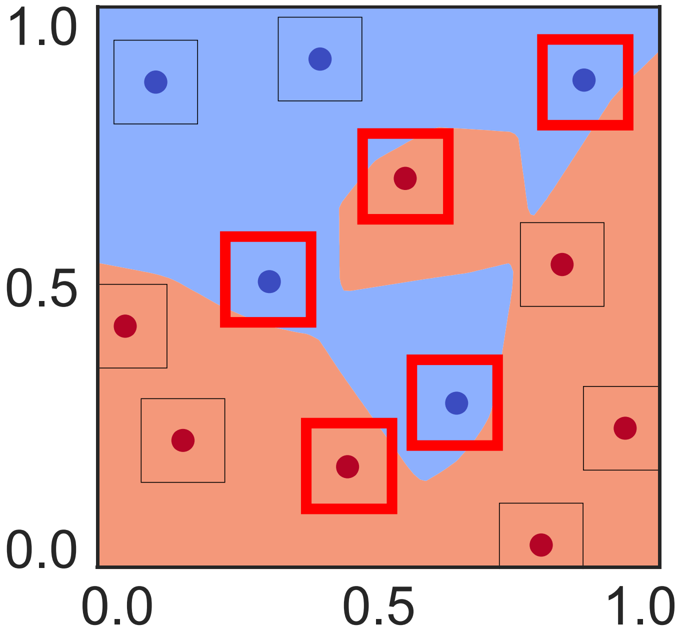
- e.g., Carlini et al., 2017; Ehlers 2017; Katz et al., 2017; Huang et al., 2017, Tjeng et al., 2017
- Limited in scalability to small networks by combinatorial optimization

Our work: tractable, provable defenses

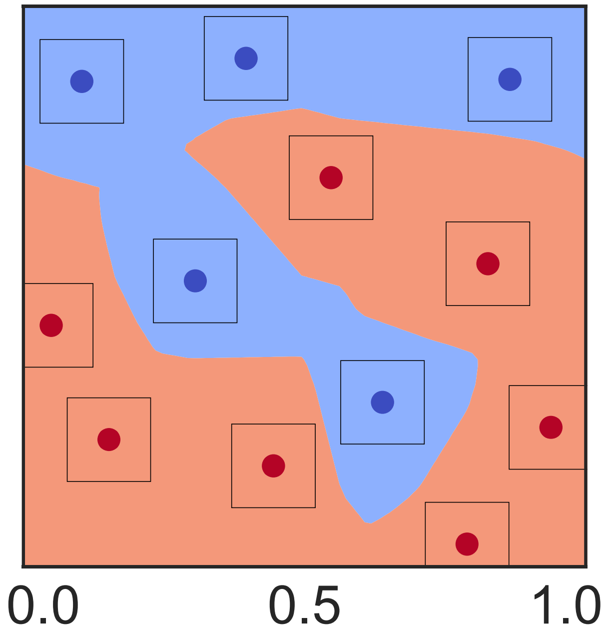
- Related: **Raghunathan et al., 2018**; Staib and Jegelka 2017; Sinha et al., 2017; Hein and Andriushchenko 2017; Madry et al., 2017; Peck et al, 2017; Krishnamurty, 2018; Gehr et al., 2018; **Mirman et al., 2018, Gowal et al., 2018**

Guaranteed, provable defenses can stop the attackers once and for all

2D visualization of the end goal



Typical classifier

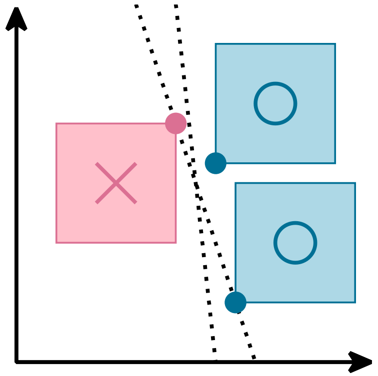


Robust classifier

Would like a “proof” or certificate that our classifier is robust

Use robust optimization to learn robust classifiers

Linear classifiers:



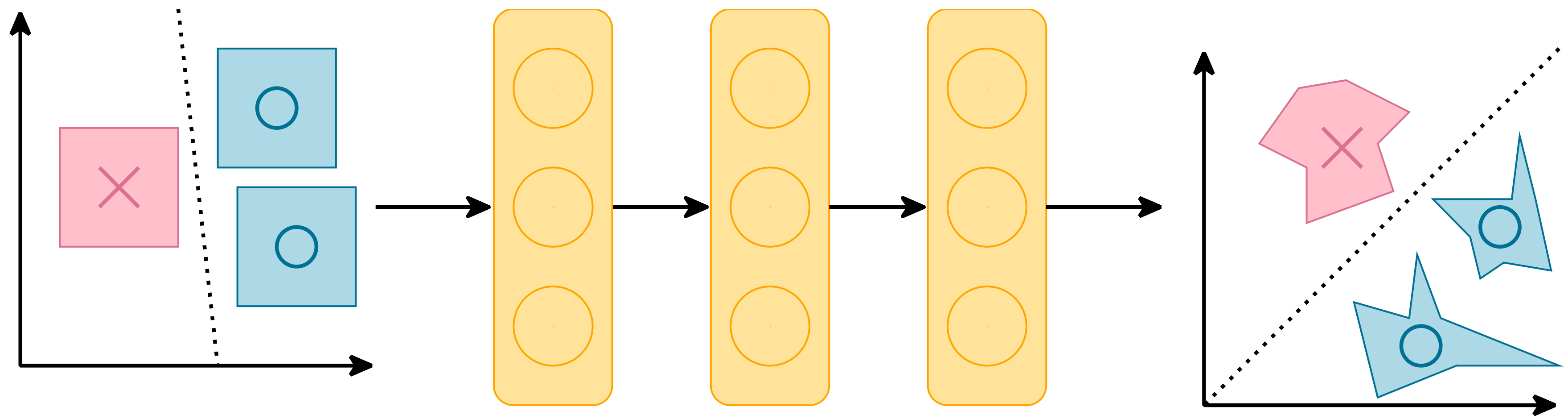
$$\underset{\theta}{\text{minimize}} \sum_i \max_{\|\Delta\|_\infty \le \epsilon} \ell(h_\theta(x_i + \Delta) \cdot y_i)$$

$$\equiv \underset{\theta}{\text{minimize}} \sum_i \ell(h_\theta(x_i) \cdot y_i + \epsilon \|\theta\|_1)$$

An area of optimization that goes back almost 50 years [Soyster, 1973]

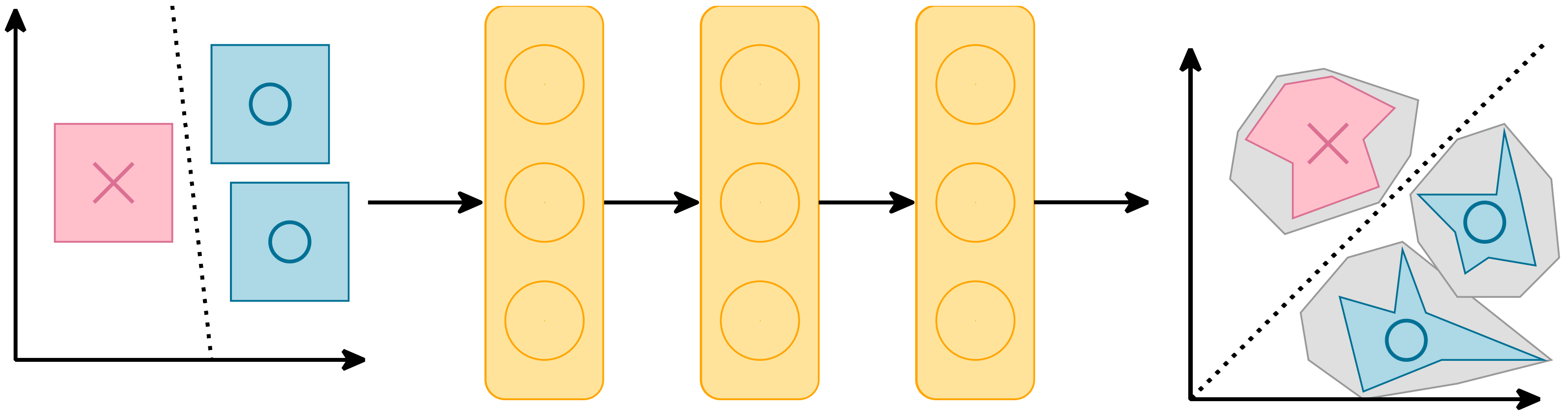
Robust optimization for deep networks

$$\underset{\theta}{\text{minimize}} \sum_i \max_{\|\Delta\|_\infty \le \epsilon} \ell(h_\theta(x_i + \Delta), y_i)$$



How to optimize the worst case over the "image" of the perturbation (adversarial polytope)?

How to optimize?



$$\underset{\theta}{\text{minimize}} \sum_i \max_{\|\Delta\|_\infty \le \epsilon} \ell(h_\theta(x_i + \Delta), y_i) \le \underset{\theta}{\text{minimize}} \sum_i \ell(J(x_i), y_i)$$

Problem: this polytope is non-convex and difficult to optimize over exactly

Our approach: bound the inner maximization with a closed form, using a convex outer bound over the adversarial polytope, and duality

Convex outer bounds for the adversarial polytope

Worst-case adversarial attack as an optimization problem

For a deep network $h_\theta(x)$ and threat model $\{\, x + \Delta : \; ||\Delta||_\infty \leq \epsilon \,\}$:

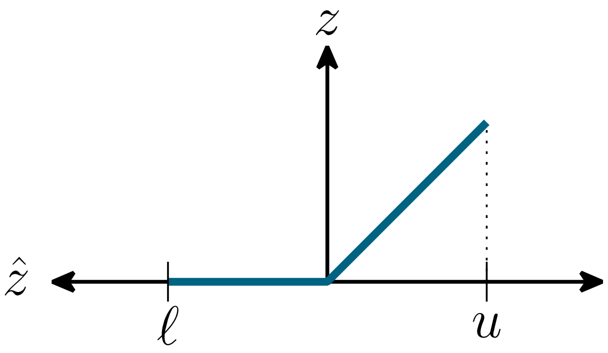
Targeted, multiclass attack for ReLU networks:

$$\max_{||\Delta||_\infty \leq \epsilon} \; h_\theta(x+\Delta)_{target} \; - h_\theta(x+\Delta)_y$$

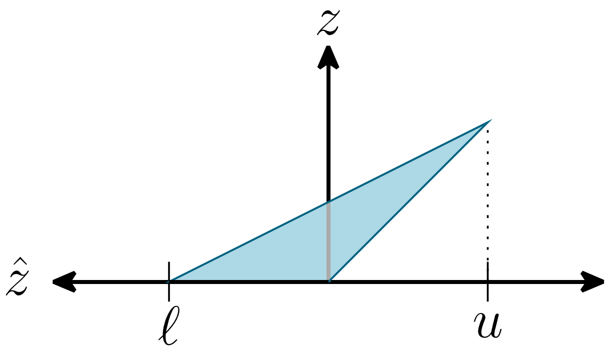
$$\begin{aligned} \equiv \underset{z}{\text{minimize}} \quad & \left(e_y - e_{target}\right)^T z_k \\ \text{subject to} \quad & \|z_1 - x\|_\infty \leq \epsilon \\ & z_{i+1} = \max(0, W_i z_i + b_i) \\ & z_k = W_{k-1} z_{k-1} + b_{k-1} \end{aligned}$$

ReLU constraint is non-convex: replace with a convex set

Convex relaxation for ReLU constraints



Bounded ReLU set



Convex relaxation

<u>Exact adversarial attack</u>

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & (e_y - e_{target})^T z_k \\ \text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\ & z_{i+1} = \max(0, W_i z_i + b_i) \\ & z_k = W_{k-1} z_{k-1} + b_{k-1} \end{aligned}$$

<u>Bound on adversarial attack</u>

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & (e_y - e_{target})^T z_k \\ \text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\ & (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i) \\ & z_k = W_{k-1} z_{k-1} + b_{k-1} \end{aligned}$$

This is now a *linear program*

Linear programs are expensive for deep networks

$$
\begin{aligned}
\underset{z}{\text{minimize}} \quad & (e_y - e_{target})^T z_k \\
\text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\
& (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i) \\
& z_k = W_{k-1} z_{k-1} + b_{k-1}
\end{aligned}
$$

This linear program has a variable for each hidden unit. For deep networks, this can easily exceed hundreds of thousands!*

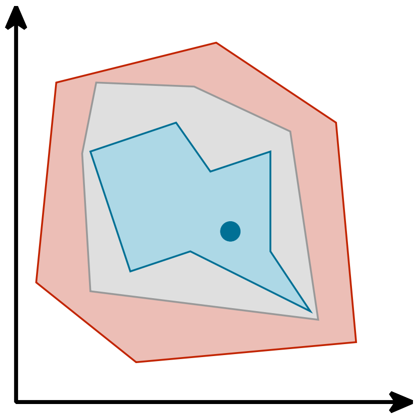
*Actually, in a paper released 3 days ago, the primal was solved for medium sized networks on clusters with 1000 CPU cores [Salman et al. 2019]

Fast duality-based bounds

Bound the linear program with its dual linear program

Not practical to solve these linear programs via standard methods

So, consider the *dual problem*: any dual feasible point provides a guaranteed bound on the original problem



<u>Adversarial Problem</u>

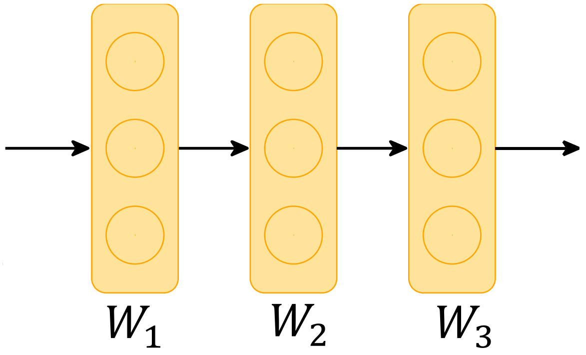
$$\begin{aligned} \underset{z}{\text{minimize}} \quad & (e_y - e_{target})^T z_k \\ \text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\ & (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i) \\ & z_k = W_{k-1} z_{k-1} + b_{k-1} \end{aligned}$$

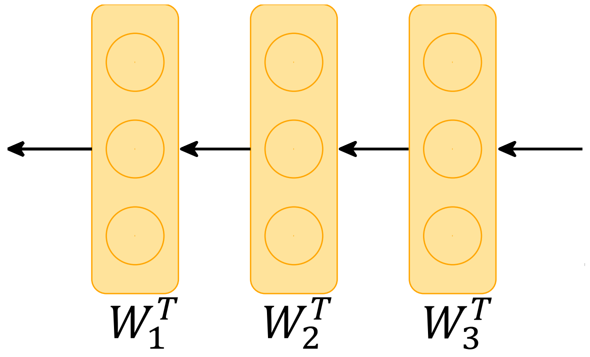
$\Rightarrow$

<u>Dual Problem</u>

$$\begin{aligned} \underset{\nu,\alpha}{\text{maximize}} \quad & J(\nu, x) \\ \text{subject to} \quad & \nu_k = e_{target} - e_y \\ & \nu_i = f_i\left(W_i^T \nu_{i+1}, \alpha; \ell_i, u_i\right) \\ & \nu_1 = W_1^T \nu_2 \end{aligned}$$

The dual linear program is a deep network





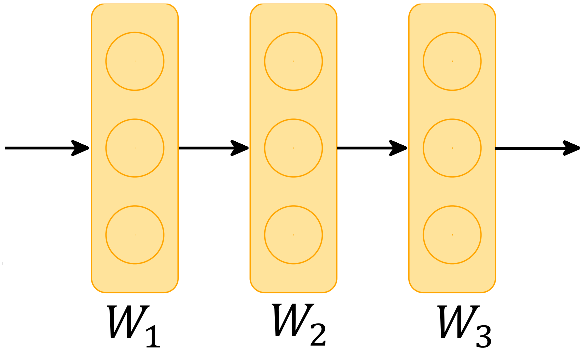
<u>Adversarial Problem</u>

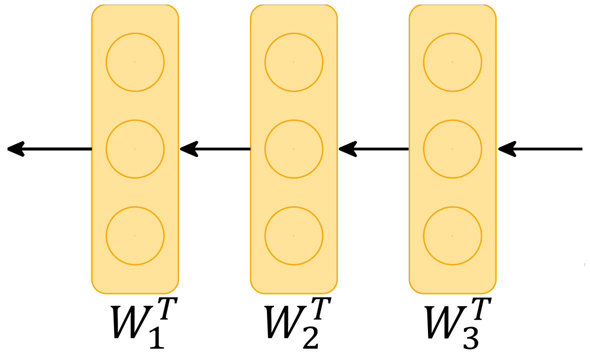
$$
\begin{aligned}
\underset{z}{\text{minimize}} \quad & (e_y - e_{target})^T z_k \\
\text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\
& (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i) \\
& z_k = W_{k-1} z_{k-1} + b_{k-1}
\end{aligned}
$$

<u>Dual Problem</u>

$$
\begin{aligned}
\underset{\nu,\alpha}{\text{maximize}} \quad & J(\nu, x) \\
\text{subject to} \quad & \nu_k = e_{target} - e_y \\
& \nu_i = f_i\left(W_i^T \nu_{i+1}, \alpha; \ell_i, u_i\right) \\
& \nu_1 = W_1^T \nu_2
\end{aligned}
$$

The dual component of a linear layer is the transpose





<u>Adversarial Problem</u>

$$
\begin{aligned}
\underset{z}{\text{minimize}} \quad & (e_y - e_{target})^T z_k \\
\text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\
& (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i) \\
& z_k = W_{k-1} z_{k-1} + b_{k-1}
\end{aligned}
$$

<u>Dual Problem</u>

$$
\begin{aligned}
\underset{\nu,\alpha}{\text{maximize}} \quad & J(\nu, x) \\
\text{subject to} \quad & \nu_k = e_{target} - e_y \\
& \nu_i = f_i\left(W_i^T \nu_{i+1}, \alpha; \ell_i, u_i\right) \\
& \nu_1 = W_1^T \nu_2
\end{aligned}
$$

The dual of the ReLU depends on the (ℓ, u) bounds

$$f_i(\nu, \alpha; \ell_i, u_i)_j = \begin{cases} 0 & \text{if } u_{ij} \le 0 \\ \nu_j & \text{if } \ell_{ij} \ge 0 \\ \dfrac{u_{ij}}{u_{ij} - \ell_{ij}} [\nu_{ij}]_+ - \alpha_{ij} [\nu_{ij}]_- & \text{if } \ell_{ij} < 0 < u_{ij} \end{cases}$$

<u>Adversarial Problem</u>

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & (e_y - e_{target})^T z_k \\ \text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\ & (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i) \\ & z_k = W_{k-1} z_{k-1} + b_{k-1} \end{aligned}$$

<u>Dual Problem</u>

$$\begin{aligned} \underset{\nu, \alpha}{\text{maximize}} \quad & J(\nu, x) \\ \text{subject to} \quad & \nu_k = e_{target} - e_y \\ & \nu_i = f_i(W_i^T \nu_{i+1}, \alpha; \ell_i, u_i) \\ & \nu_1 = W_1^T \nu_2 \end{aligned}$$

Any dual feasible solution bounds the primal problem

$$f_i(\nu, \alpha; \ell_i, u_i)_j = \begin{cases} 0 & \text{if } u_{ij} \leq 0 \\ \nu_j & \text{if } \ell_{ij} \geq 0 \\ \dfrac{u_{ij}}{u_{ij} - \ell_{ij}} [\nu_{ij}]_+ - \alpha_{ij} [\nu_{ij}]_- & \text{if } \ell_{ij} < 0 < u_{ij} \end{cases}$$

Taking $\alpha_{ij} = \frac{u_{ij}}{u_{ij} - \ell_{ij}}$ makes the entire dual ReLU activation linear

$$f_i(\nu; \ell_i, u_i)_j = \begin{cases} 0 & \text{if } u_{ij} \leq 0 \\ \nu_j & \text{if } \ell_{ij} \geq 0 \\ \dfrac{u_{ij}}{u_{ij} - \ell_{ij}} \nu_{ij} & \text{if } \ell_{ij} < 0 < u_{ij} \end{cases}$$

Takeaway: we can bound the worst-case adversarial output with a single pass through the dual network

Let $g(c)$ be the dual network defined by the equations

$$\begin{aligned} \nu_k &= -c \\ \nu_i &= f_i(W_i^T \nu_{i+1}; \ell_i, u_i) \\ \nu_1 &= W_1^T \nu_2 \end{aligned}$$

Take $\mathbf{c} = e_y - \; e_{target}$ and let $\nu = g(c)$ be a pass through the dual network.

Then, our bound on the worst-case adversarial attack is

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & c^T z_k \\ \text{subject to} \quad & \|z_1 - x\|_\infty \leq \epsilon \\ & z_{i+1} = \max(0, W_i z_i + b_i) \\ & z_k = W_{k-1} z_{k-1} + b_{k-1} \end{aligned} \qquad \geq J(\nu)$$

Some skipped details

What about bounding the ReLU pre-activations?

Last piece of the puzzle: how to compute (ℓ_i, u_i) for intermediate ReLU activations?

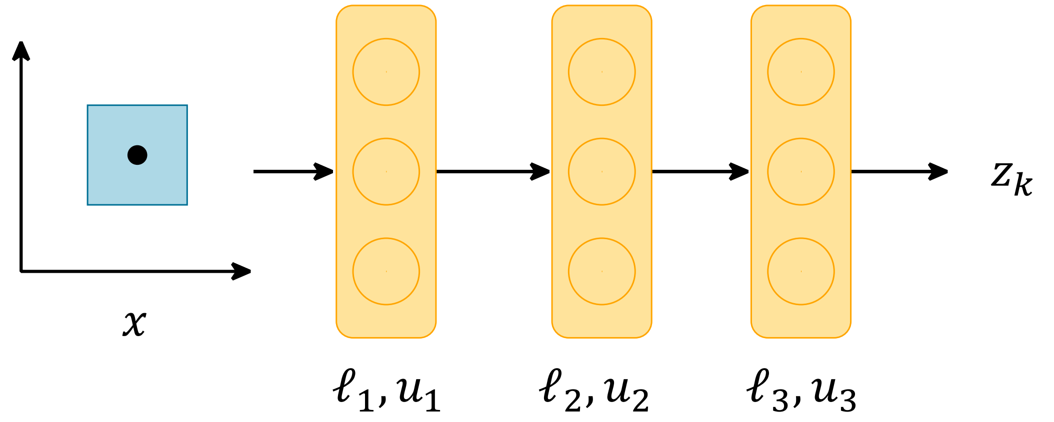
Recursively apply the same procedure!

Take $c = e_j$ and let $\nu = g(c)$ be a pass through the dual network up to the kth layer. Then, the lower bound for the jth activation in the kth layer is

$$
\begin{aligned}
\underset{z}{\text{minimize}} \quad & c^T z_k \\
\text{subject to} \quad & \|z_1 - x\|_\infty \le \epsilon \\
z_{i+1} &= \max(0, W_i z_i + b_i) \\
z_k &= W_{k-1} z_{k-1} + b_{k-1}
\end{aligned}
\qquad \ge J(\nu) = \ell_{kj}
$$

Can compute all lower and upper bounds in a single pass

We can cache intermediate results to iteratively build all lower and upper bounds with a single pass through the same dual network



Runtime: quadratic in # of hidden units, but in subsequent work (Wong & Kolter, 2018) we make this linear using random projections

Exact form of the dual objective

$$\underset{\nu,\alpha}{\text{maximize}} \quad J_{\epsilon,W,b}(\nu,x) \equiv \boxed{-\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1} \boxed{-\epsilon\|\hat{\nu}_1\|_1} \boxed{+\sum_{i=1}^{k-1}\sum_{j\in\mathcal{I}_i} \ell_{i,j}[\nu_{i,j}]_+}$$

$$\begin{aligned}
\text{subject to} \quad & \nu_k = e_{target} - e_y \\
& \nu_i = f_i(W^T \nu_{i+1}, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2 \\
& \nu_1 = W_1^T \nu_2
\end{aligned}$$

Objective at $\boldsymbol{\epsilon = 0}$

Robustness penalty (same as in linear case)

Additional penalty for the convex outer bound on the ReLU activation

Learning provably robust networks

Traditional network training minimizes empirical loss

Traditional loss on the network outputs:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i)$$

Provably robust training minimizes robust loss

Traditional loss on the network outputs:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i)$$

Robust loss on the worst case outputs:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \ell(J_{\epsilon,\theta}(x_i), y_i)$$

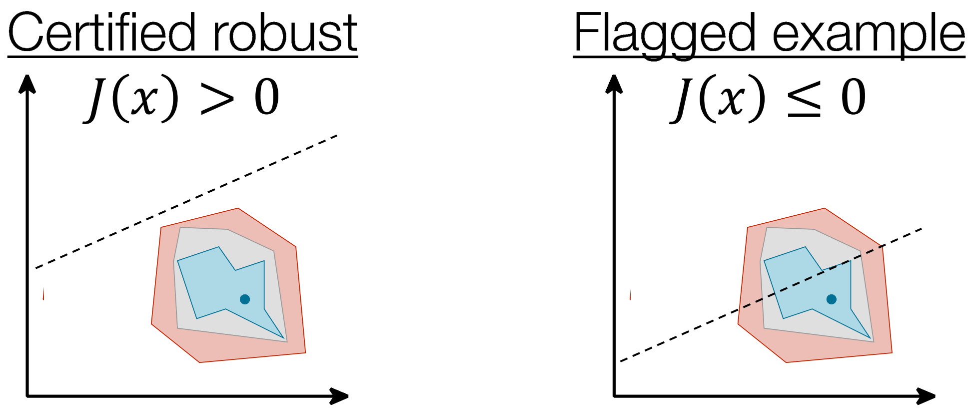
Advantages

- Drop in replacement
- Uses standard deep learning tools (dual network)
- *Guaranteed bound* on the worst case loss (or error) for *any* norm-bounded adversarial attack
- Runtime: one pass to compute intermediate bounds, one pass to compute dual objective

Examples can be certified at test time

Adversarial example in ball around original x $\Leftrightarrow$ original x in ball around adversarial example

At test time, evaluate the bound to see if example is possibly adversarial

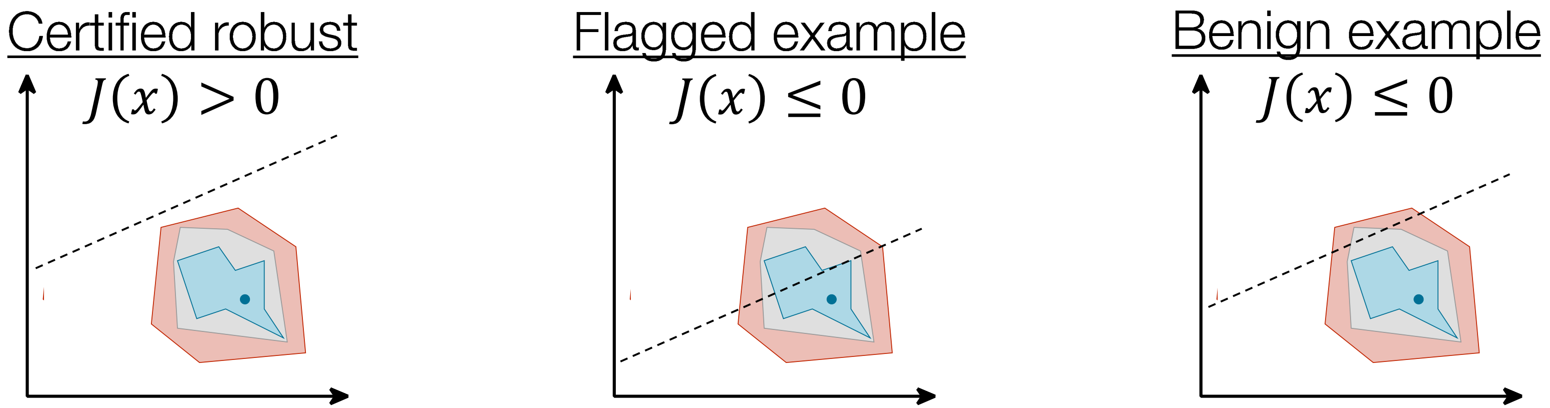


If bound is less than 0, then the example is guaranteed to be robust to any adversarial attack

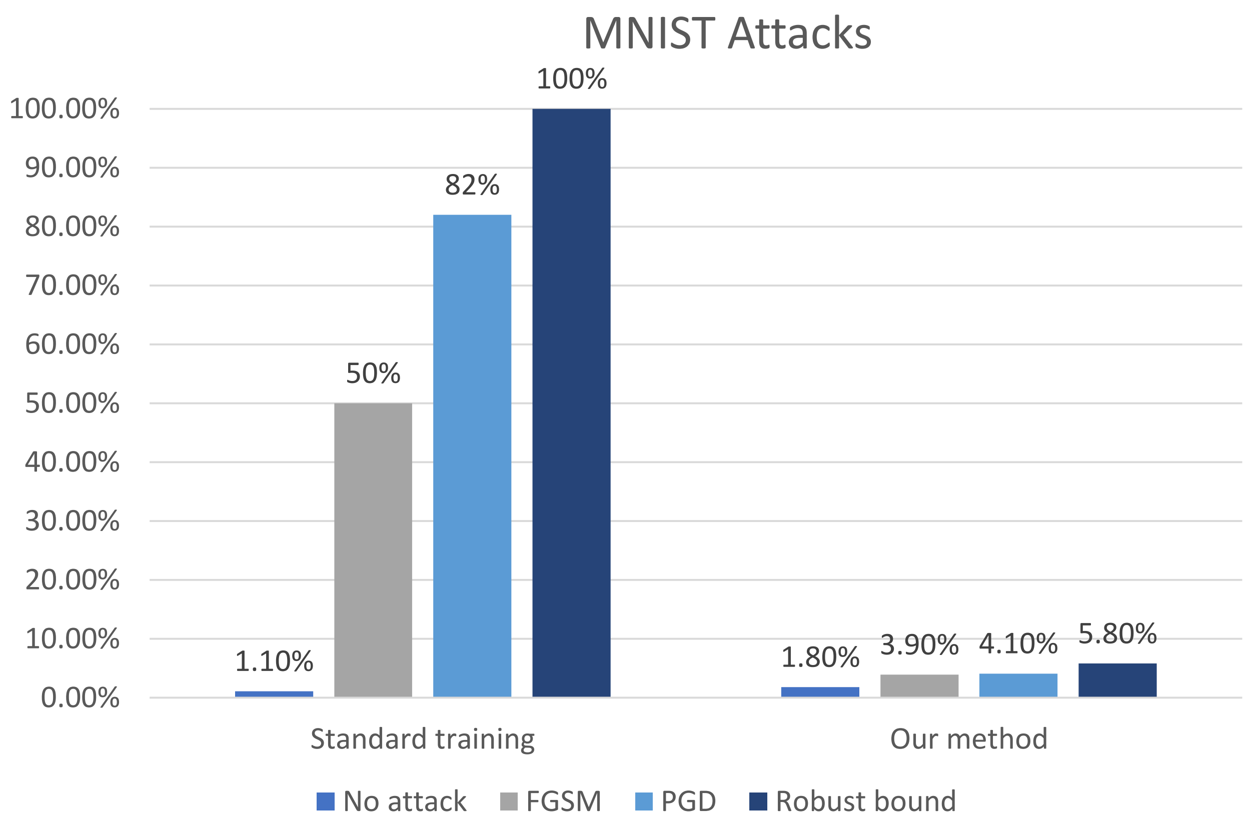
Examples can be certified at test time

Two properties:

- Zero false negatives (the bound is a proof that the example is “safe”)
- Might flag benign examples



The certified results don't contradict adversarial attacks



Similar results on other datasets

Problem	Robust	ϵ	Test error	FGSM error	PGD error	Robust error bound
MNIST	$\times$	0.1	1.07%	50.01%	81.68%	100%
MNIST	$\surd$	0.1	1.80%	3.93%	4.11%	5.82%
Fashion-MNIST	$\times$	0.1	9.36%	77.98%	81.85%	100%
Fashion-MNIST	$\surd$	0.1	21.73%	31.25%	31.63%	34.53%
HAR	$\times$	0.05	4.95%	60.57%	63.82%	81.56%
HAR	$\surd$	0.05	7.80%	21.49%	21.52%	21.90%
SVHN	$\times$	0.01	16.01%	62.21%	83.43%	100%
SVHN	$\surd$	0.01	20.38%	33.28%	33.74%	40.67%

Robustness comes at a cost to standard accuracy

On later work scaling to CIFAR10 architectures, we can guarantee at least **54%** adversarial test accuracy against $\epsilon = 2/255$ perturbations.

However, the clean test accuracy is **68%**: way below state of the art!

The bounds are only tight when trained against bound

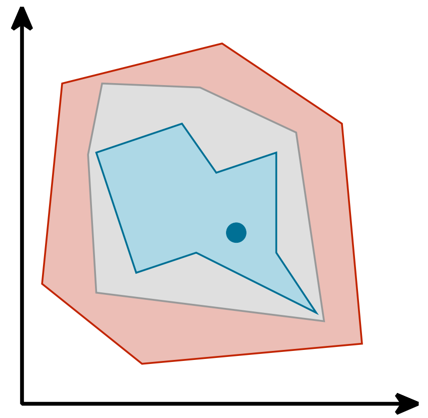
Raghunathan et al. (2018) use a trainable SDP based bound for 2-layer neural networks

If we evaluate our bound on their trained network (or vice versa), we get vacuous bounds

Network	PGD error	SDP bound	LP bound
SDP-NN	15%	35%	99%
LP-NN	22%	93%	26%

Bound introduces two layers of looseness

1. Exact adversarial problem
2. Convex relaxation to a linear program
3. Dual feasible solutions



*Recent work shows that most looseness comes from the LP relaxation [Salman et al., 2018]

Code and model weights are available on GitHub

https://github.com/locuslab/convex_adversarial

Questions?