15-780 – Graduate Artificial Intelligence: Adversarial attacks and provable defenses

J. Zico Kolter (this lecture) and Ariel Procaccia
Carnegie Mellon University
Spring 2018

Portions base upon joint work with Eric Wong

Outline

Adverarial attacks on machine learning

Robust optimization

Provable defenses for deep classifiers

Experimental results

Outline

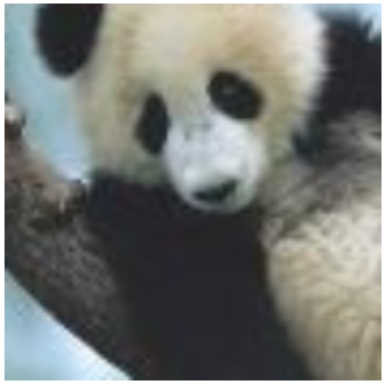
Adverarial attacks on machine learning

Robust optimization

Provable defenses for deep classifiers

Experimental results

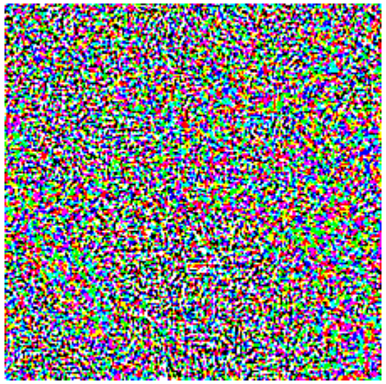
Adversarial attacks



$\boldsymbol{x}$

"panda"
57.7% confidence

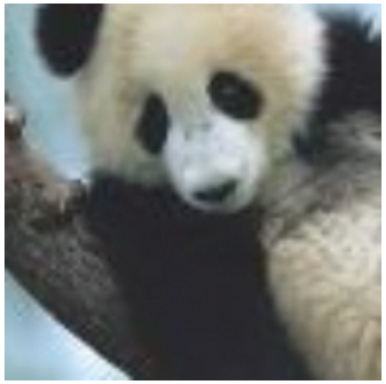
$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$



$\boldsymbol{x} +$
$\epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

[Szegedy et al., 2014, Goodfellow et al., 2015]

How adversarial attacks work

We are focusing on *test time* attacks: train on clean data and attackers tries to fool the trained classifier at test time

To keep things tractable, we are going to restrict our attention to ℓ_∞ *norm bounded attacks:* the adversary is free to manipulate inputs within some ℓ_∞ ball around the true example

$$\tilde{x} = x + \Delta, \qquad \|\Delta\|_\infty \leq \epsilon$$

Basic method: given input $x \in \mathcal{X}$, output $y \in \mathcal{Y}$, hypothesis $h_\theta : \mathcal{X} \to \mathcal{Y}$, and loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, adjust x to maximum loss:

$$\underset{\|\Delta\|_\infty \leq \epsilon}{\text{maximize}} \ \ell(h_\theta(x + \Delta), y)$$

Other variants we will see shortly (e.g., maximizing specific target class)

A summary of adversarial example research

🙂 Distillation prevents adversarial attacks! [Papernot et al., 2016]

🙁 No it doesn't! [Carlini and Wagner, 2017]

🙂 No need to worry given translation/rotation! [Lu et al., 2017]

🙁 Yes there is! [Athalye and Sutskever, 2017]

🙂 We have 9 new defenses you can use! [ICLR 2018 papers]

🙁 Broken before review period had finished! [Athalye et al., 2018]

My view: the attackers are winning, we need to get out of this arms race

A slightly better summary

Many heuristic methods for defending against against adversarial examples [e.g., Goodfellow et al., 2015; Papernot et al., 2016; Madry et al., 2017; Tramér et al., 2017; Roy et al., 2017]

- Keep getting broken, unclear if/when we'll find the right heuristic

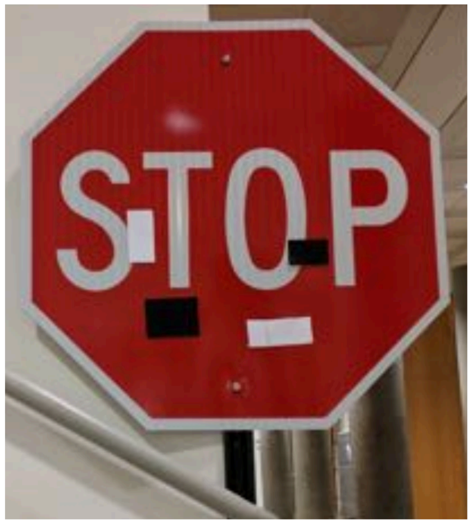
Formal methods approaches to verifying networks via tools from SMT, integer programming, SAT solving, etc. [e.g., Carlini et al., 2017; Ehlers 2017; Katz et al., 2017; Huang et al., 2017]

- Limited to small networks by combinatorial optimization

Our work: *Tractable, provable defenses* against adversarial examples via convex relaxations [also related: Raghunathan et al., 2018; Staib and Jegelka 2017; Sinha et al., 2017; Hein and Andriushchenko 2017; Peck et al, 2017]

Adversarial examples in the real world

Sharif et al., 2016



Evtimov et al., 2017

Athalye et al., 2017

Note: only the last one here is possibly an ℓ_∞ perturbation

The million dollar question

How can we design (deep) classifiers that are provably robust to adversarial attacks?

Outline

Adverarial attacks on machine learning

Robust optimization

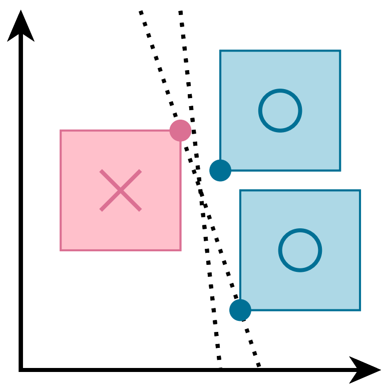
Provable defenses for deep classifiers

Experimental results

Robust optimization

A area of optimization that goes almost 50 years [Soyster, 1973; see Ben-Tal et al., 2011]

Robust optimization (as applied to machine learning): instead of minimizing loss at training points, minimize *worst case* loss in some ball around the points



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

$$\equiv \operatorname*{minimize}_{\theta} \sum_i \ell(h_\theta(x_i) \cdot y_i - \epsilon\|\theta\|_1)$$

(for *linear* classifiers)

Proof of robust machine learning property

Lemma: For linear hypothesis function $h_\theta(x) = \theta^T x$, binary output $y \in \{-1, +1\}$, and classification loss $\ell(h_\theta(x) \cdot y)$

$$\max_{\|\Delta\|_\infty \le \epsilon} \ell(h_\theta(x + \Delta) \cdot y) = \ell(h_\theta(x) \cdot y - \epsilon\|\theta\|_1)$$

Proof: Because classification loss is monotonic decreasing

$$\begin{aligned}\max_{\|\Delta\|_\infty \le \epsilon} \ell(h_\theta(x + \Delta) \cdot y) &= \ell\left(\min_{\|\Delta\|_\infty \le \epsilon} h_\theta(x + \Delta) \cdot y\right) \\ &= \ell\left(\min_{\|\Delta\|_\infty \le \epsilon} \theta^T(x + \Delta) \cdot y\right)\end{aligned}$$

Theorem follows from the fact that

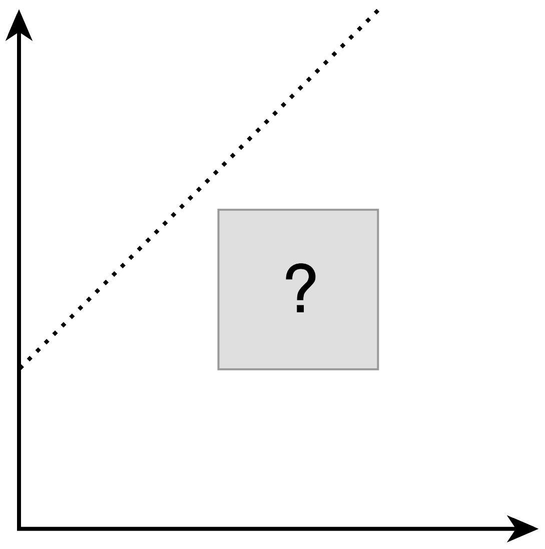
$$\min_{\|\Delta\|_\infty \le \epsilon} \theta^T \Delta = -\epsilon\|\theta\|_1$$

■

What to do at test time?

This procedure prevents the possibility of adversarial examples at training time, but what about at test time?

Basic idea: If we make a prediction at a point, and this prediction does not change within the ℓ_∞ ball of ϵ around the point, then this cannot be an adversarial example (i.e., we have a *zero-false negative detector*)



Outline

Adverarial attacks on machine learning

Robust optimization

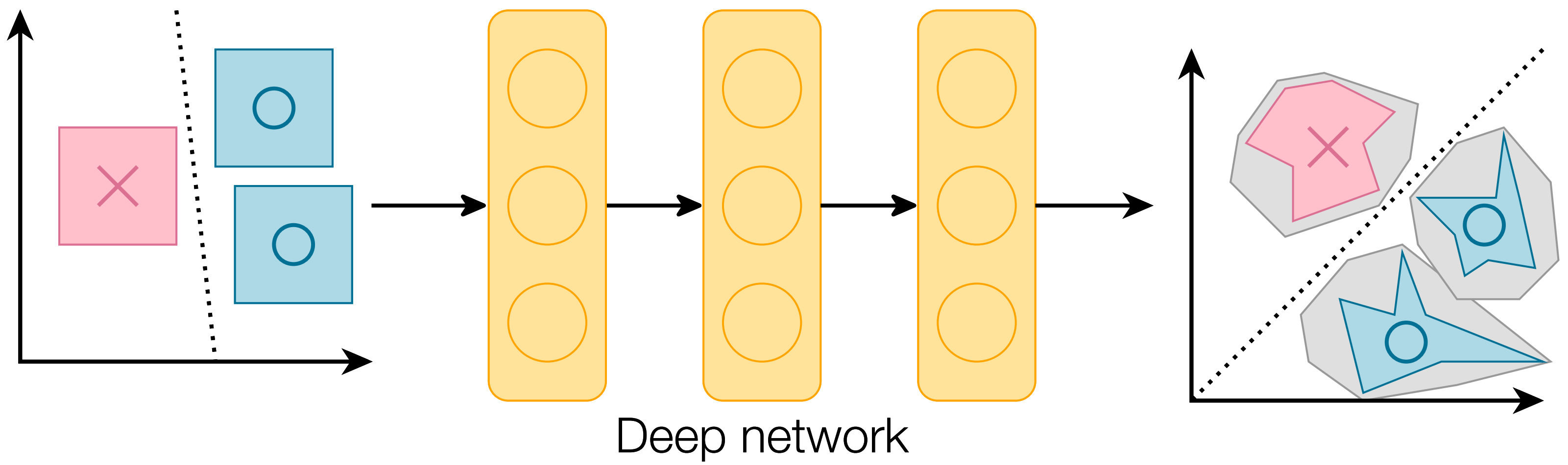
Provable defenses for deep classifiers

Experimental results

Based upon work in:
Wong and Kolter, “Provable defenses against adversarial examples via the convex adversarial polytope”, 2017
https://arxiv.org/abs/1711.00851

The trouble with deep networks

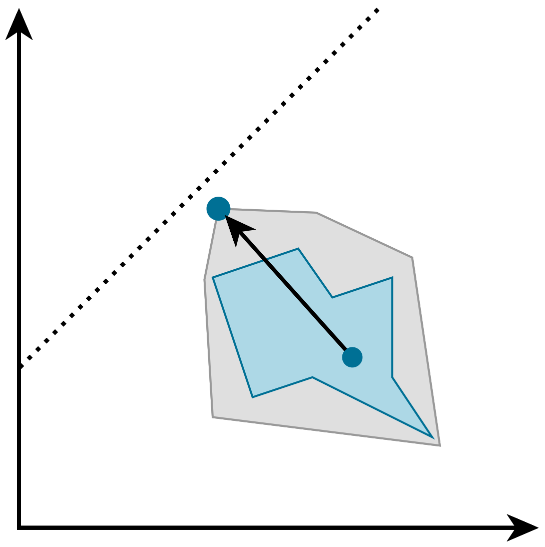
In deep networks, the “image” (adversarial polytope) of a norm bounded perturbation is non-convex, we can’t easily optimize over it



Our approach: instead, form *convex outer bound* over the adversarial polytope, and perform robust optimization over this region (applies specifically to networks with ReLU nonlinearities)

Convex outer approximations

Optimization over convex outer adversarial polytope provides *guarantees* about robustness to adversarial perturbations

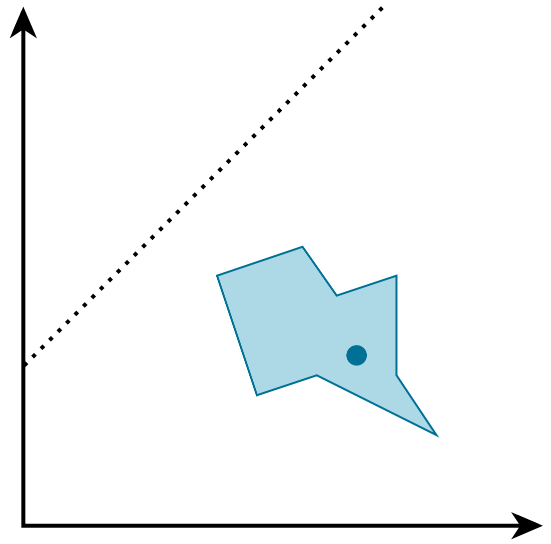


… so, how do we compute and optimize over this bound?

Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

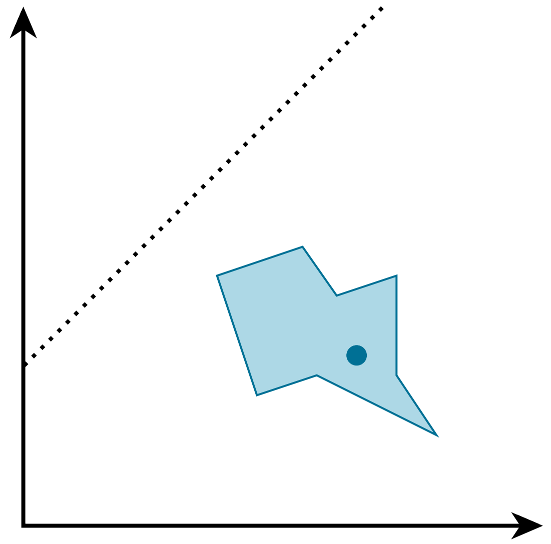
$$\begin{array}{ll} \underset{z,\hat{z}}{\text{minimize}} & (z_k)_{y^\star} - (z_k)_{y^{\text{target}}} \\ \text{subject to} & \|z_1 - x\|_\infty \le \epsilon \\ & \hat{z}_{i+1} = W_i z_i + b_i, \qquad i = 1, \ldots, k-1 \\ & z_i = \max\{\hat{z}_i, 0\}, \qquad i = 2, \ldots, k-1 \end{array}$$



Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

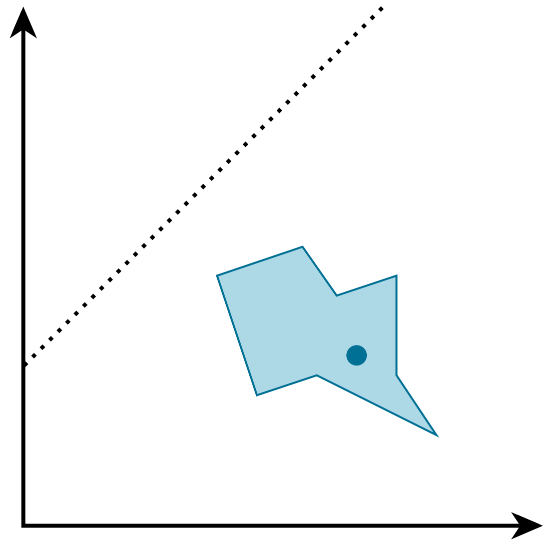
$$
\begin{array}{ll}
\underset{z,\hat{z}}{\text{minimize}} & (z_k)_{y^\star} - (z_k)_{y^{\text{target}}} \\
\text{subject to} & \|z_1 - x\|_\infty \le \epsilon \\
& \hat{z}_{i+1} = W_i z_i + b_i, \qquad i = 1, \ldots, k-1 \\
& z_i = \max\{\hat{z}_i, 0\}, \qquad i = 2, \ldots, k-1
\end{array}
$$



Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

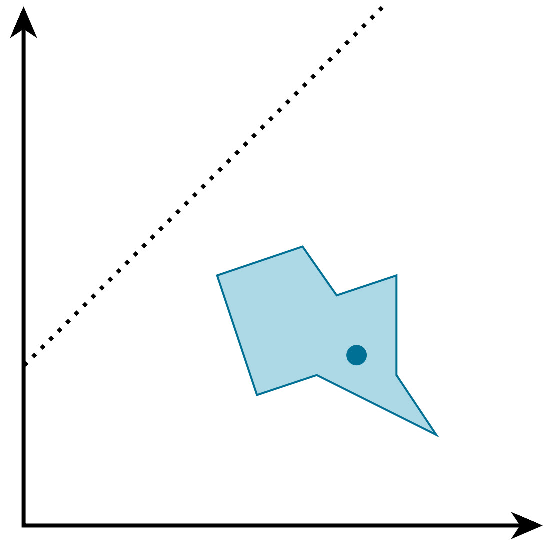
$$
\begin{array}{ll}
\underset{z,\hat{z}}{\text{minimize}} & (z_k)_{y^\star} - (z_k)_{y^{\text{target}}} \\
\text{subject to} & z_1 - x \leq \epsilon \\
& z_1 - x \geq -\epsilon \\
& \hat{z}_{i+1} = W_i z_i + b_i, \qquad i = 1, \ldots, k-1 \\
& z_i = \max\{\hat{z}_i, 0\}, \qquad i = 2, \ldots, k-1
\end{array}
$$



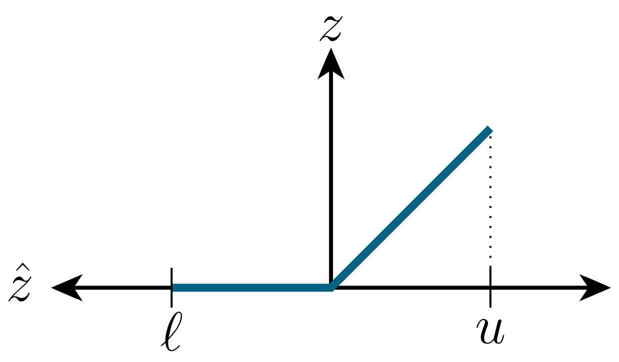
Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

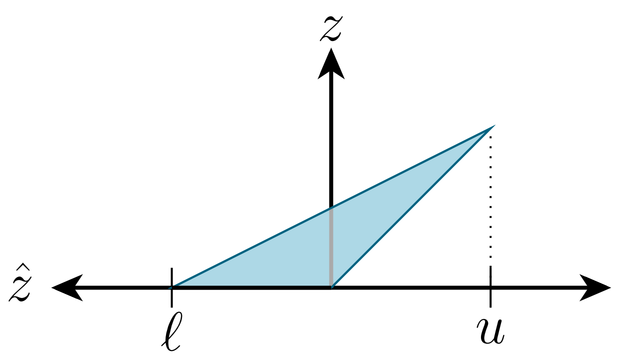
$$
\begin{aligned}
\underset{z,\hat{z}}{\text{minimize}} \quad & (z_k)_{y^\star} - (z_k)_{y^{\text{target}}} \\
\text{subject to} \quad & z_1 - x \le \epsilon \\
& z_1 - x \ge -\epsilon \\
& \hat{z}_{i+1} = W_i z_i + b_i, \qquad i = 1, \ldots, k-1 \\
& \color{red}{z_i = \max\{\hat{z}_i, 0\}, \qquad i = 2, \ldots, k-1}
\end{aligned}
$$



Idea #1: Convex bounds on ReLU nonlinearities



Bounded ReLU set



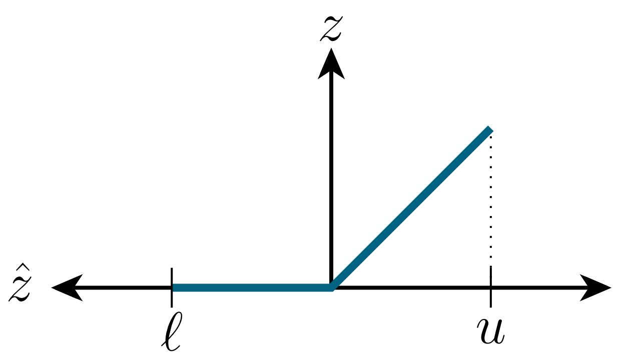
Convex relaxation

Suppose we have some upper and lower bound ℓ, u on the values that a particular (pre-ReLU) activation can take on, for this particular example x

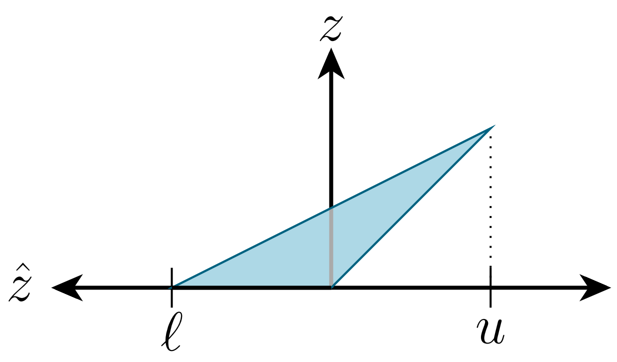
Then we can relax the ReLU "constraint" to its convex hull

$$
\begin{aligned}
\underset{z,\hat{z}}{\text{minimize}} \quad & (z_k)_{y^\star} - (z_k)_{y^{\text{target}}} \\
\text{subject to} \quad & z_1 - x \leq \epsilon \\
& z_1 - x \geq -\epsilon \\
& \hat{z}_{i+1} = W_i z_i + b_i, \qquad i = 1, \ldots, k-1 \\
& {\color{red} z_i = \max\{\hat{z}_i, 0\}, \qquad i = 2, \ldots, k-1}
\end{aligned}
$$

Idea #1: Convex bounds on ReLU nonlinearities



Bounded ReLU set



Convex relaxation

Suppose we have some upper and lower bound ℓ, u on the values that a particular (pre-ReLU) activation can take on, for this particular example x

Then we can relax the ReLU "constraint" to its convex hull

$$\begin{array}{ll}
\underset{z,\hat{z}}{\text{minimize}} & (z_k)_{y^\star} - (z_k)_{y^{\text{target}}} \\
\text{subject to} & z_1 - x \leq \epsilon \\
& z_1 - x \geq -\epsilon \\
& \hat{z}_{i+1} = W_i z_i + b_i, \qquad i = 1, \ldots, k-1 \\
& {\color{red}(\hat{z}_i, z_i) \in \mathcal{C}(\ell_i, u_i), \qquad i = 2, \ldots, k-1}
\end{array}$$

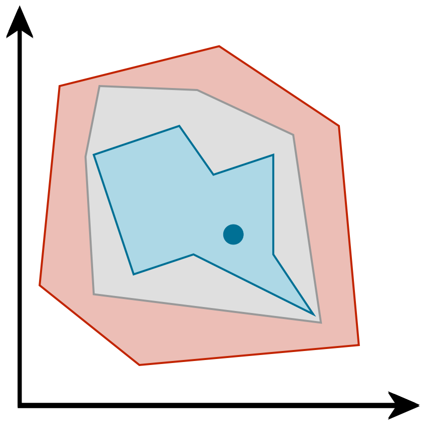
A linear program!

Idea #2: Exploiting duality

While the previous formulation is nice, it would require solving an LP (with the number of variables equal to the number of hidden units in network), once for each example, for each SGD step

- (This even ignores how to compute upper and lower bounds ℓ, u)

We're going to use the "duality trick", the fact that *any* feasible dual solution gives a lower bound on LP solution



🟦 True adversarial polytope

⬜ Convex outer bound (from ReLU convex hull)

🟥 Bound from dual feasible solution

An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution ***using a single backward pass through the network*** (really a slightly augmented form of the backprop network)

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

$$\Updownarrow$$

$$
\begin{aligned}
\underset{\nu,\alpha}{\text{maximize}} \quad & J_{\epsilon,W,b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in \mathcal{J}_i} \ell_{i,j} [\nu_{i,j}]_+ \\
\text{subject to} \quad & \nu_k = -c \\
& \hat{\nu}_i = W^T \nu_{i+1}, \qquad i = k-1, \ldots, 1 \\
& \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2
\end{aligned}
$$

An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution ***using a single backward pass through the network*** (really a slightly augmented form of the backprop network)

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

$$\Updownarrow$$

$$
\begin{aligned}
\underset{\nu,\alpha}{\text{maximize}} \quad & J_{\epsilon,W,b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in \mathcal{I}_i} \ell_{i,j} [\nu_{i,j}]_+ \\
\text{subject to} \quad & \nu_k = -c \\
& \hat{\nu}_i = W^T \nu_{i+1}, \qquad i = k-1, \ldots, 1 \\
& \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2
\end{aligned}
$$

Set of all activations in layer i that can cross zero

An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution ***using a single backward pass through the network*** (really a slightly augmented form of the backprop network)

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

$$\Updownarrow$$

$$\begin{aligned} \underset{\nu,\alpha}{\text{maximize}} \quad & J_{\epsilon,W,b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in \mathcal{J}_i} \ell_{i,j} [\nu_{i,j}]_+ \\ \text{subject to} \quad & \nu_k = -c \\ & \hat{\nu}_i = W^T \nu_{i+1}, \qquad i = k-1, \ldots, 1 \\ & \nu_i = \color{red}{f_i}(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2 \end{aligned}$$

Derivative of ReLU with slightly modification on $\mathcal{J}_i$

An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution ***using a single backward pass through the network*** (really a slightly augmented form of the backprop network)

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

$$\Updownarrow$$

$$\begin{aligned} \underset{\nu,\alpha}{\text{maximize}} \quad & J_{\epsilon,W,b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in \mathcal{I}_i} \ell_{i,j} [\nu_{i,j}]_+ \\ \text{subject to} \quad & \boxed{\begin{aligned} & \nu_k = -c \\ & \hat{\nu}_i = W^T \nu_{i+1}, \qquad i = k-1, \ldots, 1 \\ & \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2 \end{aligned}} \end{aligned}$$

Almost identical to backprop network

An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution ***using a single backward pass through the network*** (really a slightly augmented form of the backprop network)

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

$$\Updownarrow$$

$$\begin{aligned} \underset{\nu,\alpha}{\text{maximize}} \quad & J_{\epsilon,W,b}(\nu, x) \equiv \boxed{-\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1} - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in \mathcal{J}_i} \ell_{i,j} [\nu_{i,j}]_+ \\ \text{subject to} \quad & \nu_k = -c \\ & \hat{\nu}_i = W^T \nu_{i+1}, \qquad i = k-1, \ldots, 1 \\ & \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2 \end{aligned}$$

Objective at $\epsilon = 0$

An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution ***using a single backward pass through the network*** (really a slightly augmented form of the backprop network)

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

$$\Updownarrow$$

$$\begin{aligned} \underset{\nu,\alpha}{\text{maximize}} \quad & J_{\epsilon,W,b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 \boxed{- \epsilon \|\hat{\nu}_1\|_1} + \sum_{i=1}^{k-1} \sum_{j \in \mathcal{J}_i} \ell_{i,j} [\nu_{i,j}]_+ \\ \text{subject to} \quad & \nu_k = -c \\ & \hat{\nu}_i = W^T \nu_{i+1}, \qquad i = k-1, \ldots, 1 \\ & \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2 \end{aligned}$$

Robustness penalty (same form as in linear case)

An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution ***using a single backward pass through the network*** (really a slightly augmented form of the backprop network)

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

$$\Updownarrow$$

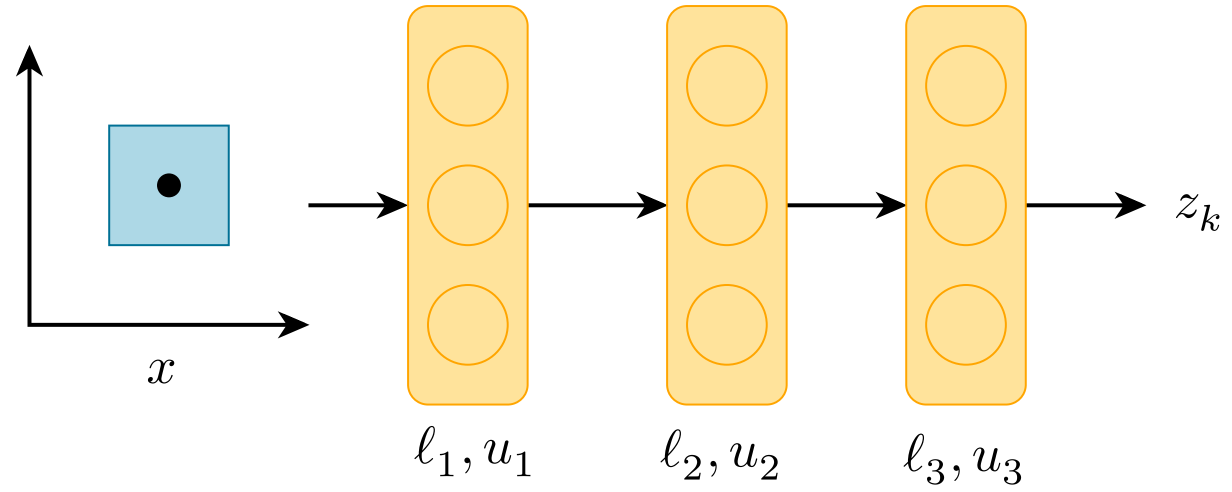
$$\begin{aligned} \underset{\nu,\alpha}{\text{maximize}} \quad & J_{\epsilon,W,b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \boxed{\sum_{i=1}^{k-1} \sum_{j \in \mathcal{J}_i} \ell_{i,j} [\nu_{i,j}]_+} \\ \text{subject to} \quad & \nu_k = -c \\ & \hat{\nu}_i = W^T \nu_{i+1}, \qquad i = k-1, \ldots, 1 \\ & \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \qquad i = k-1, \ldots, 2 \end{aligned}$$

Additional penalty for violating ReLU constraint

Idea #3: Iterative lower and upper bounds

A meaningful bound requires good lower and upper bounds ℓ_i, u_i

Incrementally build bounds by solving LP for each activation



Need some tricks to make this efficient: use same (particular) α for dual problems, compute multiplications in the right order in objective

Putting it all together

In the end, instead of minimizing the traditional loss…

$$\operatorname*{minimize}_{\theta} \sum_{i=1}^{m} \ell(h_\theta(x^{(i)}), y^{(i)})$$

…we just minimize a loss with a different network, involving a few forward and backward passes, and we get a *guaranteed* bound on worst-case loss (or error) for *any* norm-bounded adversarial attack

$$\operatorname*{minimize}_{\theta} \sum_{i=1}^{m} \ell(J_{\epsilon,\theta}(x^{(i)}), y^{(i)})$$

At test time, evaluate the bound to see if example is possibly adversarial (zero false negatives, but may incorrectly flag some benign examples)

Outline

Adverarial attacks on machine learning

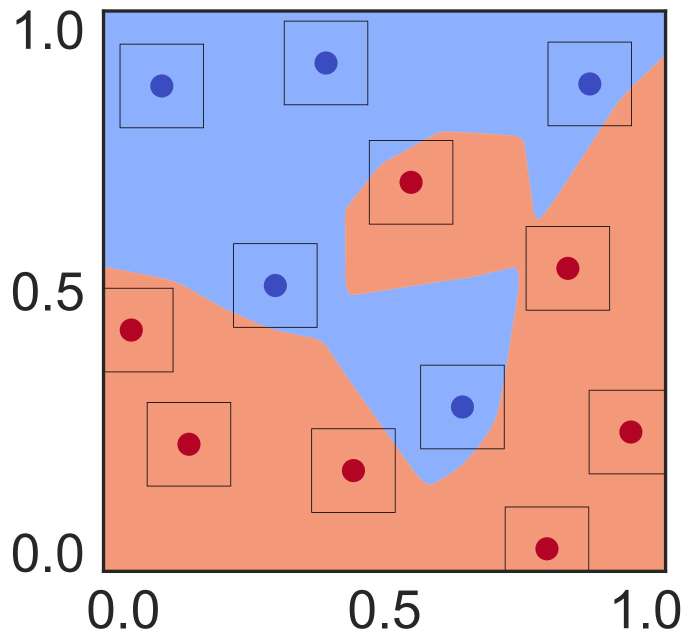
Robust optimization

Provable defenses for deep classifiers

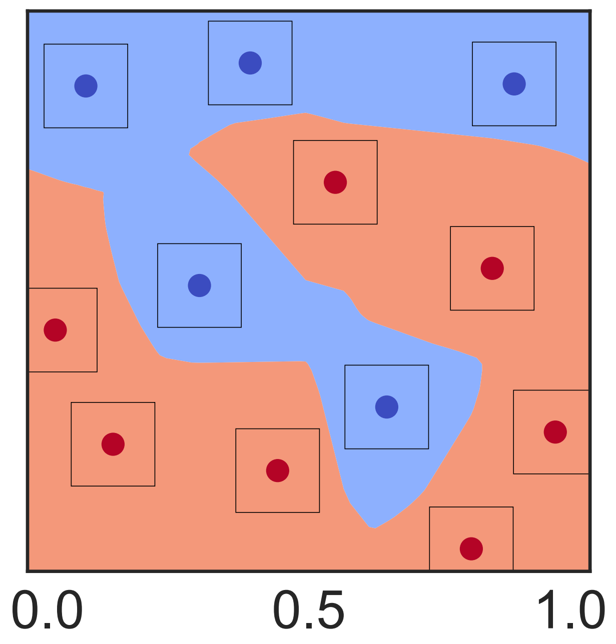
Experimental results

2D Toy Example

Simple 2D toy problem, 2-100-100-100-2 MLP network, trained with Adam (learning rate = 0.001, no real hyperparameter tuning)



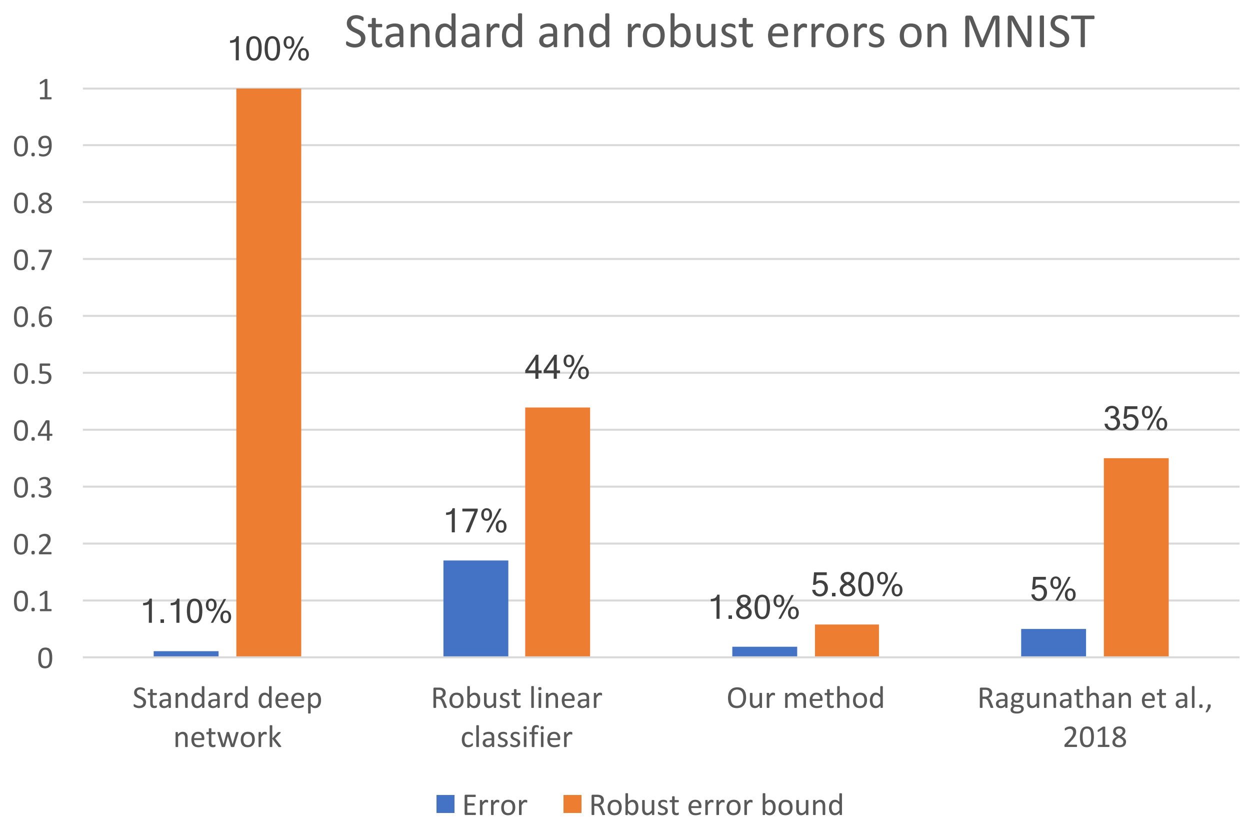
Standard training



Robust convex training

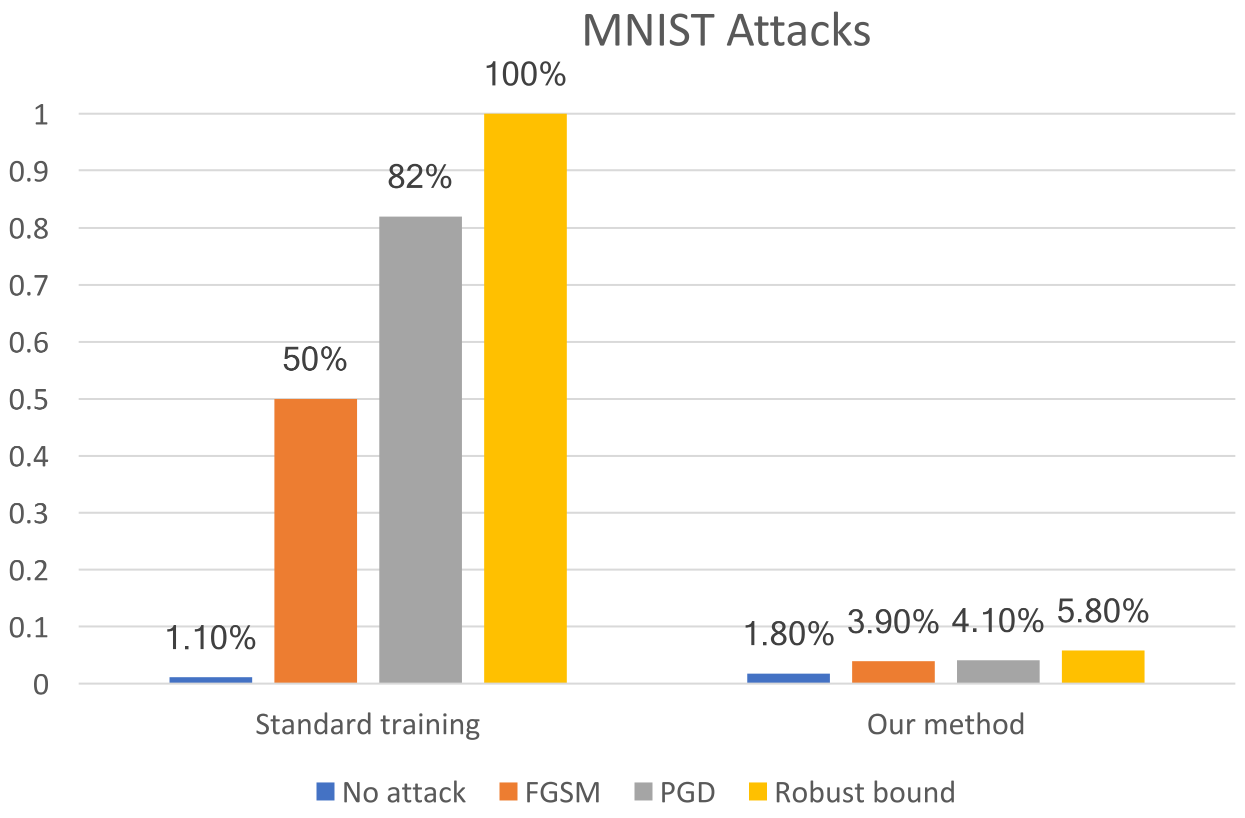
MNIST

Strided ConvNet (Conv16x4x4, Conv32x4x4, FC100, FC10) ReLUs following each layer, convolutions have stride=2



MNIST Attacks

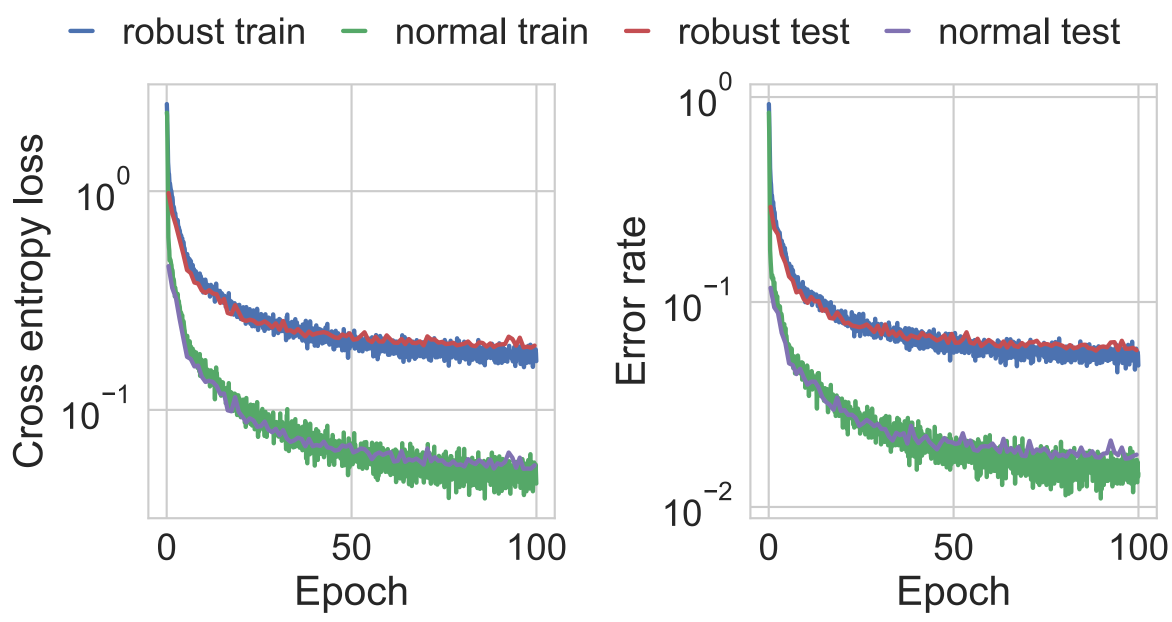
We can also look at how well real attacks perform at $\epsilon = 0.1$



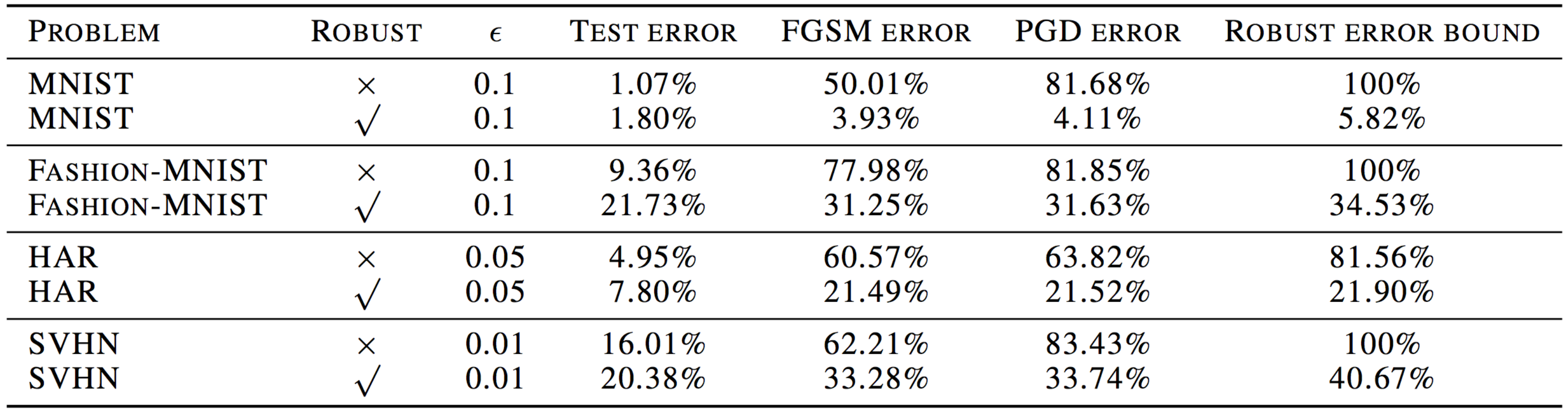
Convergence

Training does take substantially longer (2 hours), and requires more epochs than standard training

Method does largely avoid overfitting (adversarial robustness is a powerful regularizer), so we want to consider larger architectures



Results on additional tasks



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

Promising performance, but lots more work remains (right now, performance is limited by the size of architectures we can run), current work involves scaling to larger problems via random projections, bottleneck layers, and other techniques

Some take away messages

The work on adversarial defenses, up until now, has been extremely ad-hoc, defenses again *some* hypothesized attack, but not all attacks

Combining techniques from this class: convex optimization, linear programming, duality, with deep networks, is a largely unexplored and hugely fruitful area

Many open questions and practical challenges remain, but I think we are starting to be on the right course