Semidefinite relaxations for certifying robustness to adversarial examples

Jacob Steinhardt

Percy Liang

Aditi Raghunathan
ML: Powerful But Fragile
ML: Powerful But Fragile

- ML is successful on several tasks: object recognition, game playing, face recognition
ML: Powerful But Fragile

• ML is successful on several tasks: object recognition, game playing, face recognition

• ML systems fail catastrophically in presence of adversaries
ML: Powerful But Fragile

• ML is successful on several tasks: object recognition, game playing, face recognition

• ML systems fail catastrophically in presence of adversaries
ML: Powerful But Fragile

• ML is successful on several tasks: object recognition, game playing, face recognition

• ML systems fail catastrophically in presence of adversaries

• Different kinds of adversarial manipulations — data poisoning, manipulation of test inputs, model theft, membership inference etc.
ML: Powerful But Fragile

- ML is successful on several tasks: object recognition, game playing, face recognition

- ML systems fail catastrophically in presence of adversaries

- Different kinds of adversarial manipulations — data poisoning, manipulation of test inputs, model theft, membership inference etc.

- Focus on adversarial examples — manipulation of test inputs
Adversarial Examples
Adversarial Examples

Glasses \rightarrow Impersonation

[Sharif et al. 2016]
Adversarial Examples

Glasses $\rightarrow$ Impersonation
[Sharif et al. 2016]

Banana + 🐥 patch $\rightarrow$ Toaster
[Brown et al. 2017]
Adversarial Examples

- Glasses $\rightarrow$ Impersonation
  [Sharif et al. 2016]

- Banana + Apple patch $\rightarrow$ Toaster
  [Brown et al. 2017]

- Stop + Apple sticker $\rightarrow$ Yield
  [Evtimov et al. 2017]
Adversarial Examples
Adversarial Examples

3D Turtle → Rifle

[Athalye et al. 2017]
Adversarial Examples

 formatDate

![3D Turtle → Rifle](image)

[Athalye et al. 2017]

![Noise → “Ok Google”](image)

[Carlini et al. 2017]
Adversarial Examples

3D Turtle → Rifle
[Athalye et al. 2017]

Noise → “Ok Google”
[Carlini et al. 2017]

Malware → Benign
[Grosse et al. 2017]
What is an adversarial example?
What is an adversarial example?

Definition of attack model usually application specific and complex
What is an adversarial example?

Definition of attack model usually application specific and complex

We consider the well studied $\ell_\infty$ attack model.
What is an adversarial example?

Definition of attack model usually application specific and complex

We consider the well studied $\ell_\infty$ attack model

$p + .007 \times$ noise $= q$

Panda

Gibbon

Szegedy et al. 2014
Definition of attack model usually application specific and complex

We consider the well studied $\ell_\infty$ attack model

\[ |x_{\text{adv}} - x|_i \leq \epsilon \text{ for } i = 1, 2, \ldots, d \]

Panda + .007 × = Gibbon

Szegedy et al. 2014
What is an adversarial example?

Definition of attack model usually application specific and complex

We consider the well studied $\ell_\infty$ attack model

$|x_{\text{adv}} - x|_i \leq \epsilon$ for $i = 1, 2, \ldots d$

$x_{\text{adv}} \in B_\epsilon(x)$

Panda

Gibbon

Szegedy et al. 2014
History
History

Hard to defend even in this well defined model…
History

Hard to defend even in this well defined model...

- [Szegedy+ 2014]: First discover adversarial examples
History

Hard to defend even in this well defined model…

• [Szegedy+ 2014]: First discover adversarial examples

• [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
History

Hard to defend even in this well defined model…

• [Szegedy+ 2014]: First discover adversarial examples
• [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
• [Papernot+ 2015]: Defensive Distillation
History

Hard to defend even in this well defined model...

- [Szegedy+ 2014]: First discover adversarial examples
- [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
- [Papernot+ 2015]: Defensive Distillation
- [Carlini & Wagner 2016]: Distillation is not secure
History

Hard to defend even in this well defined model…

- [Szegedy+ 2014]: First discover adversarial examples
- [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
- [Papernot+ 2015]: Defensive Distillation
- [Carlini & Wagner 2016]: Distillation is not secure
- [Papernot + 2017]: Better distillation
History

Hard to defend even in this well defined model…

• [Szegedy+ 2014]: First discover adversarial examples
• [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
• [Papernot+ 2015]: Defensive Distillation
• [Carlini & Wagner 2016]: Distillation is not secure
• [Papernot + 2017]: Better distillation
• [Carlini & Wagner 2017]: Ten detection strategies fail
History

Hard to defend even in this well defined model…

- [Szegedy+ 2014]: First discover adversarial examples
- [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
- [Papernot+ 2015]: Defensive Distillation
- [Carlini & Wagner 2016]: Distillation is not secure
- [Papernot + 2017]: Better distillation
- [Carlini & Wagner 2017]: Ten detection strategies fail
- [Madry+ 2017]: AT against PGD, informal argument about optimality
History

Hard to defend even in this well defined model...

• [Szegedy+ 2014]: First discover adversarial examples
• [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
• [Papernot+ 2015]: Defensive Distillation
• [Carlini & Wagner 2016]: Distillation is not secure
• [Papernot + 2017]: Better distillation
• [Carlini & Wagner 2017]: Ten detection strategies fail
• [Madry+ 2017]: AT against PGD, informal argument about optimality
• [Lu + July 12 2017]: ”NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles”
History

Hard to defend even in this well defined model…

- [Szegedy+ 2014]: First discover adversarial examples
- [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
- [Papernot+ 2015]: Defensive Distillation
- [Carlini & Wagner 2016]: Distillation is not secure
- [Papernot + 2017]: Better distillation
- [Carlini & Wagner 2017]: Ten detection strategies fail
- [Madry+ 2017]: AT against PGD, informal argument about optimality
- [Lu + July 12 2017]: ”NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles”
- [Athalye and Sutskever July 17 2017]: Break above defense
History

Hard to defend even in this well defined model...

- [Szegedy+ 2014]: First discover adversarial examples
- [Goodfellow+ 2015]: Adversarial training (AT) against FGSM
- [Papernot+ 2015]: Defensive Distillation
- [Carlini & Wagner 2016]: Distillation is not secure
- [Papernot + 2017]: Better distillation
- [Carlini & Wagner 2017]: Ten detection strategies fail
- [Madry+ 2017]: AT against PGD, informal argument about optimality
- [Lu + July 12 2017]: ”NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles”
- [Athalye and Sutskever July 17 2017]: Break above defense
- [Athalye, Carlini, Wagner]: Break 6 out of 7 ICLR defenses
History

Hard to defend even in this well defined model…

• [Szegedy+ 2014]: First discover adversarial examples

• [Goodfellow+ 2015]: Adversarial training (AT) against FGSM

• [Papernot+ 2015]: Defensive Distillation

• [Carlini & Wagner 2016]: Distillation is not secure

• [Papernot + 2017]: Better distillation

• [Carlini & Wagner 2017]: Ten detection strategies fail

• [Madry+ 2017]: AT against PGD, informal argument about optimality

• [Lu + July 12 2017]: ”NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles”

• [Athalye and Sutskever July 17 2017]: Break above defense

• [Athalye, Carlini, Wagner]: Break 6 out of 7 ICLR defenses
Provable robustness
Provable robustness

Can we get robustness to all attacks?
Provable robustness

Can we get robustness to all attacks?

Let $f(\tilde{x})$ be the scoring function and adversary wants to maximize $f(\tilde{x})$
Provable robustness

Can we get robustness to all attacks?

Let \( f(\tilde{x}) \) be the scoring function and adversary wants to maximize \( f(\tilde{x}) \)

Contour lines of \( f(\tilde{x}) \) in \( B_\varepsilon(x) \)
Provable robustness

Can we get robustness to all attacks?

Let \( f(\tilde{x}) \) be the scoring function and adversary wants to maximize \( f(\tilde{x}) \)

Contour lines of \( f(\tilde{x}) \) in \( B_\epsilon(x) \)

**Attacks:** Generate points in \( B_\epsilon(x) \)
Provable robustness

Can we get robustness to all attacks?

Let \( f(\tilde{x}) \) be the scoring function and adversary wants to maximize \( f(\tilde{x}) \).

Contour lines of \( f(\tilde{x}) \) in \( B_\epsilon(x) \)

**Attacks:** Generate points in \( B_\epsilon(x) \)

\[ A_{fgsm}(x) = x + \epsilon \text{ sign}(\nabla f(x)) \]
Provable robustness

Can we get robustness to all attacks?

Let $f(\tilde{x})$ be the scoring function and adversary wants to maximize $f(\tilde{x})$

Contour lines of $f(\tilde{x})$ in $B_\epsilon(x)$

Attacks: Generate points in $B_\epsilon(x)$

$A_{fgsm}(x) = x + \epsilon \text{sign}(\nabla f(x))$

$A_{opt}(x) = \arg \max_{\tilde{x}} f(\tilde{x})$
Provable robustness

Can we get robustness to all attacks?

Let $f(\tilde{x})$ be the scoring function and adversary wants to maximize $f(\tilde{x})$

**Contour lines of $f(\tilde{x})$ in $B_\epsilon(x)$**

**Attacks**: Generate points in $B_\epsilon(x)$

$$A_{fgsm}(x) = x + \epsilon \text{ sign}(\nabla f(x))$$

$$A_{opt}(x) = \arg \max_{\tilde{x}} f(\tilde{x})$$

Network is provably robust if optimal attack fails
Provable robustness

Can we get robustness to all attacks?

Let $f(\tilde{x})$ be the scoring function and adversary wants to maximize $f(\tilde{x})$

Contour lines of $f(\tilde{x})$ in $B_\epsilon(x)$

Attacks: Generate points in $B_\epsilon(x)$

$$A_{fgsm}(x) = x + \epsilon \text{ sign}(\nabla f(x))$$

$$A_{opt}(x) = \arg \max_{\tilde{x}} f(\tilde{x})$$

Network is provably robust if optimal attack fails

$$f^* \equiv f(A_{opt}(x)) < 0$$
Provable robustness

Network is provably robust if \( f^* \equiv f(A_{\text{opt}}(x)) < 0 \)
Provable robustness

Network is provably robust if $f^* \equiv f(A_{opt}(x)) < 0$

Computing $f^*$ is intractable in general
Provable robustness

Network is provably robust if $f^* \equiv f(A_{\text{opt}}(x)) < 0$

Computing $f^*$ is intractable in general

- Combinatorial approaches to compute $f^*$
Provable robustness

Network is provably robust if \( f^* \equiv f(A_{\text{opt}}(x)) < 0 \)

Computing \( f^* \) is intractable in general

- **Combinatorial approaches** to compute \( f^* \)
  - SMT based Reluplex [Katz+ 2018]
Provable robustness

Network is provably robust if \( f^* \equiv f(A_{opt}(x)) < 0 \)
Computing \( f^* \) is intractable in general

- **Combinatorial approaches** to compute \( f^* \)
  - SMT based Reluplex [Katz+ 2018]
  - MILP based with specialized preprocessing [Tjeng+ 2018]
Provable robustness

Network is provably robust if \( f^* \equiv f(A_{opt}(x)) < 0 \)
Computing \( f^* \) is intractable in general

- **Combinatorial approaches** to compute \( f^* \)
  - SMT based Reluplex [Katz+ 2018]
  - MILP based with specialized preprocessing [Tjeng+ 2018]
- **Convex relaxations** to compute upper bound on \( f^* \)
Provable robustness

Network is provably robust if $f^* \equiv f(A_{opt}(x)) < 0$

Computing $f^*$ is intractable in general

- **Combinatorial approaches** to compute $f^*$
  - SMT based Reluplex [Katz+ 2018]
  - MILP based with specialized preprocessing [Tjeng+ 2018]
- **Convex relaxations** to compute upper bound on $f^*$
  - Upper bound is negative $\implies$ optimal attack fails
Provable robustness

Network is provably robust if \( f^* \equiv f(A_{\text{opt}}(x)) < 0 \)

Computing \( f^* \) is intractable in general

- **Combinatorial approaches** to compute \( f^* \)
  - SMT based Reluplex [Katz+ 2018]
  - MILP based with specialized preprocessing [Tjeng+ 2018]
- **Convex relaxations** to compute upper bound on \( f^* \)
  - Upper bound is negative \( \implies \) optimal attack fails
  - Computationally efficient upper bound
Provable robustness

Network is provably robust if $f^* = f(A_{\text{opt}}(x)) < 0$
Computing $f^*$ is intractable in general

- **Combinatorial approaches** to compute $f^*$
  - SMT based Reluplex [Katz+ 2018]
  - MILP based with specialized preprocessing [Tjeng+ 2018]
- **Convex relaxations** to compute upper bound on $f^*$
  - Upper bound is negative $\implies$ optimal attack fails
  - Computationally efficient upper bound

\[ 0 \quad f^* \quad f_{\text{upper}} \]

Not robust
Provable robustness

Network is provably robust if $f^* \equiv f(A_{\text{opt}}(x)) < 0$

Computing $f^*$ is intractable in general

- **Combinatorial approaches** to compute $f^*$
  - SMT based Reluplex [Katz+ 2018]
  - MILP based with specialized preprocessing [Tjeng+ 2018]
- **Convex relaxations** to compute upper bound on $f^*$
  - Upper bound is negative $\implies$ optimal attack fails
  - Computationally efficient upper bound

```
0   f^*   f_{\text{upper}}   f^*   f_{\text{upper}}   0
```

Not robust

Robust and certified
Provable robustness

Network is provably robust if \( f^* \equiv f(A_{\text{opt}}(x)) < 0 \)
Computing \( f^* \) is intractable in general

- **Combinatorial approaches** to compute \( f^* \)
  - SMT based Reluplex [Katz+ 2018]
  - MILP based with specialized preprocessing [Tjeng+ 2018]
- **Convex relaxations** to compute upper bound on \( f^* \)
  - Upper bound is negative \( \implies \) optimal attack fails
  - Computationally efficient upper bound

\[
\begin{array}{c|c|c}
0 & f^* & f^{\text{upper}} \\
\end{array}
\]
Not robust

\[
\begin{array}{c|c|c|c}
f^* & f^{\text{upper}} & 0 \\
\end{array}
\]
Robust and certified

\[
\begin{array}{c|c|c}
f^* & 0 & f^{\text{upper}} \\
\end{array}
\]
Robust and not certified
Two layer networks
Two layer networks

Contour lines of $f(\tilde{x})$ in $B_\epsilon(x)$
Two layer networks

Contour lines of $f(\tilde{x})$ in $B_\varepsilon(x)$

Gradient map of $f(\tilde{x})$ in $B_\varepsilon(x)$

\[ \nabla f(x) \]

\[ A_{f_{gsm}}(x) \]

\[ A_{opt}(x) \]

\[ \max \| \nabla f(\tilde{x}) \|_1 \]
Two layer networks

Key idea: Uniformly bound gradients
Two layer networks

Contour lines of $f(\tilde{x})$ in $B_\varepsilon(x)$

Gradient map of $f(\tilde{x})$ in $B_\varepsilon(x)$

Key idea: Uniformly bound gradients

$$f(\tilde{x}) \leq f(\bar{x}) + \varepsilon \max_{\tilde{x}} \|\nabla f(\tilde{x})\|_1$$
Two layer networks
Two layer networks

\[ f(x) = v^\top \sigma(Wx) \]
Two layer networks

Key idea: Uniformly bound gradients

\[ f(\tilde{x}) \leq f(x) + \epsilon \max_{\tilde{x}} \| \nabla f(\tilde{x}) \|_1 \]

\[ f(x) = v^\top \sigma(W x) \]
Two layer networks

Key idea: Uniformly bound gradients

\[ f(\tilde{x}) \leq f(\bar{x}) + \epsilon \max_{\tilde{x}} \| \nabla f(\tilde{x}) \|_1 \]

Bound on gradient:

\[ f(x) = v^\top \sigma(Wx) \]
Two layer networks

Key idea: Uniformly bound gradients

\[ f(\tilde{x}) \leq f(x) + \epsilon \max_{\tilde{x}} \| \nabla f(\tilde{x}) \|_1 \]

Bound on gradient:  
\[ \| \nabla f(\tilde{x}) \|_1 = \| W^\top \text{diag}(v) \sigma'(W\tilde{x}) \|_1 \]
Two layer networks

Key idea: Uniformly bound gradients

\[ f(\tilde{x}) \leq f(\bar{x}) + \epsilon \max_{\tilde{x}} \| \nabla f(\tilde{x}) \|_1 \]

Bound on gradient:

\[ \| \nabla f(\tilde{x}) \|_1 = \| W^\top \text{diag}(\nu) \sigma'(W\tilde{x}) \|_1 \]

\[ \leq \max_{s \in [0,1]^m, t \in [-1,1]^d} t^\top W^\top \text{diag}(\nu) s \]
Two layer networks

Key idea: Uniformly bound gradients

\[ f(\tilde{x}) \leq f(x) + \epsilon \max_{\tilde{x}} \| \nabla f(\tilde{x}) \|_1 \]

Bound on gradient:

\[ \| \nabla f(\tilde{x}) \|_1 = \| W^\top \text{diag}(v) \sigma'(W \tilde{x}) \|_1 \]

\[ \leq \max_{s \in [0,1]^m, t \in [-1,1]^d} t^\top W^\top \text{diag}(v) s \]

optimize over activations
Two layer networks

Key idea: Uniformly bound gradients

\[ f(\tilde{x}) \leq f(x) + \epsilon \max_{\tilde{x}} \| \nabla f(\tilde{x}) \|_1 \]

Bound on gradient:

\[ \| \nabla f(\tilde{x}) \|_1 = \| W^\top \text{diag}(v) \sigma'(W\tilde{x}) \|_1 \]

\[ \leq \max_{s \in [0,1]^m, t \in [-1,1]^d} t^\top W^\top \text{diag}(v) s \]

optimize over activations

optimize over signs of perturbation
Two layer networks

Key idea: Uniformly bound gradients

\[ f(\tilde{x}) \leq f(\bar{x}) + \epsilon \max_{\tilde{x}} \| \nabla f(\tilde{x}) \|_1 \]

Bound on gradient:

\[ \| \nabla f(\tilde{x}) \|_1 = \| W^\top \text{diag}(v) \sigma'(W \tilde{x}) \|_1 \]
\[ \leq \max_{s \in [0,1]^m, t \in [-1,1]^d} t^\top W^\top \text{diag}(v) s \]

optimize over activations
optimize over signs of perturbation

Final step: SDP relaxation (similar to MAXCUT) leads to Grad-cert
Relaxation → Training
Relaxation → Training

Training a neural network
Relaxation → Training

Training a neural network

Objective:
Objective: \[ \min_{W,v} \sum_{i=1}^{n} L(z_i, W, v) + \max_{P \geq 0, P_{ii} \leq 1} \text{tr}(M(W, v)P) \]

- Training loss
- Regularizer
Relaxation → Training

Training a neural network

Objective: \[
\min_{W,v} \sum_{i=1}^{n} L(z_i, W, v) + \max_{P \succeq 0, P_{ii} \leq 1} \text{tr}(M(W, v)P)
\]

Differentiable objective but expensive gradients
Relaxation ➔ Training

Training a neural network

Objective: \[
\min_{W,v} \sum_{i=1}^{n} L(z_i, W, v) + \max_{P \succeq 0, P_{ii} \leq 1} \text{tr}(M(W, v)P)
\]

Differentiable objective but expensive gradients

Duality to the rescue!
Training a neural network

Objective: \[
\min_{W,v} \sum_{i=1}^{n} L(z_i, W, v) + \max_{P \succeq 0, P_{ii} \leq 1} \text{tr}(M(W, v)P)
\]

Differentiable objective but expensive gradients

Duality to the rescue!

Regularizer: \[
D \cdot \lambda^+_{\text{max}} ((M(v, W) - \text{diag}(c)) + 1^\top \max(c, 0)
\]
Relaxation → Training

Training a neural network

Objective: $\min_{W,v} \sum_{i=1}^{n} L(z_i, W, v) + \max_{P \succeq 0, P_{ii} \leq 1} \text{tr}(M(W, v)P)$

\[ \text{training loss} \quad \text{regularizer} \]

Differentiable objective but expensive gradients

Duality to the rescue!

Regularizer: $D \cdot \lambda^+_{\max} \left( (M(v, W) - \text{diag}(c)) + 1^\top \max(c, 0) \right)$

Just one max eigenvalue computation for gradients
Results on MNIST
Results on MNIST

Attack: Projected Gradient Descent attack of Madry et al. 2018

Adversarial training: Minimizes this lower bound on training set
Results on MNIST

Attack: Projected Gradient Descent attack of Madry et al. 2018

Adversarial training: Minimizes this lower bound on training set
Results on MNIST
Results on MNIST

Attack: Projected Gradient Descent attack of Madry et al. 2018

Adversarial training: Minimizes this lower bound on training set
Results on MNIST

**Attack:** Projected Gradient Descent attack of Madry et al. 2018

**Adversarial training:** Minimizes this lower bound on training set

**Gradient based bound is quite loose**
Results on MNIST

Train with Grad-cert
(attack)
Results on MNIST

**Attack:** Projected Gradient Descent attack of Madry et al. 2018

**Our method:** Minimize gradient based upper bound
Results on MNIST
Results on MNIST

Attack: Projected Gradient Descent attack of Madry et al. 2018

Our method: Minimize gradient based upper bound
Results on MNIST

Attack: Projected Gradient Descent attack of Madry et al. 2018

Our method: Minimize gradient based upper bound
Results on MNIST
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight

Comparison with Wong and Kolter 2018 (LP-cert)
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight

Comparison with Wong and Kolter 2018 (LP-cert)

<table>
<thead>
<tr>
<th>Network</th>
<th>PGD-attack</th>
<th>LP-cert</th>
<th>Grad-cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-NN</td>
<td>22%</td>
<td>26%</td>
<td>93%</td>
</tr>
<tr>
<td>Grad-NN</td>
<td>15%</td>
<td>97%</td>
<td>35%</td>
</tr>
</tbody>
</table>
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight

Comparison with Wong and Kolter 2018 (LP-cert)

<table>
<thead>
<tr>
<th>Network</th>
<th>PGD-attack</th>
<th>LP-cert</th>
<th>Grad-cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-NN</td>
<td>22%</td>
<td>26%</td>
<td>93%</td>
</tr>
<tr>
<td>Grad-NN</td>
<td>15%</td>
<td>97%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Bounds are tight when you train
Results on MNIST
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight

Comparison with Wong and Kolter 2018 (LP-cert)

<table>
<thead>
<tr>
<th>Network</th>
<th>PGD-attack</th>
<th>LP-cert</th>
<th>Grad-cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-NN</td>
<td>22%</td>
<td>26%</td>
<td>93%</td>
</tr>
<tr>
<td>Grad-NN</td>
<td>15%</td>
<td>97%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Bounds are tight when you train
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight

Comparison with Wong and Kolter 2018 (LP-cert)

<table>
<thead>
<tr>
<th>Network</th>
<th>PGD-attack</th>
<th>LP-cert</th>
<th>Grad-cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-NN</td>
<td>22%</td>
<td>26%</td>
<td>93%</td>
</tr>
<tr>
<td>Grad-NN</td>
<td>15%</td>
<td>97%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Bounds are tight when you train

Bounds are tight **only** when you train
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight

Comparison with Wong and Kolter 2018 (LP-cert)

<table>
<thead>
<tr>
<th>Network</th>
<th>PGD-attack</th>
<th>LP-cert</th>
<th>Grad-cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-NN</td>
<td>22%</td>
<td>26%</td>
<td>93%</td>
</tr>
<tr>
<td>Grad-NN</td>
<td>15%</td>
<td>97%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Bounds are tight when you train

Bounds are tight only when you train

Some networks are empirically robust but not certified
(e.g. Adversarial Training of Madry et al. 2018)
Results on MNIST

Training a network to minimize gradient upper bound finds networks where the bound is tight

Comparison with Wong and Kolter 2018 (LP-cert)

<table>
<thead>
<tr>
<th>Network</th>
<th>PGD-attack</th>
<th>LP-cert</th>
<th>Grad-cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-NN</td>
<td>22%</td>
<td>26%</td>
<td>93%</td>
</tr>
<tr>
<td>Grad-NN</td>
<td>15%</td>
<td>97%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Bounds are tight when you train

Some networks are empirically robust but not certified (e.g. Adversarial Training of Madry et al. 2018)

Can we certify such “foreign” networks?
Summary so far...
Summary so far...

- **Certified robustness**: relaxed optimization to bound worst-case attack
Summary so far...

- **Certified robustness**: relaxed optimization to bound worst-case attack
- **Grad-cert**: Upper bound on worst case attack using uniform bound on gradient
Summary so far...

• **Certified robustness**: relaxed optimization to bound worst-case attack
• **Grad-cert**: Upper bound on worst case attack using uniform bound on gradient
• Training against the bound makes it tight
Summary so far...

- **Certified robustness**: relaxed optimization to bound worst-case attack
- **Grad-cert**: Upper bound on worst case attack using uniform bound on gradient
- Training against the bound makes it tight
- LP-cert and Grad-cert are tight only on training
Summary so far…

• **Certified robustness**: relaxed optimization to bound worst-case attack
• **Grad-cert**: Upper bound on worst case attack using uniform bound on gradient
• Training against the bound makes it tight
• LP-cert and Grad-cert are tight only on training
• **Goal**: Efficiently certify foreign multi-layer networks
New SDP-cert relaxation
New SDP-cert relaxation
New SDP-cert relaxation
New SDP-cert relaxation
New SDP-cert relaxation

Attack model constraints:

\[
\tilde{x} \quad x_1 \quad x_2 \quad x_3 \equiv x_L
\]
New SDP-cert relaxation

Attack model constraints:

$$|\bar{x} - \tilde{x}|_i \leq \epsilon$$

for $i = 1, 2, \ldots d$
New SDP-cert relaxation

Attack model constraints:

$$|\tilde{x} - \bar{x}|_i \leq \epsilon$$

for $i = 1, 2, \ldots d$

Neural net constraints

$$x_i = \text{ReLU}(W_{i-1}x_{i-1})$$

for $i = 1, 2, \ldots L$
New SDP-cert relaxation

Attack model constraints:

$$|\bar{x} - \tilde{x}|_i \leq \epsilon$$

for $$i = 1, 2, \ldots d$$

Neural net constraints

$$x_i = \text{ReLU}(W_{i-1}x_{i-1})$$

for $$i = 1, 2, \ldots L$$
New SDP-cert relaxation

**Neural net constraints**

\[ x_i = \text{ReLU}(W_{i-1} x_{i-1}) \quad \text{for } i = 1, 2, \ldots L \]

**Objective**

\[ \text{Objective} \]

**Attack model constraints:**

\[ |\bar{x} - \tilde{x}|_i \leq \epsilon \]

for \( i = 1, 2, \ldots d \)
New SDP-cert relaxation

Attack model constraints:
\[ |\bar{x} - \tilde{x}|_i \leq \epsilon \]
for \( i = 1, 2, \ldots, d \)

Neural net constraints
\[ x_i = \text{ReLU}(W_{i-1} x_{i-1}) \]
for \( i = 1, 2, \ldots, L \)

Objective
\[ f^* = \max_{\tilde{x}} (c_y - c_{\tilde{y}})^\top \tilde{x}_L \]
New SDP-cert relaxation

Source of non-convexity is the ReLU constraints

Attack model constraints:

\[ |\bar{x} - \tilde{x}|_i \leq \varepsilon \]

for \( i = 1, 2, \ldots d \)

Neural net constraints

\[ x_i = \text{ReLU}(W_{i-1}x_{i-1}) \]

for \( i = 1, 2, \ldots L \)

Objective

\[ f^* = \max_{\bar{x}} (c_y - c_{\bar{y}})^\top x_L \]
Handling ReLU constraints
Handling ReLU constraints

Consider single ReLU constraint \( z = \max(0, x) \)
Handling ReLU constraints

Consider single ReLU constraint  \( z = \max(0, x) \)

**Key insight:** Can be replaced by linear + quadratic constraints
Handling ReLU constraints

Consider single ReLU constraint $z = \max(0, x)$

Key insight: Can be replaced by linear + quadratic constraints

$$z \geq x \quad \text{Linear}$$
Handling ReLU constraints

Consider single ReLU constraint \( z = \max(0, x) \)

Key insight: Can be replaced by linear + quadratic constraints

\[
\begin{align*}
    z \geq x & \quad \text{Linear} \\
    z \geq 0 & \quad \text{Linear}
\end{align*}
\]
Handling ReLU constraints

Consider single ReLU constraint \( z = \max(0, x) \)

Key insight: Can be replaced by linear + quadratic constraints

\[
\{ 
\begin{align*}
    z & \geq x \quad \text{Linear} \\
    z & \geq 0 \quad \text{Linear}
\end{align*}
\]

\( z \) is greater than \( x, 0 \)
Handling ReLU constraints

Consider single ReLU constraint \( z = \max(0, x) \)

Key insight: Can be replaced by linear + quadratic constraints

\( z \) is greater than \( x, 0 \)

\[
\begin{aligned}
    & z \geq x & \text{Linear} \\
    & z \geq 0 & \text{Linear} \\
    & z(z - x) = 0 & \text{Quadratic}
\end{aligned}
\]
Handling ReLU constraints

Consider single ReLU constraint \( z = \max(0, x) \)

Key insight: Can be replaced by linear + quadratic constraints

\( z \) is greater than \( x, 0 \)
\[
\begin{aligned}
z & \geq x \quad \text{Linear} \\
z & \geq 0 \quad \text{Linear}
\end{aligned}
\]

\( z \) equal to one of \( x, 0 \)
\[
z(z - x) = 0 \quad \text{Quadratic}
\]
Handling ReLU constraints

Consider single ReLU constraint \( z = \max(0, x) \)

Key insight: Can be replaced by linear + quadratic constraints

\[
\begin{aligned}
z \text{ is greater than } x, 0 & \quad \begin{cases} 
z \geq x & \text{Linear} \\
z \geq 0 & \text{Linear} 
\end{cases} \\
z \text{ equal to one of } x, 0 & \quad z(z - x) = 0 \quad \text{Quadratic}
\end{aligned}
\]
Handling ReLU constraints

Consider **single** ReLU constraint \( z = \max(0, x) \)

**Key insight:** Can be replaced by linear + quadratic constraints

\[
\begin{align*}
\text{\( z \) is greater than \( x, 0 \)} & \quad \begin{cases} 
z \geq x & \text{Linear} \\
z \geq 0 & \text{Linear}
\end{cases} \\
\text{\( z \) equal to one of \( x, 0 \)} & \quad z(z - x) = 0 \quad \text{Quadratic}
\end{align*}
\]

Can relax quadratic constraints to get a semidefinite program
SDP relaxation
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv \text{Linear + Quadratic constraints}$
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv$ Linear + Quadratic constraints

$$M = \begin{bmatrix}
1 & x & z \\
x & x^2 & xz \\
z & xz & z^2 \\
\end{bmatrix}$$
SDP relaxation

Single ReLU constraint \( z = \max(0, x) \) \( \equiv \) Linear + Quadratic constraints

\[
M = \begin{bmatrix}
1 & x & z \\
x & x^2 & xz \\
z & xz & z^2 \\
\end{bmatrix}
\]

\( z \geq x \)
**SDP relaxation**

Single ReLU constraint $z = \max(0, x) \equiv \text{Linear + Quadratic constraints}$

$$M = \begin{bmatrix} 1 & x & z \\ x & x^2 & xz \\ z & xz & z^2 \end{bmatrix}$$

$z \geq x$
SDP relaxation

Single ReLU constraint \( z = \max(0, x) \equiv \) Linear + Quadratic constraints

\[
M = \begin{bmatrix}
1 & x & z \\
x & x^2 & xz \\
z & xz & z^2
\end{bmatrix}
\]

\( z \geq 0 \)
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv$ Linear + Quadratic constraints

$$M = \begin{bmatrix} 1 & x & z \\ x & x^2 & xz \\ z & xz & z^2 \end{bmatrix}$$

$z(z - x) = 0$

$z^2 = xz$
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv \text{Linear + Quadratic constraints}$

$$M = \begin{bmatrix} 1 & x & z \\ x & x^2 & xz \\ z & xz & z^2 \end{bmatrix}$$
**SDP relaxation**

Single ReLU constraint \( z = \max(0, x) \) \( \equiv \) Linear + Quadratic constraints

\[
M = \begin{bmatrix}
1 & x & z \\
x & x^2 & xz \\
z & xz & z^2
\end{bmatrix}
\]

ReLU constraints as linear constraints on matrix entries
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv$ Linear + Quadratic constraints

$$M = \begin{bmatrix} 1 & x & z \\ x & x^2 & xz \\ z & xz & z^2 \end{bmatrix}$$

ReLU constraints as linear constraints on matrix entries

Constraint on $M$
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv$ Linear + Quadratic constraints

$$M = \begin{bmatrix} 1 & x & z \\ x & x^2 & xz \\ z & xz & z^2 \end{bmatrix}$$

ReLU constraints as linear constraints on matrix entries

Constraint on $M$

$$M = vv^\top$$  Exact but non-convex set
SDP relaxation

Single ReLU constraint \( z = \max(0, x) \) \( \equiv \) Linear + Quadratic constraints

\[
M = \begin{bmatrix}
1 & x & z \\
x & x^2 & xz \\
z & xz & z^2
\end{bmatrix}
\]

ReLU constraints as linear constraints on matrix entries

Constraint on \( M \)

\[
M = vv^\top \quad \text{Exact but non-convex set}
\]

\[
M = VV^\top \quad \text{Relaxed and convex set}
\]
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv \text{Linear + Quadratic constraints}$

\[ M = \begin{bmatrix} 1 & x & z \\ x & x^2 & xz \\ z & xz & z^2 \end{bmatrix} \]

ReLU constraints as linear constraints on matrix entries

Constraint on M

\[ M = vv^\top \quad \text{Exact but non-convex set} \]
\[ M = VV^\top \quad \text{Relaxed and convex set} \]
SDP relaxation

Single ReLU constraint $z = \max(0, x) \equiv \text{Linear + Quadratic constraints}$

$$M = \begin{bmatrix} 1 & x & z \\ x & x^2 & xz \\ z & xz & z^2 \end{bmatrix}$$

ReLU constraints as linear constraints on matrix entries

Constraint on $M$

$$M = vv^\top \quad \text{Exact but non-convex set}$$

$$M = VV^\top \quad \text{Relaxed and convex set}$$

Generalizes to multiple layers: large matrix $M$ with all activations
SDP relaxation
SDP relaxation

Interaction between different hidden units
SDP relaxation

Interaction between different hidden units

\[ x_1, x_2 \in [-\epsilon, \epsilon] \]
SDP relaxation

Interaction between different hidden units

\[ x_1, x_2 \in [-\epsilon, \epsilon] \]
\[ z_1 = \text{ReLU}(x_1 + x_2) \]
\[ z_2 = \text{ReLU}(x_1 - x_2) \]
SDP relaxation

Interaction between different hidden units

\[ x_1, x_2 \in [-\epsilon, \epsilon] \]

\[ z_1 = \text{ReLU}(x_1 + x_2) \]

\[ z_2 = \text{ReLU}(x_1 - x_2) \]

\[ x_1 = x_2 = 0.5\epsilon \]
SDP relaxation

Interaction between different hidden units

\[ x_1, x_2 \in [-\epsilon, \epsilon] \]

\[ z_1 = \text{ReLU}(x_1 + x_2) \]

\[ z_2 = \text{ReLU}(x_1 - x_2) \]

\[ x_1 = x_2 = 0.5\epsilon \]
SDP relaxation

Interaction between different hidden units

\[ x_1, x_2 \in [-\epsilon, \epsilon] \]

\[ z_1 = \text{ReLU}(x_1 + x_2) \]

\[ z_2 = \text{ReLU}(x_1 - x_2) \]

LP treats units independently
SDP reasons jointly

Unrelaxed value

LP treats units independently
SDP reasons jointly
SDP relaxation

Interaction between different hidden units

\[ x_1, x_2 \in [-\epsilon, \epsilon] \]
\[ z_1 = \text{ReLU}(x_1 + x_2) \]
\[ z_2 = \text{ReLU}(x_1 - x_2) \]

LP treats units independently
SDP reasons jointly

Theorem: For a random two layer network with \( m \) hidden nodes and input dimension \( d \), \( \text{opt}(\text{LP}) = \Theta(md) \) and \( \text{opt}(\text{SDP}) = \Theta(m\sqrt{d} + d\sqrt{m}) \)
Results on MNIST
Results on MNIST

Three different robust networks
Results on MNIST

Three different robust networks

Grad-NN
[Raghunathan et al. 2018]
Results on MNIST

Three different robust networks

Grad-NN  
[Raghunathan et al. 2018]

LP-NN  
[Wong and Kolter 2018]
Results on MNIST

Three different robust networks

Grad-NN
[Raghunathan et al. 2018]

LP-NN
[Wong and Kolter 2018]

PGD-NN
[Madry et al. 2018]
## Results on MNIST

Three different robust networks

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Grad-cert</td>
<td>35%</td>
<td>93%</td>
<td>N/A</td>
</tr>
<tr>
<td>LP-cert</td>
<td>97%</td>
<td>22%</td>
<td>100%</td>
</tr>
<tr>
<td>SDP-cert</td>
<td>20%</td>
<td>20%</td>
<td>18%</td>
</tr>
<tr>
<td>PGD-attack</td>
<td>15%</td>
<td>18%</td>
<td>9%</td>
</tr>
</tbody>
</table>
## Results on MNIST

### Three different robust networks

<table>
<thead>
<tr>
<th></th>
<th>Grad-NN</th>
<th>LP-NN</th>
<th>PGD-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grad-cert</strong></td>
<td>35%</td>
<td>93%</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>LP-cert</strong></td>
<td>97%</td>
<td>22%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>SDP-cert</strong></td>
<td>20%</td>
<td>20%</td>
<td>18%</td>
</tr>
<tr>
<td><strong>PGD-attack</strong></td>
<td>15%</td>
<td>18%</td>
<td>9%</td>
</tr>
</tbody>
</table>

SDP provides good certificates on all three different networks
Results on MNIST
Results on MNIST

PGD-NN
[Madry et al. 2018]
Results on MNIST

PGD-NN

[Madry et al. 2018]
Results on MNIST

PGD-NN
[Madry et al. 2018]

Uncertified points are more vulnerable to attack
Scaling up…
Scaling up...

In general, CNNs are more robust than fully connected networks
Scaling up…

In general, CNNs are more robust than fully connected networks.

Off-the-shelf SDP solvers do not exploit the CNN structure.
Scaling up…

In general, CNNs are more robust than fully connected networks.

Off-the-shelf SDP solvers do not exploit the CNN structure.

Ongoing work:
Scaling up...

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

Ongoing work:
Scaling up…

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

Ongoing work:

First order matrix-vector product based SDP solvers
Scaling up…

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

Ongoing work:

First order matrix-vector product based SDP solvers
Scaling up...

In general, CNNs are more robust than fully connected networks.

Off-the-shelf SDP solvers do not exploit the CNN structure.

**Ongoing work:**

- First order matrix-vector product based SDP solvers
- Exploit efficient CNN implementations in Tensorflow
Scaling up…

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

Ongoing work:

First order matrix-vector product based SDP solvers

Exploit efficient CNN implementations in Tensorflow

Concurrent work:
Scaling up…

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

**Ongoing work:**

First order matrix-vector product based SDP solvers

Exploit efficient CNN implementations in Tensorflow

**Concurrent work:**
Scaling up…

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

**Ongoing work:**

First order matrix-vector product based SDP solvers

Exploit efficient CNN implementations in Tensorflow

**Concurrent work:**

MILP solving with efficient preprocessing [Tjeng+ 2018, Xiao+ 2018]
Scaling up…

In general, CNNs are more robust than fully connected networks.

Off-the-shelf SDP solvers do not exploit the CNN structure.

**Ongoing work:**

First order matrix-vector product based SDP solvers

Exploit efficient CNN implementations in Tensorflow

**Concurrent work:**

MILP solving with efficient preprocessing [Tjeng+ 2018, Xiao+ 2018]
Scaling up…

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

**Ongoing work:**

First order matrix-vector product based SDP solvers

Exploit efficient CNN implementations in Tensorflow

**Concurrent work:**

MILP solving with efficient preprocessing [Tjeng+ 2018, Xiao+ 2018]

Scaling up LP based methods [Dvijotham+ 2018, Wong and Kolter 2018]
Scaling up…

In general, CNNs are more robust than fully connected networks

Off-the-shelf SDP solvers do not exploit the CNN structure

Ongoing work:

First order matrix-vector product based SDP solvers

Exploit efficient CNN implementations in Tensorflow

Concurrent work:

MILP solving with efficient preprocessing [Tjeng+ 2018, Xiao+ 2018]

Scaling up LP based methods [Dvijotham+ 2018, Wong and Kolter 2018]
Summary
Summary

• Robustness for $\ell_\infty$ attack model
Summary

• Robustness for $\ell_\infty$ attack model
  • Certified evaluation to avoid arms race
Summary

• Robustness for \( \ell_\infty \) attack model
  • Certified evaluation to avoid arms race
  • Presented two different relaxations for certification
Summary

• Robustness for $\ell_\infty$ attack model
  • Certified evaluation to avoid arms race
  • Presented two different relaxations for certification
• Adversarial examples more broadly..
Summary

- Robustness for $\ell_\infty$ attack model
  - Certified evaluation to avoid arms race
  - Presented two different relaxations for certification
- Adversarial examples more broadly...
  - Does there exist a mathematically well defined attack model?
Summary

• Robustness for $\ell_\infty$ attack model
  • Certified evaluation to avoid arms race
  • Presented two different relaxations for certification

• Adversarial examples more broadly
  • Does there exist a mathematically well defined attack model?
  • Would the current techniques (deep learning + appropriate regularization) transfer to this attack model?
Summary

• Robustness for $\ell_{\infty}$ attack model
  • Certified evaluation to avoid arms race
  • Presented two different relaxations for certification
• Adversarial examples more broadly..
  • Does there exist a mathematically well defined attack model?
  • Would the current techniques (deep learning + appropriate regularization) transfer to this attack model?
• Secure vs. better models?
Summary

• Robustness for $\ell_\infty$ attack model
  • Certified evaluation to avoid arms race
  • Presented two different relaxations for certification
• Adversarial examples more broadly..
  • Does there exist a mathematically well defined attack model?
  • Would the current techniques (deep learning + appropriate regularization) transfer to this attack model?
• Secure vs. better models?
  • Adversarial examples expose limitations of current systems
Summary

• Robustness for $\ell_\infty$ attack model
  • Certified evaluation to avoid arms race
  • Presented two different relaxations for certification
• Adversarial examples more broadly..
  • Does there exist a mathematically well defined attack model?
  • Would the current techniques (deep learning + appropriate regularization) transfer to this attack model?
• Secure vs. better models?
  • Adversarial examples expose limitations of current systems
  • How do we get models to learn “the right thing”?
Thank you!

Jacob Steinhardt

Percy Liang

“Certified Defenses against Adversarial Examples”

“Semidefinite Relaxations for Certifying Robustness to Adversarial Examples”