Differentiable Abstract Interpretation for Provably Robust Neural Networks

safeai.ethz.ch

Matthew Mirman  Timon Gehr  Martin Vechev

ICML 2018
Adversarial Attack

Example of FGSM attack produced by Goodfellow et al. (2014)
Many developed attacks: Goodfellow et al. (2014); Madry et al. (2018); Evtimov et al. (2017); Athalye & Sutskever (2017); Papernot et al. (2017); Xiao et al. (2018); Carlini & Wagner (2017); Yuan et al. (2017); Tramèr et al. (2017)

$$L_\infty \text{ Adversarial Ball}$$

$$\text{Ball}_\epsilon(input) = \{attack \mid \|input - attack\|_\infty \leq \epsilon\}$$
\[ L_\infty \text{ Adversarial Ball} \]

Many developed attacks: Goodfellow et al. (2014); Madry et al. (2018); Evtimov et al. (2017); Athalye & Sutskever (2017); Papernot et al. (2017); Xiao et al. (2018); Carlini & Wagner (2017); Yuan et al. (2017); Tramèr et al. (2017)

\[
\text{Ball}_\epsilon(\text{input}) = \{ \text{attack} | \| \text{input} - \text{attack} \|_\infty \leq \epsilon \}
\]

A net is \( \epsilon \)-robust at \( x \) if it classifies every example in \( \text{Ball}_\epsilon(x) \) the same and correctly
Adversarial Ball

Is attack ∈ Ball_\epsilon(panda)?

<table>
<thead>
<tr>
<th>∈</th>
<th>∈</th>
<th>∉</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>∈</td>
<td>∉</td>
</tr>
<tr>
<td>0.5</td>
<td>∈</td>
<td>∈</td>
</tr>
</tbody>
</table>
Prior Work

Increase Network Robustness

Defense: Train a network so that *most* inputs are *mostly* robust.

- Madry et al. (2018); Tramèr et al. (2017); Cisse et al. (2017); Yuan et al. (2017); Gu & Rigazio (2014)

- Network still attackable
Prior Work

**Increase Network Robustness**

*Defense:* Train a network so that *most* inputs are *mostly* robust.

- Madry et al. (2018); Tramèr et al. (2017); Cisse et al. (2017); Yuan et al. (2017); Gu & Rigazio (2014)
- Network still attackable

**Certify Robustness**

*Verification:* Prove that a network is $\epsilon$-robust at a point

- Huang et al. (2017); Pei et al. (2017); Katz et al. (2017); Gehr et al. (2018)
- Experimentally robust nets not very *certifiably* robust
- Intuition: not all correct programs are provable
Train a Network to be *Certifiably Robust*\(^1\)

**Given:**
- \(\text{Net}_\theta\) with weights \(\theta\)
- Training inputs and labels

**Find:**
- \(\theta\) that maximizes number of inputs we can *certify* are \(\epsilon\)-robust

---

\(^1\)Also addressed by: Raghunathan et al. (2018); Kolter & Wong (2017); Dvijotham et al. (2018)
Train a Network to be *Certifiably Robust*\(^1\)

*Given:*
- \(\text{Net}_\theta\) with weights \(\theta\)
- Training inputs and labels

*Find:*
- \(\theta\) that maximizes number of inputs we can *certify* are \(\epsilon\)-robust

*Challenge*
- At least as hard as standard training!

\(^1\)Also addressed by: Raghunathan et al. (2018); Kolter & Wong (2017); Dvijotham et al. (2018)
High Level

Make certification the training goal

- Abstract Interpretation: certify by over-approximating output \(^2\)

\(^2\)Cousot & Cousot (1977); Gehr et al. (2018)

Image Credit: Petar Tsankov
Make certification the training goal

- Abstract Interpretation: certify by over-approximating output \(^2\)

- Use Automatic Differentiation on Abstract Interpretation

---

\(^2\) Cousot & Cousot (1977); Gehr et al. (2018)
Image Credit: Petar Tsankov
Abstract Interpretation
Cousot & Cousot (1977)

Abstract Interpretation is heavily used in industrial large-scale program analysis to compute over-approximation of program behaviors.

---


[4] \( f[\gamma(d)] \subseteq \gamma(f^\#(d)) \) where \( f[s] \) is the image of \( s \) under \( f \)
Abstract Interpretation
Cousot & Cousot (1977)

Abstract Interpretation is heavily used in industrial large-scale program analysis to compute over-approximation of program behaviors.

Provide

- abstract domain \( \mathcal{D} \) of abstract points \( d \)
- concretization function \( \gamma : \mathcal{D} \rightarrow \mathcal{P}(\mathbb{R}^n) \)
- concrete function \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n \)

Develop a sound\(^4\) abstract transformer \( f^\# : \mathcal{D} \rightarrow \mathcal{D} \)

\(^3\)For example by Astrée: Blanchet et al. (2003)

\(^4\)\( f[\gamma(d)] \subseteq \gamma(f^\#(d)) \) where \( f[s] \) is the image of \( s \) under \( f \)
Abstract Interpretation
Cousot & Cousot (1977)

Abstract Interpretation is heavily used in industrial large-scale program analysis to compute over-approximation of program behaviors.

Provide

- abstract domain $\mathcal{D}$ of abstract points $d$
- concretization function $\gamma : \mathcal{D} \rightarrow \mathcal{P}(\mathbb{R}^n)$
- concrete function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$

Develop a sound abstract transformer $f^# : \mathcal{D} \rightarrow \mathcal{D}$

- ReLU : $\mathbb{R}^n \rightarrow \mathbb{R}^n$ becomes $\text{ReLU}^# : \mathcal{D} \rightarrow \mathcal{D}$

---


$[^4]$ $f[\gamma(d)] \subseteq \gamma(f^#(d))$ where $f[s]$ is the image of $s$ under $f$
Abstract Optimization Goal

Given

- $\text{mx}(d)$: a way to compute upper bounds for $\gamma(d)$.
- $\text{ball}(x) \in \mathcal{D}$: a ball abstraction s.t. $\text{Ball}_\varepsilon(x) \subseteq \gamma(\text{ball}(x))$
- $\text{Loss}_t$: an abstractable traditional loss function for classification target $t$

$$
\text{Err}_{t,\text{Net}}(x) = \text{Loss}_t \circ \text{Net}(x) \quad \text{classical error}
$$
$$
\text{AbsErr}_{t,\text{Net}}(x) = \text{mx} \circ \text{Loss}_t^\# \circ \text{Net}^\# \circ \text{ball}(x) \quad \text{abstract error}
$$
Using Abstract Goal

Theorem
$\text{Err}_{t, \text{Net}}(y) \leq \text{AbsErr}_{t, \text{Net}}(x)$ for all points $y \in \text{Ball}_\epsilon(x)$
Abstract Domains

- Many abstract domains $\mathcal{D}$ with different speed/accuracy tradeoffs
- Transformers must be parallelizable, and work well with SGD
Abstract Domains

- Many abstract domains $\mathcal{D}$ with different speed/accuracy tradeoffs
- Transformers must be parallelizable, and work well with SGD

Box Domain

- $p$ dimension axis-aligned boxes
- Ball$_{\epsilon}$: perfect
- $(\cdot M)^\#: \text{uses abs}$
- ReLU$^\#: 6 \text{ linear operations, 2 ReLUs}$
Abstract Domains

- Many abstract domains $\mathcal{D}$ with different speed/accuracy tradeoffs
- Transformers must be parallelizable, and work well with SGD

**Box Domain**
- $p$ dimension axis-aligned boxes
- Ball$_{\epsilon}$: perfect
- $(\cdot M)^\#$: uses abs
- ReLU$^\#$: 6 linear operations, 2 ReLUs

**Zonotope Domain**
- Affine transform of $k$-cube onto $p$ dims
- $k$ increases with non-linear transformers
- Ball$_{\epsilon}$: perfect
- $(\cdot M)^\#$: perfect
- ReLU$^\#$: zBox, zDiag, zSwitch, zSmooth,
- Hybrid: hSwitch, hSmooth
Implementation
DiffAI Framework

- Can be found at: safeai.ethz.ch
- Implemented in PyTorch\(^5\)
- Tested with modern GPUs

\(^5\)Paszke et al. (2017)
Scalability

CIFAR10

<table>
<thead>
<tr>
<th>Model</th>
<th>#Neurons</th>
<th>#Weights</th>
<th>Train 1 Epoch (s)</th>
<th>Test 2k Pts (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvSuper</td>
<td>∼124k</td>
<td>∼16mill</td>
<td>Base 23, Attack(^6) 149, Box 74</td>
<td>Box 0.09, hSwitch 40</td>
</tr>
</tbody>
</table>

- Can use a less precise domain for training than for certification
- Can test/train Resnet18\(^8\): 2k points tested on ∼500k neurons in ∼1s with Box
- tldr: can test and train with larger nets than prior work

\(^6\) 5 iterations of PGD Madry et al. (2018) for both training and testing
\(^7\) ConvSuper: 5 layers deep, no Maxpool.
\(^8\) like that described by He et al. (2016) but without pooling or dropout.
Robustness Provability

MNIST with $\epsilon = 0.1$ on ConvSuper

<table>
<thead>
<tr>
<th>Training Method</th>
<th>%Correct</th>
<th>%Attack Success</th>
<th>%hSwitch Certified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>98.4</td>
<td>2.4</td>
<td>2.8</td>
</tr>
<tr>
<td>Madry et al. (2018)</td>
<td>98.8</td>
<td>1.6</td>
<td>11.2</td>
</tr>
<tr>
<td>Box</td>
<td>99.0</td>
<td>2.8</td>
<td>96.4</td>
</tr>
</tbody>
</table>

- Usually loses only small amount of accuracy (sometimes gains)
- Significantly increases provability\(^9\)

\(^9\)Much more thorough evaluation in appendix of Mirman et al. (2018).
**hSmooth Training**

FashionMNIST with $\epsilon = 0.1$ on FFNN

<table>
<thead>
<tr>
<th>Method</th>
<th>Train Total (s)</th>
<th>%Correct</th>
<th>%zSwitch</th>
<th>Certified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>119</td>
<td>94.6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Box</td>
<td>608</td>
<td>8.6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hSmooth</td>
<td>4316</td>
<td>84.4</td>
<td>21.0</td>
<td></td>
</tr>
</tbody>
</table>

- Training unexpectedly fails with Box (very rare)
- Training slow but reliable with hSmooth
Conclusion

First application of automatic differentiation to abstract interpretation (that we know of)

Trained and verified the largest verifiable neural networks to date

A way to train networks on regions, not just points\(^\text{10}\)

\(^{10}\)Further examples of this use-case in paper


Box Domain

- Interval for each of the $p$ nodes in network graph
- Represented by center $c \in \mathbb{R}^P$ and radius $b \in \mathbb{R}_+^P$

- Concretization$^{11}$:
  \[ \gamma_I(\langle c, b \rangle) = \{ c + b \odot \beta \mid \beta \in [-1, 1]^p \} \]

- Constant matrix multiply transformer$^{12}$:
  \[ (\cdot M)^#(\langle c, b \rangle) = \langle c \cdot M, b \cdot \text{abs}(M) \rangle \]

- ReLU$^\#$: 6 linear operations, 2 ReLUs

---

$^{11}\odot$ is pointwise multiply

$^{12}p = m \times n$ and $M \in \mathbb{R}^{n \times w}$
Zonotope Domain
Goubault & Putot (2006)

- Affine transform of $k$-dimensional unit-cube onto the $p$ network graph nodes
- Represented by center $c \in \mathbb{R}^{p \times 1}$ and $k$ error terms $r \in \mathbb{R}^{p \times k}$

- Concretization:
  \[ \gamma_Z(\langle c, r \rangle) = \{ c + re | e \in [-1, 1]^{k \times 1} \} \]

- Constant matrix multiply transformer\textsuperscript{13}:
  \[ (\cdot M)^\#(\langle c, r \rangle) = \langle c \ast M, r \ast M \rangle \]

- ReLU\textsuperscript{#}: zBox, zDiag, zSwitch, zSmooth

\textsuperscript{13}for $p = m \times n$ and $M \in \mathbb{R}^{n \times w}$ and $\ast$ is batched matrix multiply

Zonotope Image uploaded to Wikipedia by user Tomruen and licensed under CC
Zonotope Domain

SGD Suitable ReLU Transformers

- zBox: Treat as Box when surrounding zero
- zDiag: Add possible error when surrounding zero

Three examples of zBox (blue) and zDiag (red), with in \((i)\) visualized on X and out on Y axis. Dashed line is \(ReLU(in)\)

- zSwitch: Choose between zBox and zDiag to use based on volume heuristic
- zSmooth: Linear combination of zBox and zDiag based on volume heuristic
Hybrid Zonotope

- Zonotope ReLU transformers all introduce a new error terms for every node
- **Hybrid Zonotope**: minkowski sum of a $p$-box with $k$-zonotope
- $k$ fixed to be number of pixels
- ReLU#: hSwitch, hSmooth
### Prior Results

<table>
<thead>
<tr>
<th>System</th>
<th>Model</th>
<th>#Neurons</th>
<th>#Weights</th>
<th>Train 1 Epoch (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiffAI</td>
<td>ConvSuper</td>
<td>~124k</td>
<td>~16mill</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>Resnet18</td>
<td>~500k</td>
<td>~15mill</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>ConvHuge</td>
<td>~500k</td>
<td>~65mill</td>
<td>142</td>
</tr>
<tr>
<td>Wong et al. (2018)</td>
<td>Large</td>
<td>~62k</td>
<td>~2.5mill</td>
<td>466</td>
</tr>
<tr>
<td></td>
<td>Resnet</td>
<td>~107k</td>
<td>~4.2mill</td>
<td>1685</td>
</tr>
<tr>
<td>Wong &amp; Kolter (2018)</td>
<td>MNIST Conv</td>
<td>~4k</td>
<td>~10k</td>
<td>180</td>
</tr>
<tr>
<td>Raghunathan et al. (2018)</td>
<td>MNIST 2 layer FFNN</td>
<td>~1k</td>
<td>~650k</td>
<td>-</td>
</tr>
<tr>
<td>Dvijotham et al. (2018)</td>
<td>Convnets</td>
<td>~21k</td>
<td>~650k</td>
<td>-</td>
</tr>
</tbody>
</table>

- Numbers as reported by prior work and not rerun on our hardware
- When hidden unit numbers and weight numbers were included, they were approximated using the network specifications in the paper with over-approximations where the specifications were not complete as in Dvijotham et al. (2018); Raghunathan et al. (2018)
Ongoing Work

- More provability for deeper networks
- Sound testing w/ respect to floating point
- Inferring maximal provability $\epsilon$