NATTACK
by learning the
Distributions of Adversarial Examples

Boqing Gong
Joint work with Yandong Li, Lijun Li, Liqiang Wang, & Tong Zhang
Published in ICML 2019
Intriguing properties of deep neural networks (DNNs)

“panda”
57.7% confidence

+.007 ×

“gibbon”
99.3% confidence


Intriguing properties of deep neural networks (DNNs)

\[ +\eta\text{sign}(\nabla L(x_t, y)) = \]

“panda” 57.7% confidence

“gibbon” 99.3% confidence


Projected gradient descent (PGD) attack

\[ x_{t+1} \leftarrow \text{Proj}_S(x_t + \eta \text{sign}(\nabla L(x_t, y))) \]

Intriguing results (1)

~100% attack success rates on CIFAR10 & ImageNet
Intriguing results (2)

<table>
<thead>
<tr>
<th></th>
<th>ADV-TRAIN</th>
<th>ADV-BNN</th>
<th>THERM-ADV</th>
<th>LID</th>
<th>THERM</th>
<th>SAP</th>
<th>VANILLA-1</th>
<th>VANILLA-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV-TRAIN</td>
<td>100.00</td>
<td>33.58</td>
<td>44.07</td>
<td>36.88</td>
<td>34.60</td>
<td>28.49</td>
<td>36.00</td>
<td>22.80</td>
</tr>
<tr>
<td>ADV-BNN</td>
<td>19.18</td>
<td>100.00</td>
<td>21.07</td>
<td>45.54</td>
<td>37.89</td>
<td>26.56</td>
<td>42.24</td>
<td>23.10</td>
</tr>
<tr>
<td>THERM-ADV</td>
<td>9.10</td>
<td>11.33</td>
<td>100.00</td>
<td>20.83</td>
<td>20.02</td>
<td>19.64</td>
<td>12.75</td>
<td>17.10</td>
</tr>
<tr>
<td>LID</td>
<td>3.55</td>
<td>3.76</td>
<td>3.14</td>
<td>100.00</td>
<td>52.66</td>
<td>30.54</td>
<td>48.51</td>
<td>37.93</td>
</tr>
<tr>
<td>THERM</td>
<td>2.75</td>
<td>4.08</td>
<td>2.15</td>
<td>34.26</td>
<td>100.00</td>
<td>80.02</td>
<td>25.16</td>
<td>88.23</td>
</tr>
<tr>
<td>SAP</td>
<td>3.85</td>
<td>6.29</td>
<td>4.13</td>
<td>30.10</td>
<td>67.11</td>
<td>100.00</td>
<td>22.15</td>
<td>97.62</td>
</tr>
<tr>
<td>VANILLA-1</td>
<td>1.60</td>
<td>3.90</td>
<td>1.87</td>
<td>21.49</td>
<td>30.11</td>
<td>32.13</td>
<td>100.00</td>
<td>33.42</td>
</tr>
<tr>
<td>VANILLA-2</td>
<td>2.31</td>
<td>4.09</td>
<td>1.78</td>
<td>17.76</td>
<td>36.79</td>
<td>77.10</td>
<td>30.56</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Intriguing results (2)

Adversarial examples generalize between different DNNs

E.g., AlexNet vs InceptionV3
Intriguing results (3)

A universal adversarial perturbation

In a nutshell, **white-box** adversarial attacks can

**Fool different DNNs for almost all test examples**

*Most data points lie near the classification boundaries.*

**Fool different DNNs by the same adversarial examples**

*The classification boundaries of various DNNs are close.*

**Fool different DNNs by a single universal perturbation**

*We can turn most examples to adversarial by moving them along the same direction by the same amount.*
However, **white-box** adversarial attacks can:

- Not apply to most real-world scenarios
- Not work when the network architecture is unknown
- Not work when the weights are unknown
- Not work when querying networks is (e.g., cost) prohibitive
Black-box attacks

Panda: 0.88493
Indri: 0.00878
Red Panda: 0.00317

Substitute attack (Papernot et al., 2017)

Decision-based (Brendel et al., 2017)

Boundary-tracing (Cheng et al., 2019)

Zero-th order (Chen et al., 2017)

Natural evolution strategies (Ilyas et al., 2018)
The adversarial perturbation (for an input)

Bad local optimum, non-smooth optimization, curse of dimensionality, defense-specific gradient estimation, etc.

Our work

Learns the distribution of adversarial examples (for any input)

\[ \pi_S(x' | \theta) \]
Our work

Learns the distribution of adversarial examples (for an input)

Reduces the “attack dimension” \[ \dim(\theta) \ll \dim(x') \]

Fewer queries into the network.

Smothers the optimization

Higher attack success rates.

Characterizes the risk of the input example

New defense methods.
Our work

Learns the **distribution of adversarial examples** (for an input)

\[
\max_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y)
\]

\[
\pi_S(x'|\theta) \quad \quad \quad \pi_S(x'|\theta_0) \quad \pi_S(x'|\theta_1) \quad \pi_S(x'|\theta_2) \quad \pi_S(x'|\theta_3)
\]
Our work

Learns the distribution of adversarial examples (for an input)

$$\max_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y)$$

A sample from the distribution fails DNN by a high chance.
Which family of distributions? \( \pi_{\mathcal{S}}(x' | \theta) \)

1. draw \( z \sim \mathcal{N}(\mu, \sigma^2) \), compute \( g(z) \) as
   \[
   g(z) = \frac{1}{2}(\tanh(g_0(z)) + 1),
   \]
2. clip \( \delta' = \text{clip}_p(g(z) - x) \), \( p = 2 \) or \( \infty \), and
3. return \( \text{proj}_\mathcal{S}(g(z)) \) as \( x' = x + \delta' \)
Natural evolution strategies (NES)

\[ \nabla_\theta \mathbb{E}_{x' \sim \pi} L(x', y) \]

\[ \max_\theta \mathbb{E}_{x' \sim \pi} L(x', y) \]

Natural evolution strategies (NES)

\[ \nabla_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y) = \nabla_{\theta} \int L(x', y) \pi(x'|\theta) dx' \]

\[ \max_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y) \]

Natural evolution strategies (NES)

\[ \nabla_\theta \mathbb{E}_{x' \sim \pi} L(x', y) \]

\[ = \nabla_\theta \int L(x', y) \pi(x' | \theta) dx' \]

\[ = \int L(x', y) \nabla_\theta \pi(x' | \theta) dx' \]

\[
\max_\theta \mathbb{E}_{x' \sim \pi} L(x', y) \]

Natural evolution strategies (NES)

\[
\max_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y) \\
= \nabla_{\theta} \int L(x', y) \pi(x' | \theta) dx'
\]

Natural evolution strategies (NES)

\[
\max_\theta \mathbb{E}_{x' \sim \pi} L(x', y) \\
= \nabla_\theta \mathbb{E}_{x' \sim \pi} L(x', y) \\
= \nabla_\theta \int L(x', y) \pi(x' | \theta) dx' \\
= \int L(x', y) \nabla_\theta \pi(x' | \theta) dx' \\
= \int L(x', y) \frac{\nabla_\theta \pi(x' | \theta)}{\pi(x' | \theta)} \pi(x' | \theta) dx' \\
= \int L(x', y) \left[ \nabla_\theta \log \pi(x' | \theta) \right] \pi(x' | \theta) dx'
\]
Natural evolution strategies (NES)

\[
\max_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y) \\
= \nabla_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y) \\
= \nabla_{\theta} \int L(x', y) \pi(x'|\theta) dx' \\
= \int L(x', y) \mathbb{E}_{x' \sim \pi} \nabla_{\theta} \pi(x'|\theta) dx' \\
= \int L(x', y) \frac{\nabla_{\theta} \pi(x'|\theta)}{\pi(x'|\theta)} \pi(x'|\theta) dx' \\
= \int L(x', y) \left[ \nabla_{\theta} \log \pi(x'|\theta) \right] \pi(x'|\theta) dx' \\
= \mathbb{E}_{x' \sim \pi} L(x', y) \nabla_{\theta} \log \pi(x'|\theta)
\]

Natural evolution strategies (NES)

\[
\max_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y)
\]

\[
\nabla_{\theta} \mathbb{E}_{x' \sim \pi} L(x', y)
\]

\[
= \nabla_{\theta} \int L(x', y) \pi(x' | \theta) dx'
\]

\[
= \int L(x', y) \nabla_{\theta} \pi(x' | \theta) dx'
\]

\[
= \int L(x', y) \frac{\nabla_{\theta} \pi(x' | \theta)}{\pi(x' | \theta)} \pi(x' | \theta) dx'
\]

\[
= \int L(x', y) \left[ \nabla_{\theta} \log \pi(x' | \theta) \right] \pi(x' | \theta) dx'
\]

\[
= \mathbb{E}_{x' \sim \pi} L(x', y) \nabla_{\theta} \log \pi(x' | \theta)
\]

\[
\approx \frac{1}{b} \sum_{i=1}^{b} L(x'(i), y) \nabla_{\theta} \log \pi(x'(i) | \theta)
\]

Algorithm 1 Black-box adversarial \$N\text{ATTACK}\$

**Input:** DNN \(F(\cdot)\), input \(x\) and its label \(y\), initial mean \(\mu_0\), standard deviation \(\sigma\), learning rate \(\eta\), sample size \(b\), and the maximum number of iterations \(T\)

**Output:** \(\mu_T\), mean of the normal distribution

1: \textbf{for} \(t = 0, 1, \ldots, T - 1\) \textbf{do}
2: \hspace{1em} Sample \(\epsilon_1, \ldots, \epsilon_b \sim \mathcal{N}(0, I)\)
3: \hspace{1em} Compute \(g_i = g(\mu_t + \epsilon_i \sigma)\) by Step 1 \(\forall i \in \{1, \ldots, b\}\)
4: \hspace{1em} Obtain \(\text{proj}(g_i)\) by steps 2–3, \(\forall i\)
5: \hspace{1em} Compute losses \(f_i := f(\text{proj}(g_i)), \forall i\)
6: \hspace{1em} Z-score \(\hat{f}_i = (f_i - \text{mean}(f)) / \text{std}(f), \forall i\)
7: \hspace{1em} Set \(\mu_{t+1} \leftarrow \mu_t - \frac{\eta}{b \sigma} \sum_{i=1}^{b} \hat{f}_i \epsilon_i\)
8: \textbf{end for}
Experiment setup

Attack 13 defended DNNs & 2 vanilla DNNs

Consider both ○ and □

Examine all test examples of CIFAR10 & 1000 of ImageNet

*Excluding those misclassified by the targeted DNN*

Evaluate by attack success rates
Attack success rates, *ImageNet*
Attack success rates, \textit{CIFAR10}
Attack success rate vs. optimization steps
Transferabilities of the adversarial examples
A universally effective defense technique?

Adversarial training / defensive learning

$$\min_{\phi} \mathbb{E}_{x,y} \max_{x' \in S(x)} L(x', y)$$

The PGD attack
In a nutshell, \textit{NATTACK}

Is a \textbf{powerful black-box} attack, \(\geq\) white-box attacks

Is \textit{universal}, failed various defenses by the same algorithm

Characterizes the distributions of adversarial examples

Reduces the “attack dimension”

Speeds up the defensive learning \textit{(ongoing work)}
Physical adversarial attack

Boqing Gong

Joint work with Yang Zhang, Hassan Foroosh, & David Phil

Published in ICLR 2019
Recall the following result

A universal adversarial perturbation

Physical attack: universal perturbation → 2D mask

Physical attack: \textbf{2D mask} $\rightarrow$ \textbf{3D camouflage}

Gradient descent \textit{w.r.t.} \textbf{camouflage} $c$

in order to \textbf{minimize detection scores}

for the vehicle under all feasible locations

\[ C = C \cup \{c\} \]

Scores $\{V_t(c)|t \in T_S\}$

\[ \arg\min_c \frac{\sum_{t \in T_S} V_\theta(c, t)}{|T_S|} \]

Non-differentiable
Physical attack: 2D mask $\rightarrow$ **3D camouflage**

Repeat until done

1. Camouflage a vehicle
2. Drive it around and take many pictures of it
3. Detect it by Faster-RCNN & save the detection scores

$\rightarrow$ **Dataset:** {(camouflage, vehicle, background, detection score)}
Physical attack: 2D mask → 3D camouflage

Fit a DNN to predict any camouflage’s corresponding detection scores

$$\arg\min_{\theta} \frac{1}{|C||T_S|} \sum_{c \in C} \sum_{t \in T_S} H[s, V_\theta(c, t)] + \alpha \|\theta\|^2$$

$$C = C \cup \{c\}$$

Scores $$\{V_t(c)|t \in T_S\}$$
Physical attack: 2D mask $\rightarrow$ 3D camouflage

Gradient descent w.r.t. Camouflage $c$

in order to **minimize detection scores**

for the vehicle under all feasible locations

$C = C \cup \{c\}$

Scores $\{V_t(c) | t \in T_S\}$

Non-differentiable

But approximated by a DNN

$$\arg\min_c \frac{\sum_{t \in T_S} V_\theta(c, t)}{|T_S|}$$
Why do we care?
Observation, re-observation, & future work

Defended DNNs are still vulnerable to transfer attacks (only to some moderate degree though)

Adversarial examples from black-box attacks are less transferable than those from white-box attacks

All future work on defenses will adopt adversarial training

Adversarial training will become faster (we are working on it)

We should certify DNNs’ expected robustness by $\mathcal{N}$ ATTACK
New works to watch


