Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability

Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna
University of California, Santa Barbara
Security Threats in Machine Learning

‘Duck’ + \times 0.07 = ‘Horse’

‘How are you?’ + \times 0.01 = ‘Open the door’
Security Threats in Machine Learning

Training:

Training Data → Trained Classifier

Testing:

Test Sample → Trained Classifier → Prediction

Adversarial Examples/Evasion Attacks
Security Threats in Machine Learning

Training:
- Training Data
- Trained Classifier
- Dataset Poisoning

Testing:
- Test Sample
- Trained Classifier
- Prediction
Targeted Poisoning Against Transfer Learning

• Targeted ➔ No effect on general performance!
• Clean-label

• Introduced first against transfer learning:
  • Feature Collision (Shafahi et al., 2018)
  • Convex Polytope (Zhu et al., 2019)
What Is Transfer Learning?

- Use a pre-trained network as the feature extractor to feed the features of the input to a linear classifier.
What Is Transfer Learning?

- Use a pre-trained network as the feature extractor to feed the features of the input to a linear classifier
Goal?

- Goal: The attacker wants sample $t$ to be classified into class $P$ after the \textit{fine-tuning} phase.
- How? By adding some poisoned data to the fine-tuning set.
Feature Collision (Shafahi et al., 2018)

- $f$: The feature extractor (known to the attacker and used by victim)
  - White-box!
- $g$: The linear classifier (used by victim, not known to the attacker)
- $t$: The attacker wants sample $t$ to be classified into class $P$.

![Diagram](image)

$f(x') \sim f(t)$

- The ultimate linear classifier learns to associate $f(x')$ with the target class $P$. 
Feature Collision Attack
Feature Collision Attack

- Black-box: different feature extractor, i.e., different feature space
Convex Polytope (Zhu et al., 2019)

- Poison samples create a convex shape around the target, instead of all being close to the point!

Linear classifier
Convex Polytope – CP

- Compared to FC, CP creates a bigger shape in the feature space.
- Thus, it increases the chance of transferability in black-box settings!
- CP outperforms FC by 20% on average across all experiments.
Convex Polytope – CP

• But how such a polytope is created?

Using \( m \) surrogate networks, with corresponding \( m \) feature spaces \( \{ \phi^{(i)} \}_{i=1}^{m} \)

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2m} \sum_{i=1}^{m} \frac{\left\| \phi^{(i)}(x_t) - \sum_{j=1}^{k} c_j^{(i)} \phi^{(i)}(x_p^{(j)}) \right\|^2}{\left\| \phi^{(i)}(x_t) \right\|^2} \\
\text{subject to} & \quad \sum_{j=1}^{k} c_j^{(i)} = 1, c_j^{(i)} \geq 0, \forall i, j, \\
& \quad \left\| x_p^{(j)} - x_b^{(j)} \right\|_\infty \leq \epsilon, \forall j,
\end{align*}
\]
Convex Polytope – CP

• But how such a polytope is created?

Using $m$ surrogate networks,
with corresponding $m$ feature spaces $\{\phi^{(i)}\}_{i=1}^{i=m}$

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2m} \sum_{i=1}^{m} \left\| \phi^{(i)}(x_t) - \sum_{j=1}^{k} c_j^{(i)} \phi^{(i)}(x_p^{(j)}) \right\|^2 \\
\text{subject to} & \quad \sum_{j=1}^{k} c_j^{(i)} = 1, c_j^{(i)} \geq 0, \forall i, j, \\
& \quad \left\| x_p^{(j)} - x_b^{(j)} \right\|_{\infty} \leq \epsilon, \forall j,
\end{align*}
\]
Convex Polytope – CP

• But how such a polytope is created?

Using \( m \) surrogate networks, with corresponding \( m \) feature spaces \( \{ \phi^{(i)} \}_{i=1}^{m} \)

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2m} \sum_{i=1}^{m} \left\| \phi^{(i)}(x_t) - \sum_{j=1}^{k} c_j^{(i)} \phi^{(i)}(x_p^{(j)}) \right\|^2 \\
\text{subject to} & \quad \sum_{j=1}^{k} c_j^{(i)} = 1, c_j^{(i)} \geq 0, \forall i, j, \\
& \quad \left\| x_p^{(j)} - x_b^{(j)} \right\|_\infty \leq \epsilon, \forall j,
\end{align*}
\]
Using $m$ surrogate networks, with corresponding $m$ feature spaces $\{\phi^{(i)}\}_{i=1}^{i=m}$,

$$\text{minimize} \quad \frac{1}{2m} \sum_{i=1}^{m} \frac{\left\| \phi^{(i)}(x_t) - \frac{1}{k} \sum_{j=1}^{k} \phi^{(i)}(x_p^{(j)}) \right\|^2}{\left\| \phi^{(i)}(x_t) \right\|^2}$$

subject to $\left\| x_p^{(j)} - x_b^{(j)} \right\|_{\infty} \leq \epsilon$, $\forall j$. 

Bullseye Polytope – BP
What About End-to-end Transfer Learning?

• We enforce the convex hull heuristic at each layer of the neural network

\[
\text{minimize} \quad \frac{1}{2m} \left( \sum_{i=1}^{m} \frac{||\phi^{(i)}(x_i) - \frac{1}{k} \sum_{j=1}^{k} \phi^{(i)}(x^{(j)}_p)||^2}{||\phi^{(i)}(x_i)||^2} + \sum_{i=1}^{m} \frac{||\phi^{(i)}(x_i) - \frac{1}{k} \sum_{j=1}^{k} \phi^{(i)}(x^{(j)}_p)||^2}{||\phi^{(i)}(x_i)||^2} + \cdots + \sum_{i=1}^{m} \frac{||\phi^{(i)}(x_i) - \frac{1}{k} \sum_{j=1}^{k} \phi^{(i)}(x^{(j)}_p)||^2}{||\phi^{(i)}(x_i)||^2} \right)
\]

subject to

\[
||x^{(j)}_p - x^{(j)}_b|| \leq \epsilon, \forall j.
\]
Much More Scalable, With Improved Transferability

• Experiments Setup:
  • Using surrogate networks with 6 different architectures
  • Tested against two unseen architecture (black-box), and 6 known architectures, but with unseen parameters (different random seed is used)
  • #poisons=5, $\epsilon = 0.1$, #fine-tuning-set = 500.
Much More Scalable, With Improved Transferability

• Experiments Setup:
  • Using surrogate networks with 6 different architectures
  • Tested against two unseen architecture (black-box), and 6 known architectures, but with unseen parameters (different random seed is used)
    • #poisons=5, $\epsilon = 0.1$, #fine-tuning-set = 500.

• In linear transfer learning, **BP outperforms CP by 10%, while being 7x faster!**
Much More Scalable, With Improved Transferability

- Experiments Setup:
  - Using surrogate networks with 6 different architectures
  - Tested against two unseen architecture (black-box), and 6 known architectures, but with unseen parameters (different random seed is used)
  - \#poisons=5, $\epsilon = 0.1$, #fine-tuning-set = 500.

- In linear transfer learning, **BP outperforms CP by 10%, while being 7x faster!**

- In end-to-end transfer learning, **BP outperforms CP by 27%, while being 12x faster!**
Why is BP better?

- Is it the “bullseye idea” contributing to its superior performance?
- Or its faster algorithm allows for better optimization?
Why is BP better?

BP with different fixed coefficients.
Independent Benchmark (Schwarzschild et al., 2020)

- Linear transfer learning:

<table>
<thead>
<tr>
<th>Attack</th>
<th>CIFAR-10</th>
<th>TinyImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>White-box</td>
<td>White-box</td>
</tr>
<tr>
<td>FC</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>CP</td>
<td>33</td>
<td>7</td>
</tr>
<tr>
<td>BP</td>
<td>85</td>
<td>10</td>
</tr>
<tr>
<td>WiB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CLBD</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>HTBD</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>
Independent Benchmark (Schwarzschild et al., 2020)

• Training from scratch:
  • Specifically taken into consideration by another attack, Witches’ Brew (WiB) (Geiping et al., 2020)
  • Was published on arXiv (parallel to this work).

### Training From Scratch

<table>
<thead>
<tr>
<th>Attack</th>
<th>CIFAR-10</th>
<th>TinyImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VGG16+ResNet34+MobileNetV2</td>
<td>VGG16</td>
</tr>
<tr>
<td>FC</td>
<td>1.33</td>
<td>4</td>
</tr>
<tr>
<td>CP</td>
<td>0.67</td>
<td>0</td>
</tr>
<tr>
<td>BP</td>
<td>2.33</td>
<td>44</td>
</tr>
<tr>
<td>WiB</td>
<td>26</td>
<td>32</td>
</tr>
<tr>
<td>CLBD</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>HTBD</td>
<td>2.67</td>
<td>0</td>
</tr>
</tbody>
</table>
Defenses (Peri et al. 2019)

- Neighborhood conformity tests to sanitize the dataset!
- We evaluated against the only two effective defenses:
  - $l_2$-norm centroid
  - Deep K-NN
Deep K-NN

- For each sample in the training set:
  - Looks at its $k$ nearest neighbors, if the sample’s label is not the mode, it’s flagged!

<table>
<thead>
<tr>
<th>$k$</th>
<th># Deleted Poisons</th>
<th># Deleted Samples</th>
<th>Adv. Success Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BP</td>
<td>CP</td>
<td>BP</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>3.18</td>
<td>4.28</td>
<td>36.46</td>
</tr>
<tr>
<td>2</td>
<td>2.42</td>
<td>3.86</td>
<td>21.91</td>
</tr>
<tr>
<td>3</td>
<td>3.81</td>
<td>4.66</td>
<td>27.86</td>
</tr>
<tr>
<td>4</td>
<td>3.48</td>
<td>4.60</td>
<td>25.83</td>
</tr>
<tr>
<td>6</td>
<td>4.22</td>
<td>4.85</td>
<td>25.39</td>
</tr>
<tr>
<td>8</td>
<td>4.77</td>
<td>4.84</td>
<td>25.69</td>
</tr>
<tr>
<td>10</td>
<td>4.97</td>
<td>4.95</td>
<td>26.36</td>
</tr>
<tr>
<td>12</td>
<td>4.98</td>
<td>4.96</td>
<td>26.58</td>
</tr>
<tr>
<td>14</td>
<td>4.98</td>
<td>4.96</td>
<td>26.21</td>
</tr>
<tr>
<td>16</td>
<td>4.98</td>
<td>4.96</td>
<td>26.95</td>
</tr>
<tr>
<td>18</td>
<td>4.98</td>
<td>4.96</td>
<td>26.36</td>
</tr>
<tr>
<td>22</td>
<td>4.98</td>
<td>4.96</td>
<td>26.62</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k$</th>
<th># Deleted Poisons</th>
<th># Deleted Samples</th>
<th>Adv. Success Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BP</td>
<td>CP</td>
<td>BP</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>4.30</td>
<td>7.56</td>
<td>38.77</td>
</tr>
<tr>
<td>2</td>
<td>2.71</td>
<td>6.38</td>
<td>22.75</td>
</tr>
<tr>
<td>3</td>
<td>4.92</td>
<td>8.16</td>
<td>30.36</td>
</tr>
<tr>
<td>4</td>
<td>3.94</td>
<td>7.76</td>
<td>26.74</td>
</tr>
<tr>
<td>6</td>
<td>4.82</td>
<td>8.51</td>
<td>26.57</td>
</tr>
<tr>
<td>8</td>
<td>5.68</td>
<td>9.03</td>
<td>27.24</td>
</tr>
<tr>
<td>10</td>
<td>6.53</td>
<td>9.31</td>
<td>28.30</td>
</tr>
<tr>
<td>12</td>
<td>7.42</td>
<td>9.44</td>
<td>29.19</td>
</tr>
<tr>
<td>14</td>
<td>8.17</td>
<td>9.54</td>
<td>29.42</td>
</tr>
<tr>
<td>16</td>
<td>8.86</td>
<td>9.59</td>
<td>30.63</td>
</tr>
<tr>
<td>18</td>
<td>9.50</td>
<td>9.61</td>
<td>30.60</td>
</tr>
<tr>
<td>22</td>
<td>9.91</td>
<td>9.61</td>
<td>31.18</td>
</tr>
</tbody>
</table>

(a) # Poisons = 5

(b) # Poisons = 10
Bullseye Polytope Attack - Summary

- Clean-label data poisoning against transfer learning
- Fixes an inherent flaw of Convex Polytope!
- An order of magnitude faster!
- Higher attack success rate!
- More resilient against defenses!
References


