Adversarial Robustness via Runtime Masking and Cleansing

Yi-Hsuan Wu  Chia-Hung Yuan  Shan-Hung Wu

Department of Computer Science, National Tsing Hua University, Taiwan

International Conference on Machine Learning, 2020
Why many adversarial defenses are broken?

- Deep neural networks are shown to be vulnerable to adversarial attacks, which motivates robust learning techniques

1Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. ICML’ 2018

Y.H. Wu, C.H. Yuan, S.H. Wu
Runtime Masking and Cleansing

https://www.tensorflow.org/tutorials/generative/images/adversarial_example.png
Why many adversarial defenses are broken?

- Deep neural networks are shown to be vulnerable to adversarial attacks, which motivates robust learning techniques.

![Adversarial Example Image](https://www.tensorflow.org/tutorials/generative/images/adversarial_example.png)

- A plethora of defenses have been proposed, however, *many of these have been shown to fail*\(^1\)

---

\(^1\) Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. ICML’ 2018

Y.H. Wu, C.H. Yuan, S.H. Wu Runtime Masking and Cleansing ICML’20
Why many adversarial defenses are broken?

- Recent study\(^2\) shows the sample complexity of robust learning can be significantly larger than standard training.

\(^2\)Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. NeurIPS, 2018
Why many adversarial defenses are broken?

- Recent study\textsuperscript{2} shows the sample complexity of robust learning can be significantly larger than standard training.
- A theoretically grounded way to increase the adversarial robustness is to \textit{acquire more data}.

Why many adversarial defenses are broken?

- Recent study\(^2\) shows the sample complexity of robust learning can be significantly larger than standard training.
- A theoretically grounded way to increase the adversarial robustness is to *acquire more data*.
- This partially explains why the adversarial training, a data augmentation technique, is empirically strong.

Outline

1. Goal

2. Related Works

3. Runtime Masking and Cleansing (RMC)

4. Experiments
   - Train-Time Attacks
   - Defense-Aware Attacks

5. Implications & Conclusion
WebNN$^3$

- Use a **web-scale image database** as a manifold and project a test image onto the manifold
- Make more robust prediction by taking only the projected image as inputs

---

Drawback: 50 Billion Images May be Too Large

- Web-scale database may not be available in other domains
- Performance drops when using smaller datasets

Y.H. Wu, C.H. Yuan, S.H. Wu
Runtime Masking and Cleansing
ICML'20
Outline

1. Goal

2. Related Works

3. Runtime Masking and Cleansing (RMC)

4. Experiments
   - Train-Time Attacks
   - Defense-Aware Attacks

5. Implications & Conclusion

Y.H. Wu, C.H. Yuan, S.H. Wu  
Runtime Masking and Cleansing  
ICML’20
Goal

- Most existing defenses try to get more data at *training time*
Goal

- Most existing defenses try to get more data at *training time*
- We propose a **runtime defense**
  1. Adapts network weights $\theta$ for a test point $\hat{x}$
  2. Makes inference $\hat{y} = f(\hat{x}; \theta)$
Goal

- Most existing defenses try to get more data at \textit{training time}
- We propose a \textbf{runtime defense}
  1. Adapts network weights $\theta$ for a test point $\hat{x}$
  2. Makes inference $\hat{y} = f(\hat{x}; \theta)$
- Merits:
  - Uses \textit{potentially large test data} to improve adversarial robustness
  - Is compatible with existing train-time defenses
Challenge: Test Data are Unlabeled

- How to adapt network weights $\theta$ for unlabeled $\hat{x}$?
- Online adversarial training is not applicable

1. For each $\hat{x}$, find its KNN $N(\hat{x}; D)$ from the training set $D$
2. Augment $N(\hat{x}; D)$ with adversarial examples (cyan points) perturbed from $N(\hat{x}; D)$
3. Fine-tune the networks weights $\theta$ based on $N(\hat{x}; D)$
4. Inference $\hat{y} = f(\hat{x}; \theta)$
Challenge: Test Data are Unlabeled

- How to adapt network weights $\theta$ for unlabeled $\hat{x}$?
  - Online adversarial training is not applicable
- Extension: KNN-based online adversarial training
  1. For each $\hat{x}$, find its KNN $N(\hat{x}; D)$ from the training set $D$
  2. Augment $N(\hat{x}; D)$ with adversarial examples (cyan points) perturbed from $N(\hat{x}; D)$
  3. Fine-tune the networks weights $\theta$ based on $N(\hat{x}; D)$
  4. Inference $\hat{y} = f(\hat{x}; \theta)$

![Diagram](image)

(a)
Unfortunately, It Does Not Work!

Figure (b) shows a histogram of $N(\hat{x}; D)$ w.r.t. different labels (x-axis). $N(\hat{x}; D)$ contains examples of the same label. The adversarial point $\hat{x}$ can mislead KNN selection. Therefore, the fine-tuned $q$ ends up being less robust.
Unfortunately, It Does Not Work!

- Figure (b) shows a histogram of $N(\hat{x}; D)$ w.r.t. different labels (x-axis)
Unfortunately, It Does Not Work!

- Figure (b) shows a histogram of $N(\hat{x}; D)$ w.r.t. different labels (x-axis).
- $N(\hat{x}; D)$ contains examples of the same label.
- The adversarial point $\hat{x}$ can mislead KNN selection.
Unfortunately, It Does Not Work!

- Figure (b) shows a histogram of $\mathbb{N}(\hat{x}; D)$ w.r.t. different labels (x-axis)
- $\mathbb{N}(\hat{x}; D)$ contains examples of the same label
  - The adversarial point $\hat{x}$ can mislead KNN selection
- Therefore, the fine-tuned $\theta$ ends up being less robust
Runtime Masking and Cleansing (RMC)

- RMC *precomputes* adversarial examples
  1. Augment $D$ with adversarial examples to get $D'$
  2. Given a test point $\hat{x}$, find its KNN $\mathbb{N}(\hat{x}; D')$ from $D'$

![Diagram showing the process of RMC](image)
Runtime Masking and Cleansing (RMC)

- RMC **precomputes** adversarial examples
  1. Augment $D$ with adversarial examples to get $D'$
  2. Given a test point $\hat{x}$, find its KNN $N(\hat{x}; D')$ from $D'$
  3. Adapt the networks weights $\theta$ based on $N(\hat{x}; D')$
  4. Inference $\hat{y} = f(\hat{x}; \theta)$

![Diagram of RMC process](image)
Why Does It Work?

- As Figure (c) shows, $N(\hat{x}; D')$ is no longer misled by the adversarial $\hat{x}$.
Why Does It Work?

- As Figure (c) shows, $\mathbb{N}(\hat{x}; D')$ is no longer misled by the adversarial $\hat{x}$
- Defense effects:
  - The diverse-labeled $\mathbb{N}(\hat{x}; D')$ cleanses the $\theta$ of the non-robust patterns
  - Also, dynamically masks the network gradients
Outline

1. Goal
2. Related Works
3. Runtime Masking and Cleansing (RMC)
4. Experiments
   - Train-Time Attacks
   - Defense-Aware Attacks
5. Implications & Conclusion
Datasets

- MNIST
- CIFAR-10
- ImageNet
Outline

1 Goal

2 Related Works

3 Runtime Masking and Cleansing (RMC)

4 Experiments
   - Train-Time Attacks
   - Defense-Aware Attacks

5 Implications & Conclusion
## MNIST & CIFAR-10

### Table 1. Train-time white-box attacks
(\(\epsilon = 0.3\)) on MNIST.

<table>
<thead>
<tr>
<th></th>
<th>Acc.</th>
<th>Robustness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FGSM</td>
</tr>
<tr>
<td>Regularly Trained</td>
<td>None</td>
<td>99.3</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>99.2</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>98.2</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>99.3</td>
</tr>
<tr>
<td>Adversarially Trained w. FGSM</td>
<td>None</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>98.8</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>98.6</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>99.2</td>
</tr>
<tr>
<td>Adversarially Trained w. PGD</td>
<td>None</td>
<td>99.1</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>98.8</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>98.7</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>99.2</td>
</tr>
<tr>
<td>Regularly Trained w. Jacobian Reg.</td>
<td>None</td>
<td>94.8</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>95.9</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>94.2</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>99.3</td>
</tr>
<tr>
<td>Regularly Trained w. Cross-Lipschitz Reg.</td>
<td>None</td>
<td>99.3</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>99.2</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>99.3</td>
</tr>
</tbody>
</table>

### Table 2. Train-time white-box attacks
(\(\epsilon = 8/255\)) on CIFAR-10.

<table>
<thead>
<tr>
<th></th>
<th>Acc.</th>
<th>Robustness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FGSM</td>
</tr>
<tr>
<td>Regularly Trained</td>
<td>None</td>
<td>83.3</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>84.3</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>81.8</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>89.3</td>
</tr>
<tr>
<td>Adversarially Trained w. FGSM</td>
<td>None</td>
<td>83.2</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>89.3</td>
</tr>
<tr>
<td>Adversarially Trained w. PGD</td>
<td>None</td>
<td>78.7</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>75.6</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>73.5</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>88.3</td>
</tr>
<tr>
<td>Regularly Trained w. Jacobian Reg.</td>
<td>None</td>
<td>86.3</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>87.8</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>76.2</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>87.1</td>
</tr>
<tr>
<td>Regularly Trained w. Cross-Lipschitz Reg.</td>
<td>None</td>
<td>85.3</td>
</tr>
<tr>
<td></td>
<td>DeepNN</td>
<td>86.9</td>
</tr>
<tr>
<td></td>
<td>WebNN</td>
<td>74.5</td>
</tr>
<tr>
<td></td>
<td>RMC</td>
<td>85</td>
</tr>
</tbody>
</table>
### Table 3. Train-time white-box attacks on ImageNet.

<table>
<thead>
<tr>
<th></th>
<th>Acc.</th>
<th>Robustness $\epsilon = 8/255$</th>
<th>Robustness $\epsilon = 16/255$</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>72.9</td>
<td>8.5</td>
<td>5.2</td>
</tr>
<tr>
<td>Adv. Trained</td>
<td>62.3</td>
<td>N/A</td>
<td>52.5</td>
</tr>
<tr>
<td>DB</td>
<td>65.3</td>
<td>N/A</td>
<td>55.7</td>
</tr>
<tr>
<td>DeepNN</td>
<td>26.6</td>
<td>12.9</td>
<td>8.7</td>
</tr>
<tr>
<td>WebNN</td>
<td>27.8</td>
<td>18.8</td>
<td>15.2</td>
</tr>
<tr>
<td>RMC</td>
<td>73.6</td>
<td>62.4</td>
<td>55.9</td>
</tr>
</tbody>
</table>
For all datasets, RMC achieves the state-of-the-art robustness.
RMC yields significantly higher clean accuracy.
ImageNet

Table 3. Train-time white-box attacks on ImageNet.

<table>
<thead>
<tr>
<th></th>
<th>Acc.</th>
<th>Robustness $\epsilon = 8/255$</th>
<th>Robustness $\epsilon = 16/255$</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>72.9</td>
<td>8.5</td>
<td>5.2</td>
</tr>
<tr>
<td>Adv. Trained</td>
<td>62.3</td>
<td>N/A</td>
<td>52.5</td>
</tr>
<tr>
<td>DB</td>
<td>65.3</td>
<td>N/A</td>
<td>55.7</td>
</tr>
<tr>
<td>DeepNN</td>
<td>26.6</td>
<td>12.9</td>
<td>8.7</td>
</tr>
<tr>
<td>WebNN</td>
<td>27.8</td>
<td>18.8</td>
<td>15.2</td>
</tr>
<tr>
<td>RMC</td>
<td>73.6</td>
<td>62.4</td>
<td>55.9</td>
</tr>
</tbody>
</table>

- For all datasets, RMC achieves the state-of-the-art robustness
- RMC yields significantly *higher clean accuracy*
  - RMC does not enforce a smooth decision boundary
For all datasets, RMC achieves the state-of-the-art robustness.

RMC yields significantly higher clean accuracy.

RMC does not enforce a smooth decision boundary.

For gray- black-box attacks, please refer to our main paper.
Outline

1 Goal

2 Related Works

3 Runtime Masking and Cleansing (RMC)

4 Experiments
   - Train-Time Attacks
   - Defense-Aware Attacks

5 Implications & Conclusion
Defense-Aware Attacks

- At runtime, attackers may be aware of RMC and try to circumvent it.
Strong Attack: PGD-Skip

- Assumes that all information is exposed, including
  - Test sequence
  - $D'$ and adapted model weights $\theta$'s
Strong Attack: PGD-Skip

- Assumes that all information is exposed, including:
  - Test sequence
  - $D'$ and adapted model weights $\theta$'s
- I.e., the attack point $\hat{x}^{\text{att}}$ can \textit{bypass all previous adaptations}
RMC Could be Broken by PGD-Skip

- About 15% robustness

<table>
<thead>
<tr>
<th>q</th>
<th>0</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = 100</td>
<td>14.9</td>
<td>19.8</td>
<td>20.8</td>
</tr>
</tbody>
</table>

(a) PGD-Skip-Delayed
However, PGD-Skip is Unrealistic

- Two strong assumptions

1. Access to all data points at runtime

2. No delay to place an attack point $\hat{x}^{\text{att}}$
However, PGD-Skip is Unrealistic

- Two strong assumptions

1. Access to all data points at runtime
   - When model is publicly deployed, it is unlikely to eavesdrop every user’s input \(\hat{x}\)
2. No delay to place an attack point \(\hat{x}^{\text{att}}\)
However, PGD-Skip is Unrealistic

- Two strong assumptions

1. Access to all data points at runtime
   - When model is publicly deployed, it is unlikely to eavesdrop every user’s input $\hat{x}$

2. No delay to place an attack point $\hat{x}^{\text{att}}$
   - It is hard to mute other users
More Realistic Defense-Aware Attacks

- PGD-Skip-Partial
  - Only partial points in the input sequence are known
- PGD-Skip-Delayed
  - The adversary generates/places an attack point $\hat{x}^{\text{att}}$ with some delay
PGD-Skip-Partial

Test Sequence → Runtime

\( \hat{x}_1 \) → \( \theta_1 \) → \( \hat{y}_1 \)
\( \hat{x}_2 \) → \( \theta_2 \) → \( \hat{y}_2 \)
⋮
\( \hat{x}_{n-1} \) → \( \theta_{n-1} \) → \( \hat{y}_{n-1} \)
\( \hat{x}_n \) → \( \theta_n \) → \( \hat{y}_n \)
\( \hat{x}^{att} \) → \( \theta_{n+1} \) → \( \hat{y}^{att} \)

Hacker

Y.H. Wu, C.H. Yuan, S.H. Wu
Runtime Masking and Cleansing
ICML’20
PGD-Skip-Delayed

Test Sequence

\( \hat{x}_1 \) \( \xrightarrow{} \) \( \theta_1 \) \( \xrightarrow{} \) \( \hat{y}_1 \)

\( \hat{x}_2 \) \( \xrightarrow{} \) \( \theta_2 \) \( \xrightarrow{} \) \( \hat{y}_2 \)

\vdots

\( \hat{x}_p \) \( \xrightarrow{} \) \( \theta_p \) \( \xrightarrow{} \) \( \hat{y}_p \)

Runtime

Hacker

Y.H. Wu, C.H. Yuan, S.H. Wu
Runtime Masking and Cleansing
ICML’20 27 / 34
PGD-Skip-Delayed

Test Sequence:

\[ \hat{x}_1 \rightarrow \theta_1 \rightarrow \hat{y}_1 \]

\[ \hat{x}_2 \rightarrow \theta_2 \rightarrow \hat{y}_2 \]

\[ \vdots \]

\[ \hat{x}_p \rightarrow \theta_p \rightarrow \hat{y}_p \]

Create Attacks

Runtime

Hacker

Y.H. Wu, C.H. Yuan, S.H. Wu

Runtime Masking and Cleansing

ICML’20
PGD-Skip-Delayed

Test Sequence

\[ \hat{x}_1 \rightarrow \rightarrow \rightarrow \rightarrow \theta_1 \rightarrow \hat{y}_1 \]
\[ \hat{x}_2 \rightarrow \rightarrow \rightarrow \rightarrow \theta_2 \rightarrow \hat{y}_2 \]
\[ \vdots \rightarrow \rightarrow \rightarrow \rightarrow \vdots \rightarrow \vdots \]
\[ \hat{x}_p \rightarrow \rightarrow \rightarrow \rightarrow \theta_p \rightarrow \hat{y}_p \]

Create Attacks

Delay \( q \)

Hacker

Y.H. Wu, C.H. Yuan, S.H. Wu
Runtime Masking and Cleansing
ICML’20
PGD-Skip-Delayed

Test Sequence

\(\hat{x}_1\)  \[\rightarrow\]  \(\theta_1\)  \[\rightarrow\]  \(\hat{y}_1\)

\(\hat{x}_2\)  \[\rightarrow\]  \(\theta_2\)  \[\rightarrow\]  \(\hat{y}_2\)

\(\vdots\)

\(\hat{x}_p\)  \[\rightarrow\]  \(\theta_p\)  \[\rightarrow\]  \(\hat{y}_p\)

Create Attacks

\(\hat{x}_{\text{att}}\)  \[\rightarrow\]  \(\theta_{p+q}\)  \[\rightarrow\]  \(\hat{y}_{\text{att}}\)

Delay \(q\)

Insert Attack
The Revenge of RMC

- With some minor tweaks, RMC can defend these two attacks
  - $q$: delay of PGD-Skip-Delayed
  - “known:” portion of eavesdropped points by PGD-Skip-Partial

<table>
<thead>
<tr>
<th>$q$</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>0</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p = 50$</td>
<td>19.3</td>
<td>51</td>
<td>63.7</td>
<td>20.4</td>
<td>48.9</td>
<td>62.8</td>
<td>20.9</td>
<td>44.1</td>
<td>48.6</td>
</tr>
<tr>
<td>$p = 100$</td>
<td>25.3</td>
<td>50.8</td>
<td>55.1</td>
<td>25.5</td>
<td>51.5</td>
<td>56.1</td>
<td>39.5</td>
<td>41</td>
<td>30.6</td>
</tr>
</tbody>
</table>

(a) PGD-Skip-Delayed with $\mathbb{D}'$ replacement

<table>
<thead>
<tr>
<th>$\delta = 0.5$</th>
<th>$\delta = 0.75$</th>
<th>$\delta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>known 30%</td>
<td>50%</td>
<td>70%</td>
</tr>
<tr>
<td>$p = 50$</td>
<td>48.4</td>
<td>48.1</td>
</tr>
<tr>
<td>$p = 100$</td>
<td>64.1</td>
<td>63.1</td>
</tr>
<tr>
<td>$p = 150$</td>
<td>69.2</td>
<td>69.2</td>
</tr>
</tbody>
</table>

(b) PGD-Skip-Partial with $\mathbb{D}'$ replacement
How Long is the Delay Incurred by RMC at Runtime?

- About 1 second on CIFAR-10 and a delay of 20-40 seconds on ImageNet
  - May be acceptable for non-realtime applications
  - Can be accelerated by existing techniques
Outline

1. Goal

2. Related Works

3. Runtime Masking and Cleansing (RMC)

4. Experiments
   - Train-Time Attacks
   - Defense-Aware Attacks

5. Implications & Conclusion
Conclusions & Implications

- We proposed RMC, the first runtime defense
  - Leverages *potentially large test data* to improve the robustness of a model after deployment

Implications:

Currently, new attacks trigger new deployments
RMC could end this endless chasing game

Questions? Chat with us at session time!
Or email to: chyuan@datalab.cs.nthu.edu.tw
Conclusions & Implications

- We proposed RMC, the first runtime defense
  - Leverages *potentially large test data* to improve the robustness of a model after deployment

- Implications:
  - Currently, new attacks trigger new deployments
  - RMC could end this endless chasing game
Conclusions & Implications

- We proposed RMC, the first runtime defense
  - Leverages *potentially large test data* to improve the robustness of a model after deployment

- Implications:
  - Currently, new attacks trigger new deployments
  - RMC could end this endless chasing game

- Questions? Chat with us at session time!
  - Or email to: chyuan@datalab.cs.nthu.edu.tw