Adversarial Training for Free!

Presenters: Mehdi Dousti & Erfan Zarinkia

Security and Privacy in Machine Learning
Instructor: Dr. Sadeghzadeh

Table of Contents

The importance of Neural Networks Security

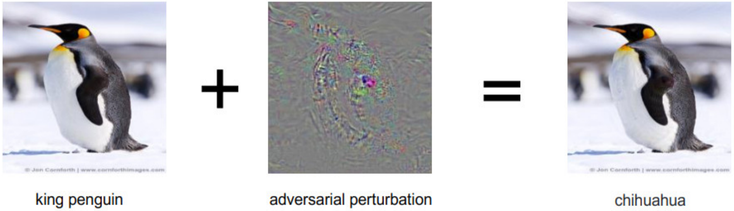
Deep neural networks have demonstrated high accuracy on various tasks in recent years

- Image classification
- Malware classification
- Autonomous driving

What is Adversarial Example?

x' is called adversarial if

- $D(x, x') < \epsilon$
- $c(x') \neq c^*(x)$



Xie et al., 2018

Types of Adversarial Examples

- Non-targeted attack

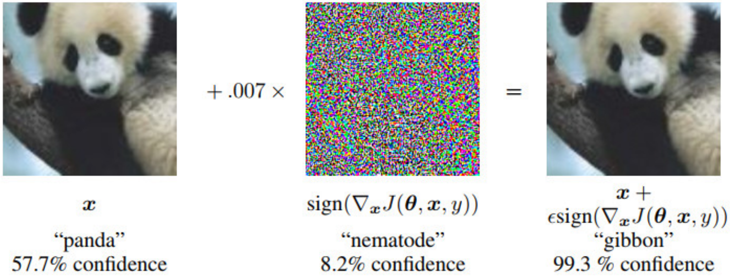
$$\max_{\delta} l(x+\delta, y, \theta)\ , \text{ subject to } \ \|\delta\|_p \leq \epsilon$$

- Targeted attack

$$\min_{\delta} l(x+\delta, t, \theta)\ , \text{ subject to } \ \|\delta\|_p \leq \epsilon$$

Fast Gradient Sign Method

$$x_{adv} = x + \epsilon \; . \; sign(\nabla_x l(x, y, \theta))$$



Goodfellow et al., 2015

K-step Projected Gradient Descent

$$x_{adv}^{n+1} = Clip_{x,\epsilon}\{x_{adv}^{n} + \alpha \, . \, sign(\nabla_x l(x_{adv}^{n}, y, \theta))\}$$

How can we defend?

- Thermometer encoding ✗
- Input transformations ✗
- Stochastic activation pruning ✗
- Leveraging generative models✗
- Using generative models ✗
- Adversarial training ✓

K-PGD Adversarial Training Algorithm

Algorithm 1 Standard Adversarial Training (K-PGD)

Require: Training samples X, perturbation bound ϵ, step size ϵ_s, maximization iterations per minimization step K, and minimization learning rate τ

Initialize θ
for epoch $= 1 \dots N_{ep}$ **do**
    for minibatch $B \subset X$ **do**
        Build x_{adv} for $x \in B$ with PGD:
            Assign a random perturbation
                $r \leftarrow U(-\epsilon, \epsilon)$
                $x_{adv} \leftarrow x + r$
            for $k = 1 \dots K$ **do**
                $g_{adv} \leftarrow \nabla_x l(x_{adv}, y, \theta)$
                $x_{adv} \leftarrow x_{adv} + \epsilon_s \cdot \mathrm{sign}(g_{adv})$
                $x_{adv} \leftarrow \mathrm{clip}(x_{adv}, x - \epsilon, x + \epsilon)$
            end for
        Update θ with stochastic gradient descent:
            $g_\theta \leftarrow \mathbb{E}_{(x,y) \in B}[\nabla_\theta \, l(x_{adv}, y, \theta)]$
            $\theta \leftarrow \theta - \tau g_\theta$
    end for
end for

Adversarial Training Problems

- Extra computations
- Time consuming

What are the alternatives?

- Replace the perturbation with a parameterized generator network ✗
- Regularize the training loss using label smoothing, or logit squeezing✗
- Certified defenses ✗
- Free adversarial training ✓

Adversarial Training for Free!

Adversarial Training for Free!

Ali Shafahi
University of Maryland
ashafahi@cs.umd.edu

Mahyar Najibi
University of Maryland
najibi@cs.umd.edu

Amin Ghiasi
University of Maryland
amin@cs.umd.edu

Zheng Xu
University of Maryland
xuzh@cs.umd.edu

John Dickerson
University of Maryland
john@cs.umd.edu

Christoph Studer
Cornell University
studer@cornell.edu

Larry S. Davis
University of Maryland
lsd@umiacs.umd.edu

Gavin Taylor
United States Naval Academy
taylor@usna.edu

Tom Goldstein
University of Maryland
tomg@cs.umd.edu

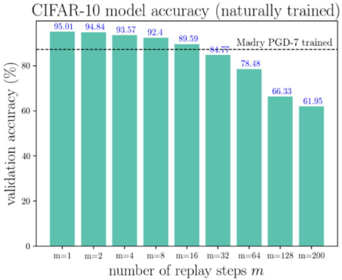
Free Adversarial Training Algorithm

Algorithm 1 "Free" Adversarial Training (Free-m)

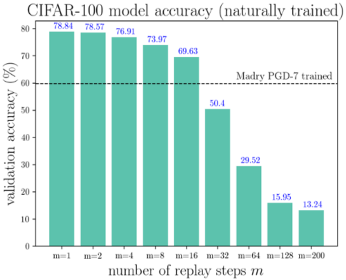
Require: Training samples X, perturbation bound ϵ, learning rate τ, hop steps m

```
Initialize θ
δ ← 0
for epoch = 1 ... N_ep/m do
    for minibatch B ⊂ X do
        for i = 1 ... m do
            Update θ with stochastic gradient descent
                g_θ ← E_{(x,y)∈B}[∇_θ l(x + δ, y, θ)]
                g_adv ← ∇_x l(x + δ, y, θ)]
                θ ← θ − τ g_θ
            Use gradients calculated for the minimization step to update δ
                δ ← δ + ε · sign(g_adv)
                δ ← clip(δ, −ε, ε)
        end for
    end for
end for
```

The Effect of Mini-batch Replay



(a) CIFAR-10 sensitivity to m



(b) CIFAR-100 sensitivity to m

CIFAR-10

Table 1: Validation accuracy and robustness of CIFAR-10 models trained with various methods.

Training	Evaluated Against					Train Time (min)
	Nat. Images	PGD-20	PGD-100	CW-100	10 restart PGD-20	
Natural	**95.01%**	0.00%	0.00%	0.00%	0.00%	**780**
Free $m = 2$	91.45%	33.92%	33.20%	34.57%	33.41%	816
Free $m = 4$	87.83%	41.15%	40.35%	41.96%	40.73%	800
Free $m = 8$	85.96%	**46.82%**	**46.19%**	**46.60%**	**46.33%**	785
Free $m = 10$	83.94%	46.31%	45.79%	45.86%	45.94%	785
7-PGD trained	87.25%	45.84%	45.29%	46.52%	45.53%	5418

CIFAR-100

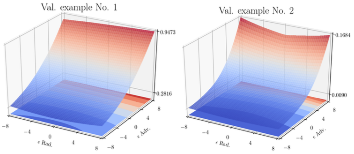
Table 2: Validation accuracy and robustness of CIFAR-100 models trained with various methods.

Training	Evaluated Against			Training Time (minutes)
	Natural Images	PGD-20	PGD-100	
Natural	**78.84%**	0.00%	0.00%	811
Free $m = 2$	69.20%	15.37%	14.86%	816
Free $m = 4$	65.28%	20.64%	20.15%	**767**
Free $m = 6$	64.87%	23.68%	23.18%	791
Free $m = 8$	62.13%	**25.88%**	**25.58%**	780
Free $m = 10$	59.27%	25.15%	24.88%	776
Madry *et al.* (2-PGD trained)	67.94%	17.08%	16.50%	2053
Madry *et al.* (7-PGD trained)	59.87%	22.76%	22.52%	5157

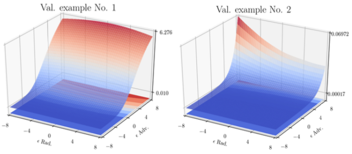
Generative Behavior

Figure 2: Attack images built for adversarially trained models look like the class into which they get misclassified. We display the last 9 CIFAR-10 clean validation images (top row) and their adversarial examples built for a 7-PGD adversarially trained (middle) and our "free" trained (bottom) models.

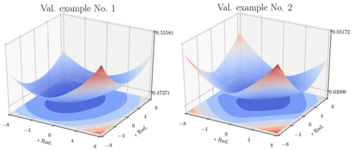
Smooth Loss Surface



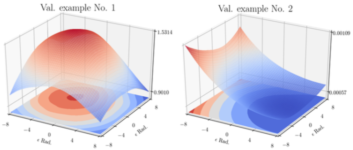
(a) Free $m = 8$



(b) 7-PGD adv trained



(c) Free $m = 8$ both rad



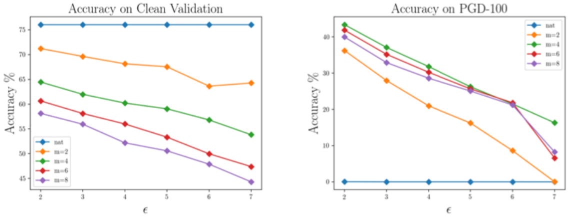
(d) 7-PGD adv trained both rad

ImageNet

Table 3: ImageNet validation accuracy and robustness of ResNet-50 models trained with various replay parameters and $\epsilon = 2$.

Training	Evaluated Against			
	Natural Images	PGD-10	PGD-50	PGD-100
Natural	**76.038%**	0.166%	0.052%	0.036%
Free $m = 2$	71.210%	37.012%	36.340%	36.250%
Free $m = 4$	64.446%	**43.522%**	**43.392%**	**43.404%**
Free $m = 6$	60.642%	41.996%	41.900%	41.892%
Free $m = 8$	58.116%	40.044%	40.008%	39.996%

ImageNet



(a) Clean (b) PGD-100

Figure 4: The effect of the perturbation bound ϵ and the mini-batch replay hyper-parameter m on the robustness achieved by free training.

ImageNet

Table 4: Validation accuracy and robustness of "free" and 2-PGD trained ResNet-50 models – both trained to resist ℓ_∞ $\epsilon = 4$ attacks. Note that **2-PGD training time is** $3.46\times$ **that of "free" training**.

Model & Training	Evaluated Against				Train time (minutes)
	Natural Images	PGD-10	PGD-50	PGD-100	
RN50 – Free $m = 4$	60.206%	32.768%	31.878%	31.816%	**3016**
RN50 – 2-PGD trained	**64.134%**	**37.172%**	**36.352%**	**36.316%**	10,435

Table 5: Validation accuracy and robustness of free-$m = 4$ trained ResNets with various capacities.

Architecture	Evaluated Against			
	Natural Images	PGD-10	PGD-50	PGD-100
ResNet-50	60.206%	32.768%	31.878%	31.816%
ResNet-101	63.340%	35.388%	34.402%	34.328%
ResNet-152	**64.446%**	**36.992%**	**36.044%**	**35.994%**

Conclusions

- Pros
  - Boosts the robustness and interpretability of neural networks
  - Can be further combined with other defenses to produce robust models without a slowdown
  - Cost nearly equal to natural training
- Cons
  - The effect of mini-batch size on the robustness of models is not scrutinized.
  - All the experiments are done on different types of Res-Net.
  - They've not compared their approach with the FGSM

References

[1] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!," in Proceedings of annual Conference on Neural Information Processing Systems, NeurIPS, 2019

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in Proceedings of the 3rd International Conference on Learning Representations, ICLR (Poster), 2015

[3] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," in Proceedings of the 35th International Conference on Machine Learning, ICML, 2018

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in Proceedings of the 6th International Conference on Learning Representations, ICLR (Poster), 2018

[5] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating Adversarial Effects Through Randomization," in Proceedings of the 6th International Conference on Learning Representations, ICLR (Poster), 2018

Questions?

بسم الله الرحمن الرحیم

Data Augmentation Can Improve Robustness

DeepMind

NeurIPS 2021

Reihaneh Zohrabi, Masoud Khodaverdian

SPML Course Presentation

Spring 2023

Outline

- Introduction
- Preliminaries
- Related Works
- Observations & Hypothesis
- Experiments and Results
- Strengths and Weaknesses
- Conclusion and Future Works

Introduction

robust overfitting

large absolute improvements robust accuracy
compared to previous state-of-the-art methods

Preliminaries

Data augmentation

Preliminaries

More sophisticated techniques

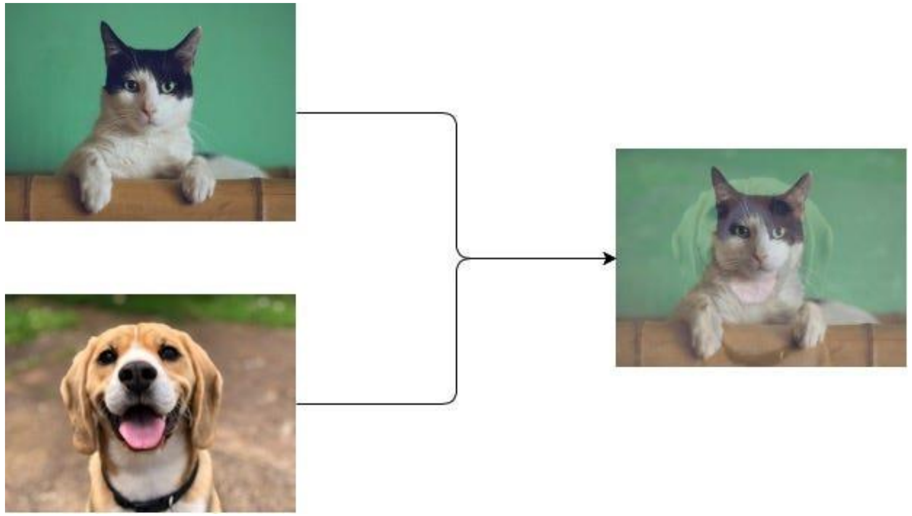
Cutout

random occlusions

CutMix

replaces parts of an image
with another

MixUp



linearly interpolates
between two images

Preliminaries

Averaging Weights Leads to Wider Optima and Better Generalization
2018

Model weight averaging

$$\boldsymbol{\theta}' \leftarrow \tau \cdot \bar{\boldsymbol{\theta}}' + (1 - \tau) \cdot \boldsymbol{\theta}$$

model parameters θ with a decay rate τ <u>at each training step</u>

Related Works

Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang. Unlabeled data improves adversarial robustness. In Adv. Neural Inform. Process. Syst., 2019.

D. Hendrycks, K. Lee, and M. Mazeika. Using Pre-Training Can Improve Model Robustness and Uncertainty. Int. Conf. Mach. Learn., 2019.

A. Najafi, S.-i. Maeda, M. Koyama, and T. Miyato. Robustness to adversarial perturbations in learning from incomplete data. Adv. Neural Inform. Process. Syst., 2019.

J. Uesato, J.-B. Alayrac, P.-S. Huang, R. Stanforth, A. Fawzi, and P. Kohli. Are labels required for improving adversarial robustness? Adv. Neural Inform. Process. Syst., 2019.

R. Zhai, T. Cai, D. He, C. Dan, K. He, J. Hopcroft, and L. Wang. Adversarially Robust Generalization Just Requires More Unlabeled Data. arXiv preprint arXiv:1906.00555, 2019.

using additional data improves adversarial robustness

Related Works

S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. arXiv preprint arXiv:2010.03593, 2020. URL https://arxiv.org/pdf/2010.03593.

L. Rice, E. Wong, and J. Z. Kolter. Overfitting in adversarially robust deep learning. Int. Conf. Mach. Learn., 2020.

D. Wu, S.-t. Xia, and Y. Wang. Adversarial weight perturbation helps robust generalization. Adv. Neural Inform. Process. Syst., 2020.

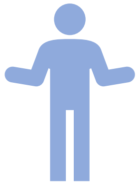
data augmentation techniques did not boost robustness

Observation & Hypothesis

using additional data improves adversarial robustness

data augmentation techniques did not boost robustness

dichotomy



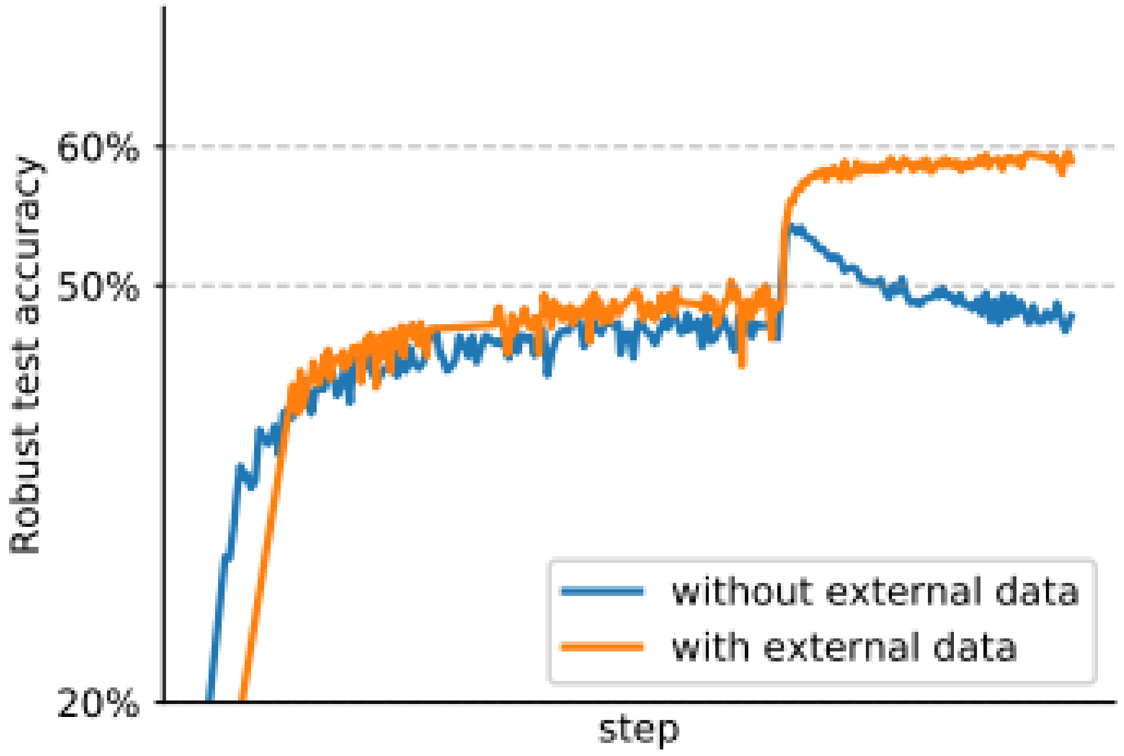
Observation & Hypothesis

is it possible to fix the training procedure such
that data augmentation becomes useful ?

data augmentation + weight averaging

robust generalization

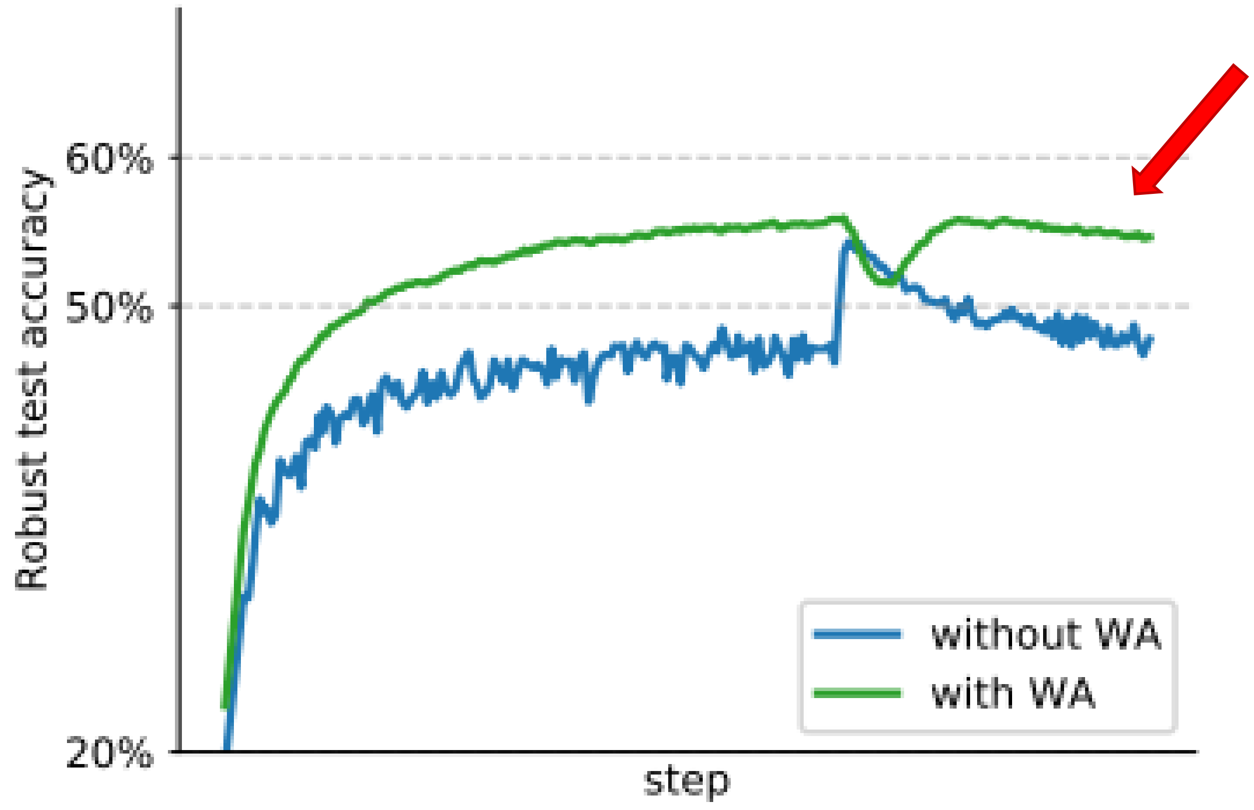
Observation & Hypothesis



Adversarial training with and without additional data from 80M-TI (without WA)

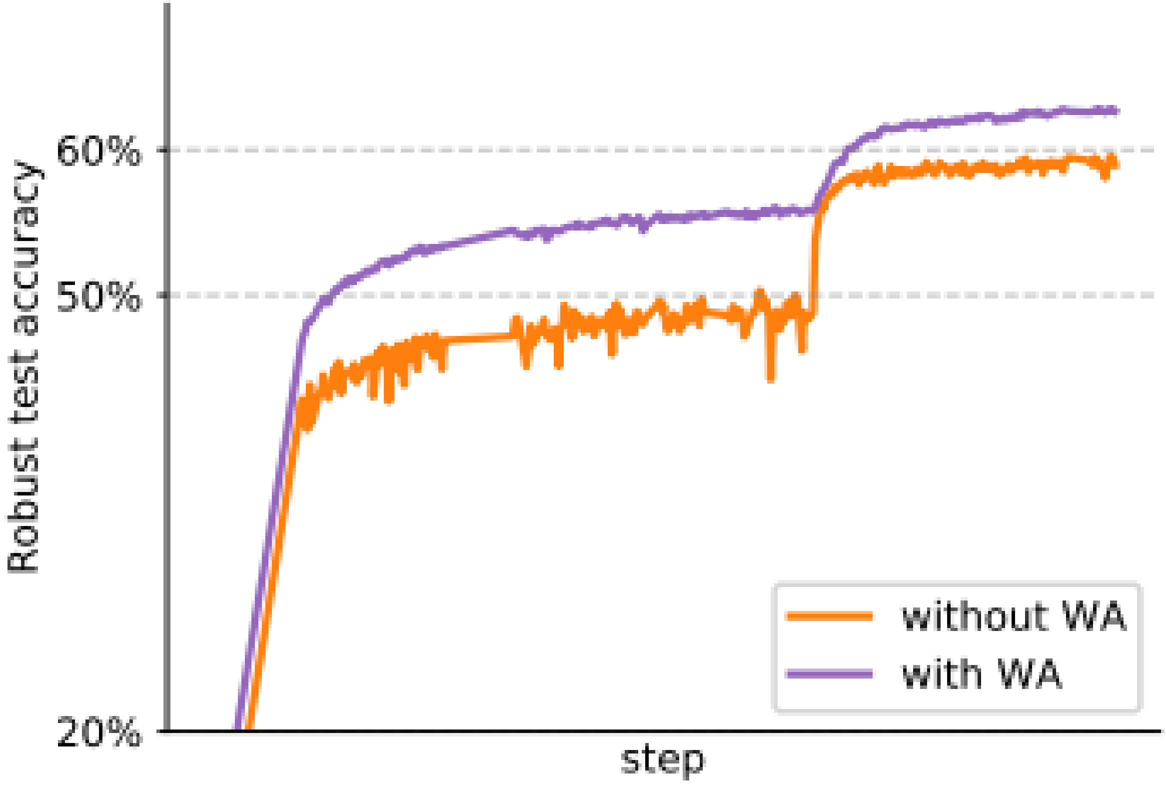
80 million tiny images

Observation & Hypothesis



Effect of WA without external data

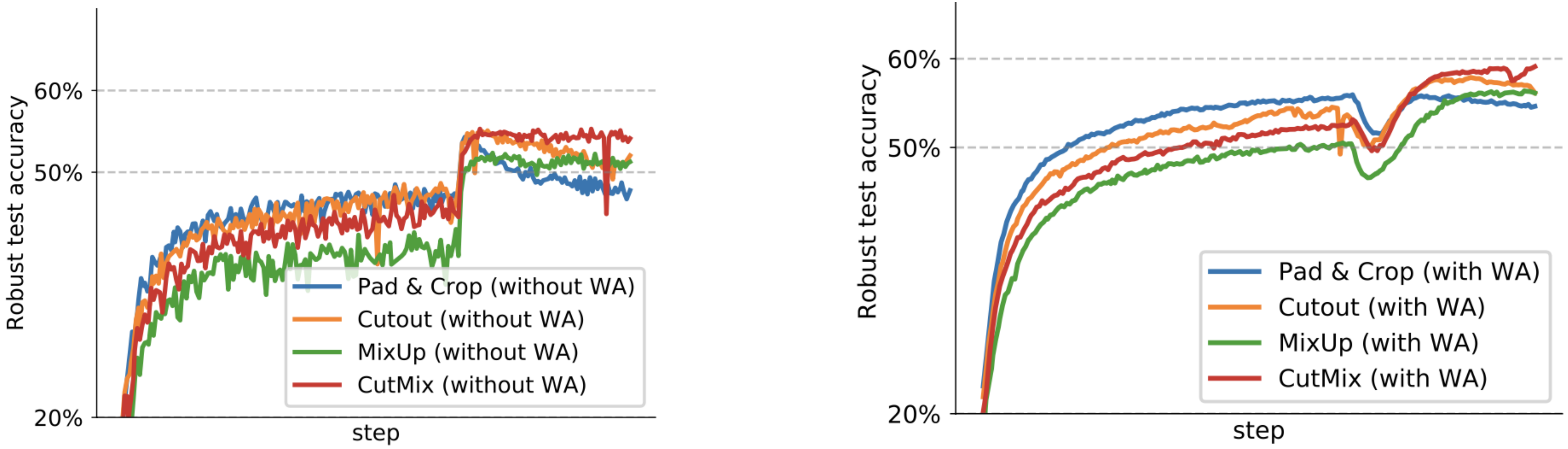
Observation & Hypothesis



Effect of WA with external data

WA remains effective and useful even when robust overfitting disappears

Observation & Hypothesis



(a) Without WA

(b) With WA

Observation & Hypothesis

model weight averaging helps robustness to a greater extent when robust accuracy between model iterations can be maintained

WA acts as a temporal ensemble

Experimental Results

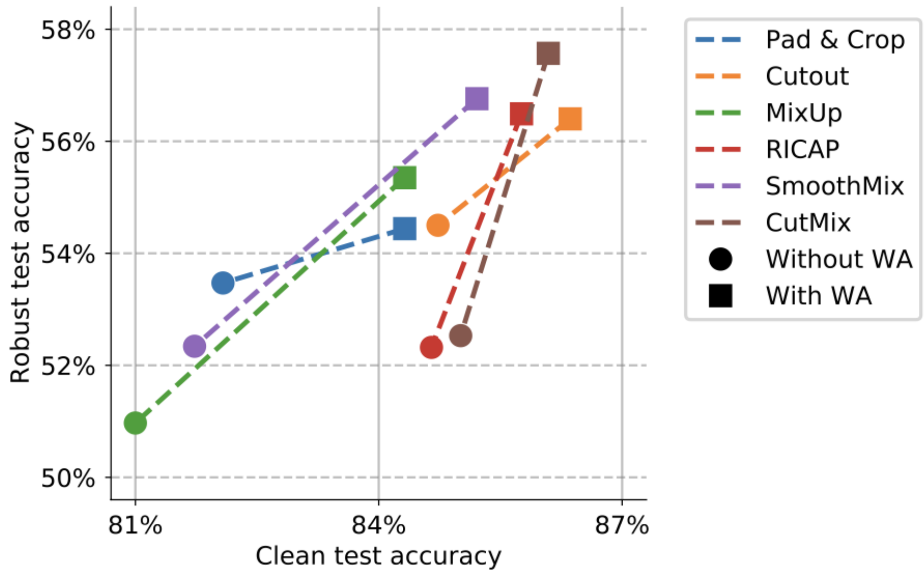
Comparing Data Augmentations:

4 Top squares

- occlude local information with patching
- +3.06% in robust accuracy for *CutMix*
- +1.54% in clean accuracy

Pad & Crop and Cutout

- suffering from robust overfitting
- benefit the least of WA

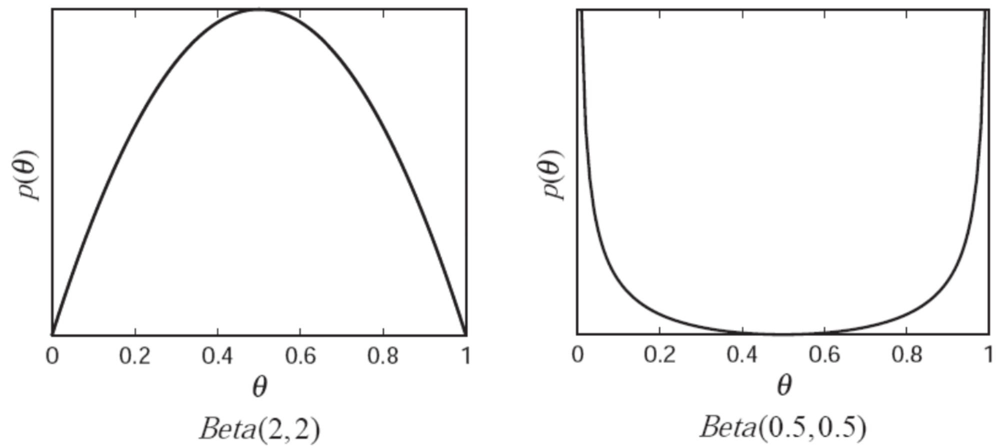


Experimental Results

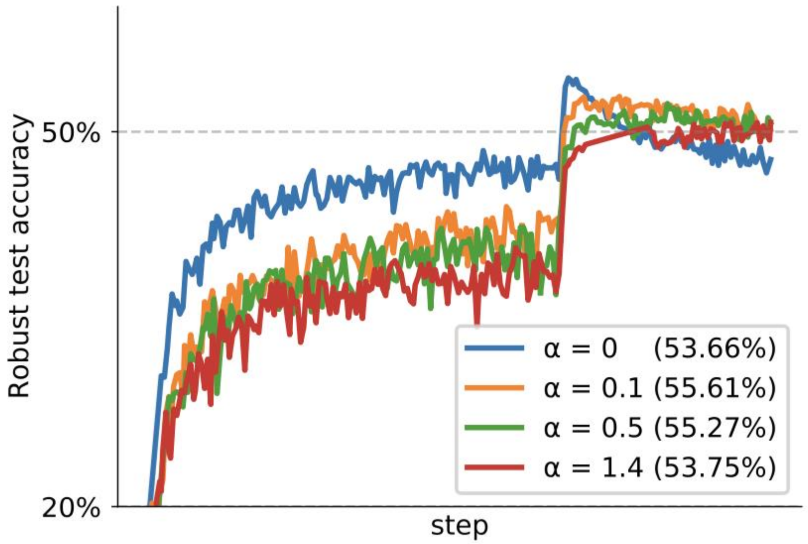
Comparing Data Augmentations:

MixUp

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j,$$



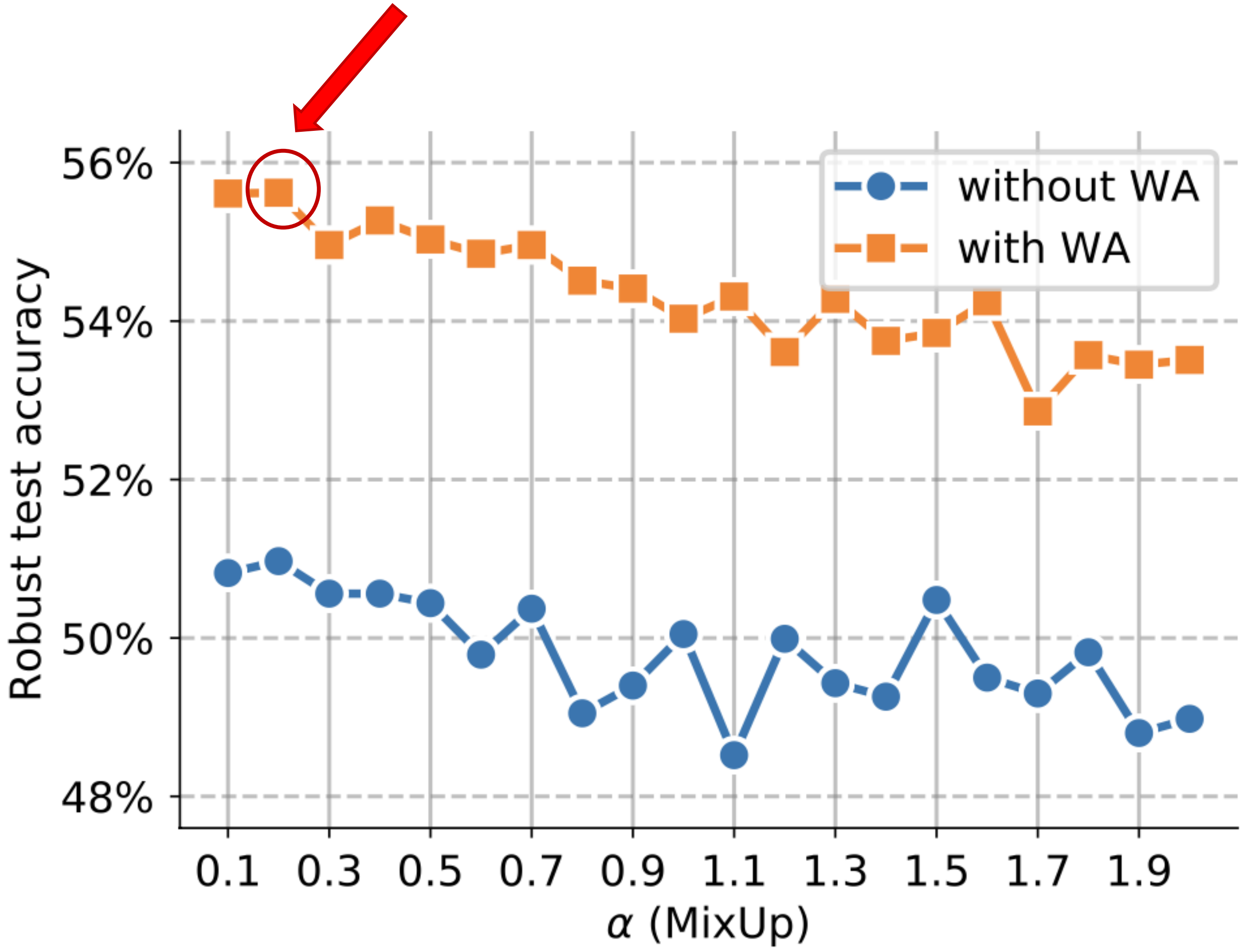
$$\lambda \sim Beta(\alpha, \alpha)$$



Experimental Results

Comparing Data Augmentations:

MixUp

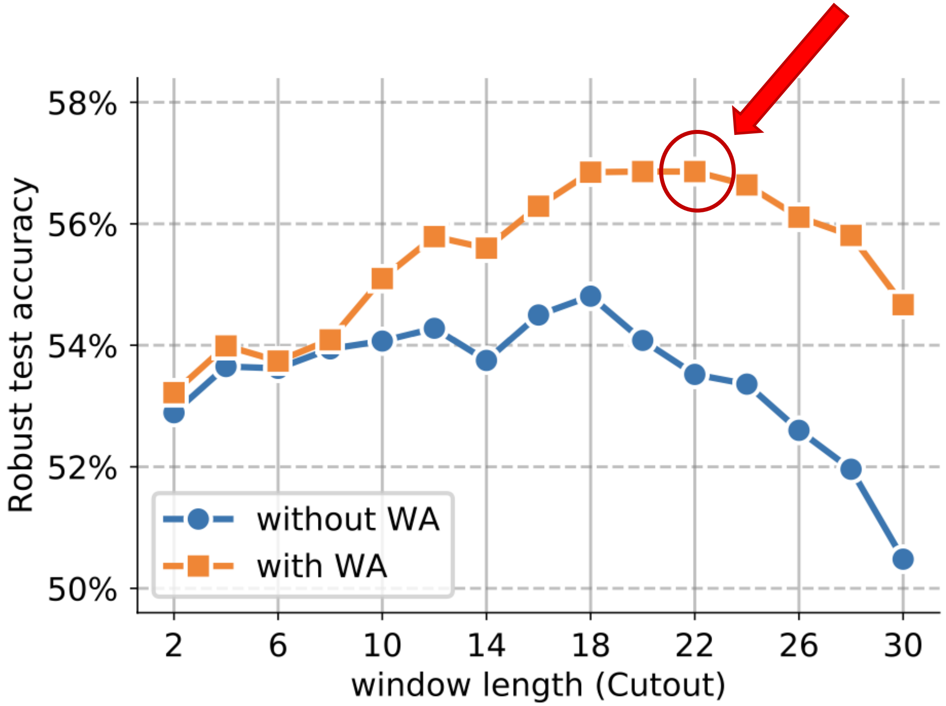


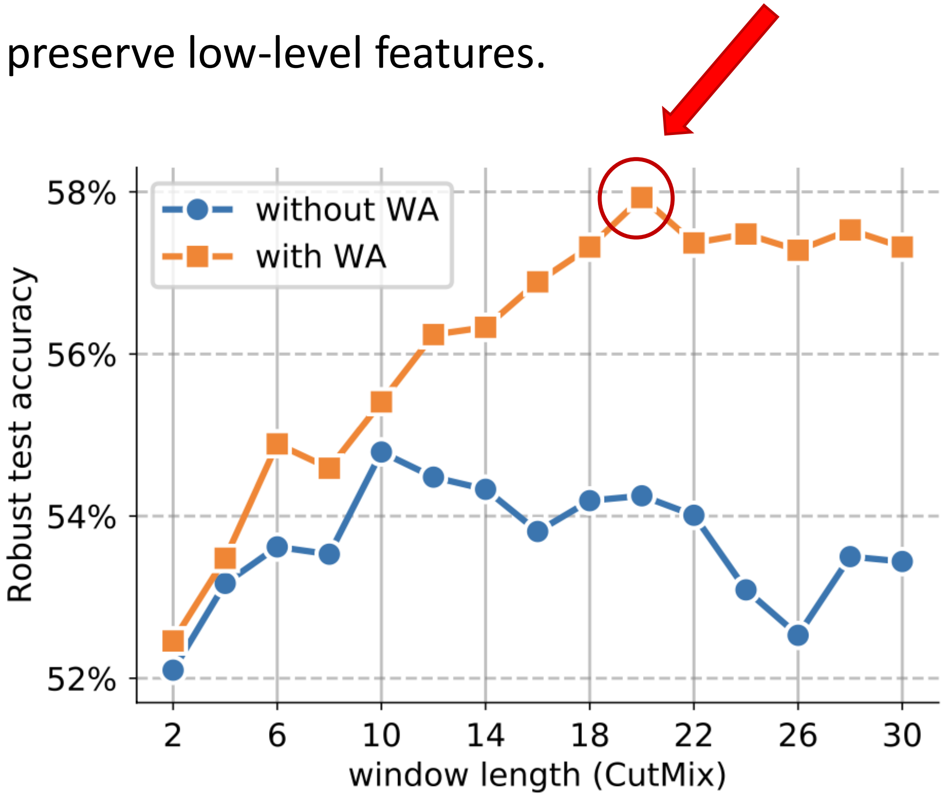
Experimental Results

Comparing Data Augmentations:

Spatial Composition Techniques

Augmentations designed for robustness need to preserve low-level features.





Experimental Results

Generalization to other architectures:

Setup	Pad & Crop Clean	Pad & Crop Robust	CutMix Clean	CutMix Robust
Varying the architecture				
ResNet-18	83.12%	50.52%	80.57%	**52.28%**
ResNet-34	84.68%	52.52%	83.35%	**54.80%**
WRN-28-10	84.32%	54.44%	86.09%	**57.50%**
WRN-34-10	84.89%	55.13%	86.18%	**58.09%**
WRN-34-20	85.80%	55.69%	87.80%	**59.25%**
WRN-70-16	86.02%	57.17%	87.25%	**60.07%**

Experimental Results

Generalization to another threat model:

Setup	ℓ_∞ Clean	ℓ_∞ Robust	ℓ_2 Clean	ℓ_2 Robust
WRN-28-10				
Gowal et al. [20] (trained by us)	84.32%	54.44%	88.60%	72.56%
Ours (CutMix)	86.22%	**57.50%**	91.35%	**76.12%**
WRN-70-16				
Gowal et al. [20] (trained by us)	85.29%	57.14%	90.90%	74.50%
Ours (CutMix)	87.25%	**60.07%**	92.43%	**76.66%**

Experimental Results

Generalization to other Datasets:

MODEL	CLEAN	AA+MT	AA
CIFAR-100			
Cui et al. [14] (WRN-34-10)	60.64%	–	29.33%
WRN-28-10 (retrained)	59.05%	28.75%	–
WRN-28-10 (CutMix)	62.97%	**30.50%**	**29.80%**
Gowal et al. [20] (WRN-70-16)	60.86%	30.67%	30.03%
WRN-70-16 (retrained)	59.65%	30.62%	–
WRN-70-16 (CutMix)	65.76%	**33.24%**	**32.43%**
SVHN			
WRN-28-10 (retrained)	92.87%	56.83%	–
WRN-28-10 (CutMix)	94.52%	**57.32%**	–
TINYIMAGENET			
WRN-28-10 (retrained)	53.27%	21.83%	–
WRN-28-10 (CutMix)	53.69%	**23.83%**	–

Experimental Results

Model Ensemebling:

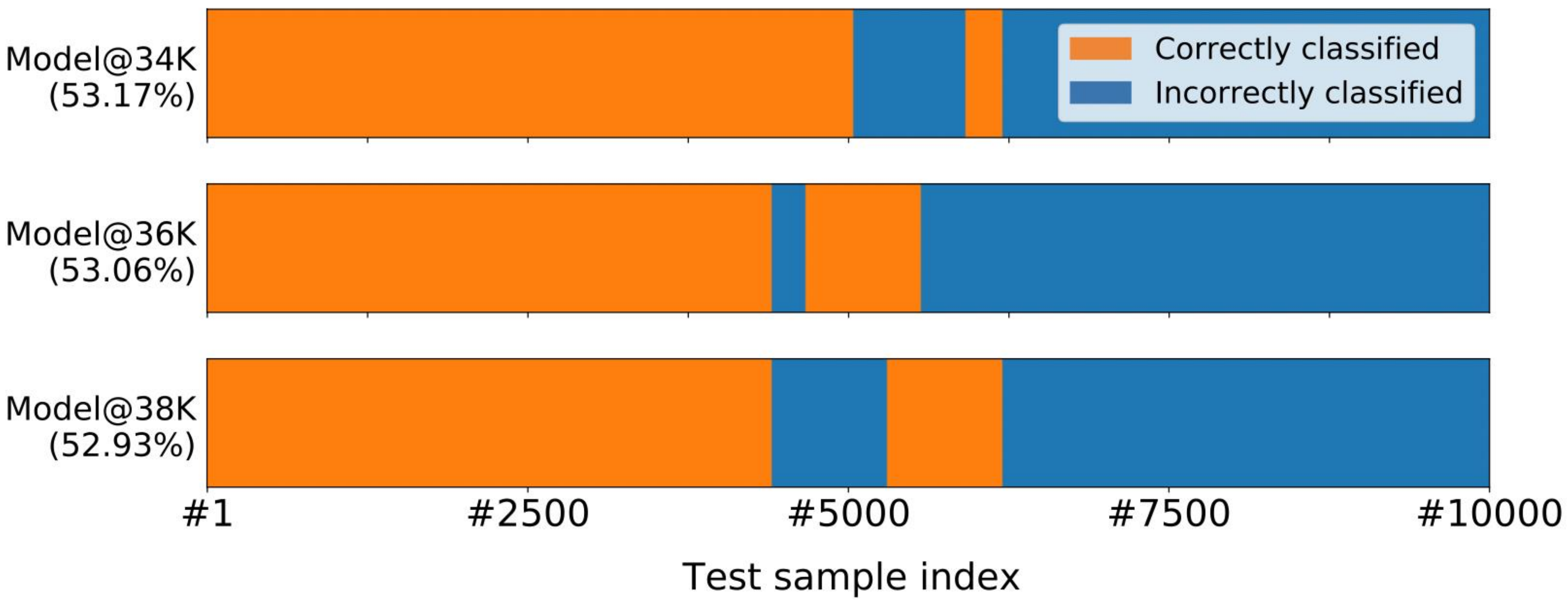
- Two ensembled early-stopped WRN
- Trained from scratch independently
- Cifar10
- 54.44 (Single) ->55.69 (Ensemble-Pad & Crop)
- 54.44 (Single) ->56.35 (Ensemble-CutMix) More Diversity

Ensembling by its ability of exploiting the diversity of the models is mainly responsible for robustness improvements.

Experimental Results

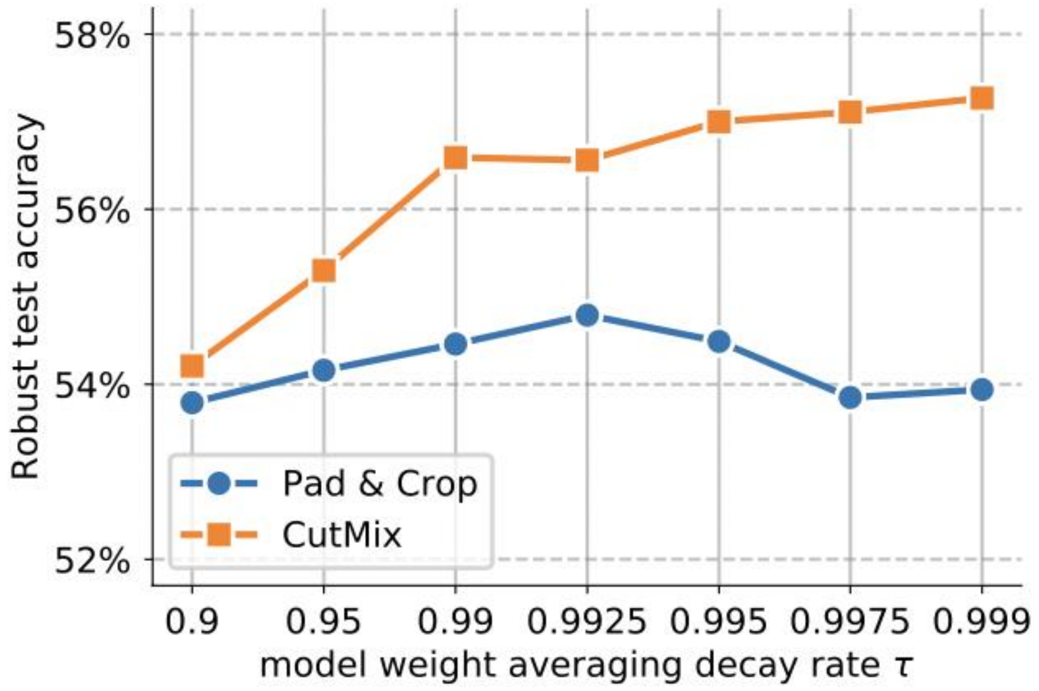
Model Ensemebling by WA:

Similar robust performance but also some diversity in individual robust predictions



Experimental Results

The limits of exploiting diversity:



Strengths and Weaknesses

- New insights into effectiveness of weight averaging with data augmentations for adversarial robustness
- Thorough experimental evaluation of approach on multiple datasets and against several strong adversarial attacks, demonstrating significant improvements in robust accuracy.
- A practical approach for model ensembeling by using weight averaging, which is computationally efficient and can be easily integrated into existing training pipelines.

- No Novelty in proposed method
- No analysis of the robust overfitting problem
- The weight averaging decay rate can be sensitive to the specific dataset and architecture used. The optimal decay rate for weight averaging may vary depending on the characteristics of the dataset and the complexity of the model, and finding the best decay rate may require some trial and error experimentation.

Conclusion and Future Works

- The combination of **data augmentation** and **model weight averaging** improves adversarial robustness.
- Previous attempts using only data augmentation were not successful.
- Weight averaging works better with data augmentations that reduce robust overfitting.
- Model snapshots during training have diverse individual predictions, allowing for a performance boost when ensembled.
- These insights can be used to improve the robustness of machine learning models against adversarial attacks.

Conclusion and Future Works

- Investigation of other data augmentation techniques
- Exploration of other ensembling techniques: other ensembling methods could be explored, such as boosting or bagging.
- Extension to other domains: such as natural language processing or speech recognition.
- Investigation of the limits of ensembling: The paper shows that ensembling can improve adversarial robustness, but there are likely limits to how much ensembling can help. Further investigation into the limits of ensembling could help researchers understand when ensembling is most effective and when it is less useful.

References

[7] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang. Unlabeled data improves adversarial robustness. In *Adv. Neural Inform. Process. Syst.*, 2019.
[10] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.
[11] F. Croce, M. Andriushchenko, V. Sehwag, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.
[15] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.
[25] D. Hendrycks, K. Lee, and M. Mazeika. Using Pre-Training Can Improve Model Robustness and Uncertainty. *Int. Conf. Mach. Learn.*, 2019.

Thanks for your attention.

Any Question?

Sharif University of Technology
Department of Computer Engineering

Adversarial Examples for Malware Detection

Presented by :

Reza Saeedi
Abolfazl Farhadi

As:

Course Seminar of Security and Privacy in Machine Learning

2023

Overview

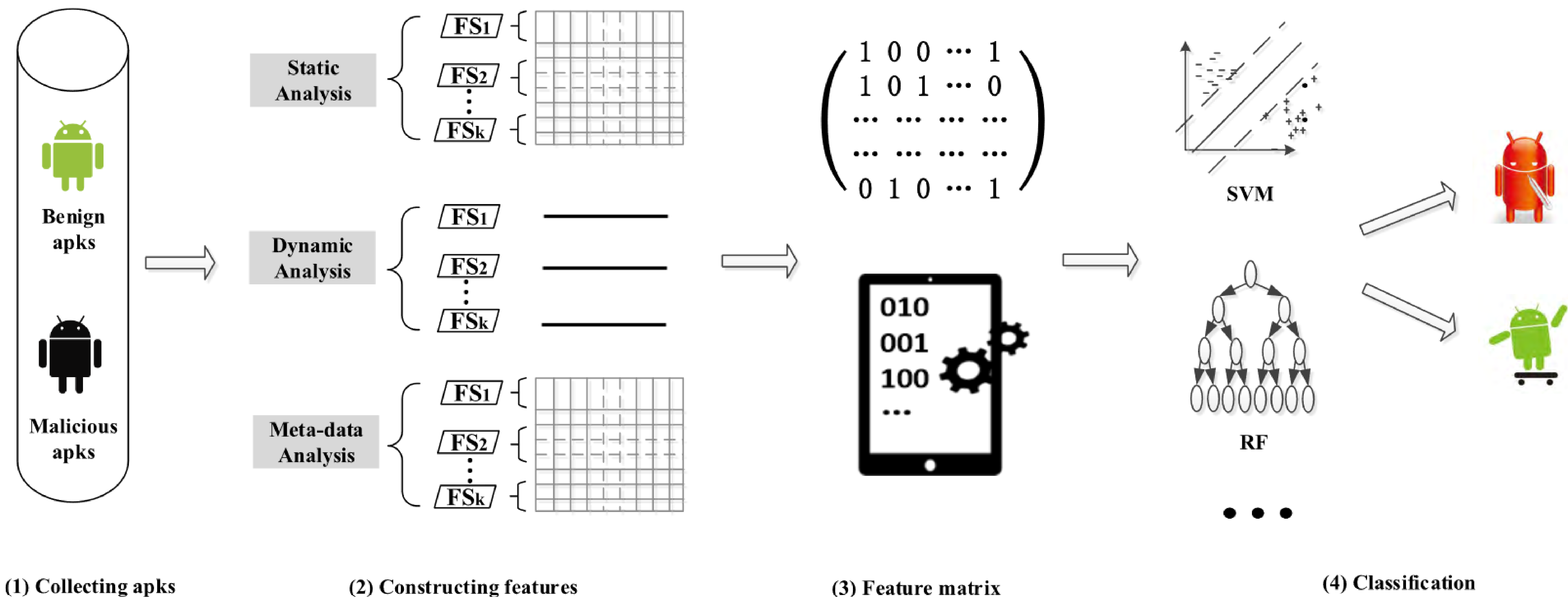
About this paper

Adversarial Examples for Malware Detection

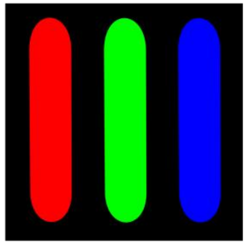
- Grosse, K., Papernot, N., Manoharan, P., Backes, M., McDaniel, P. (2017).
- Lecture Notes in Computer Science(), vol 10493. Springer, Cham.

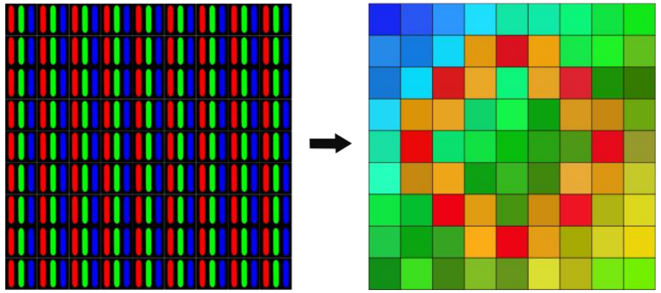
- Expand on existing adversarial example crafting algorithms to construct a highly-effective attack that uses adversarial examples against **android malware detection** models.
- Their technique **guarantees** the malware **functionality** of the adversarially manipulated program
- Using the augmented adversarial crafting algorithm They then manage to mislead this classifier for **63%** of all malware samples
- They investigate potential **defense mechanisms** for hardening their neural networks against adversarial examples..

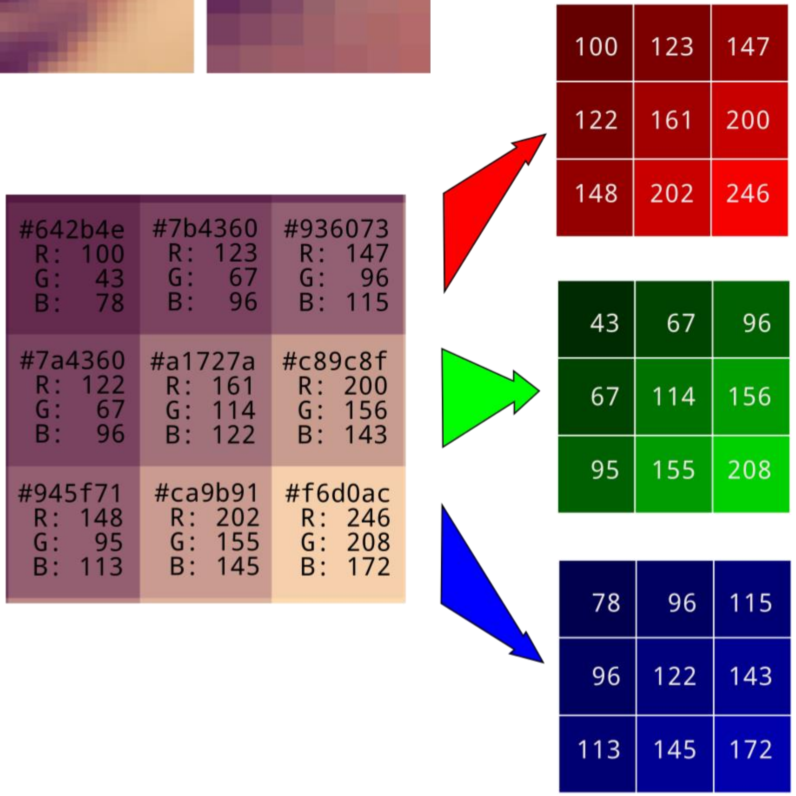
Android Malware Detection



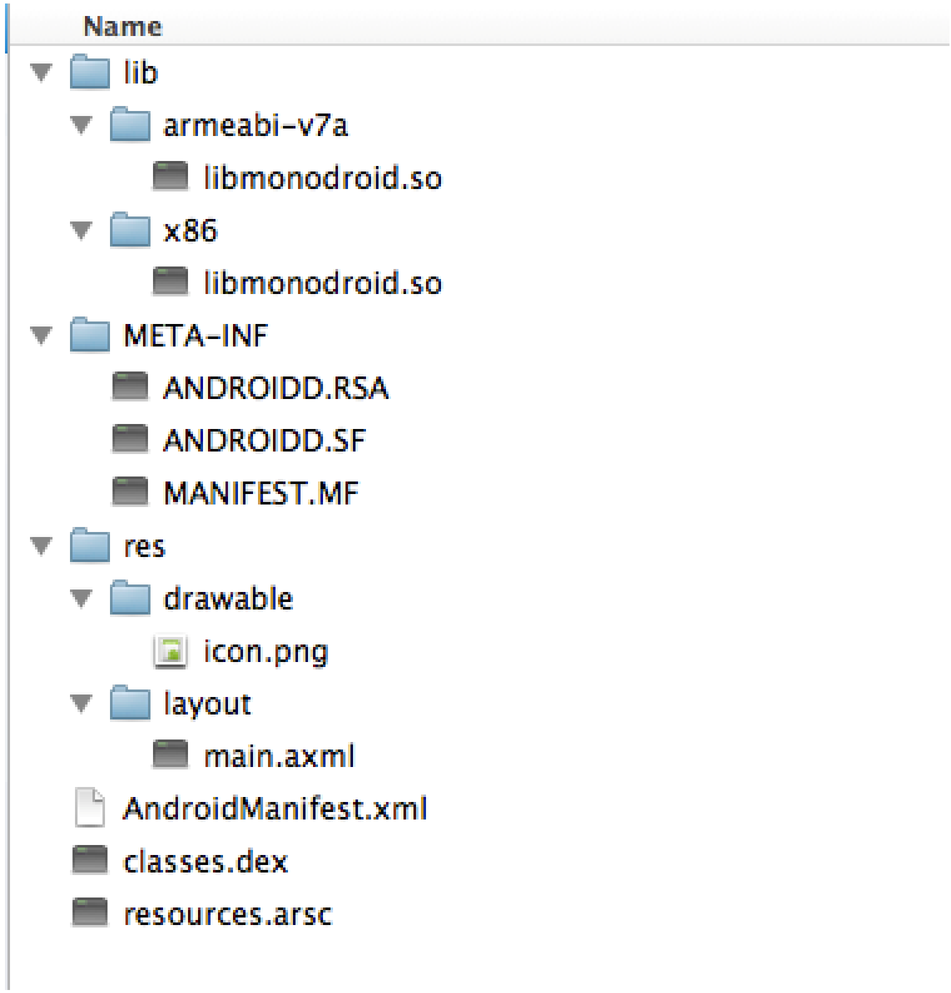
Input Domains In DNN

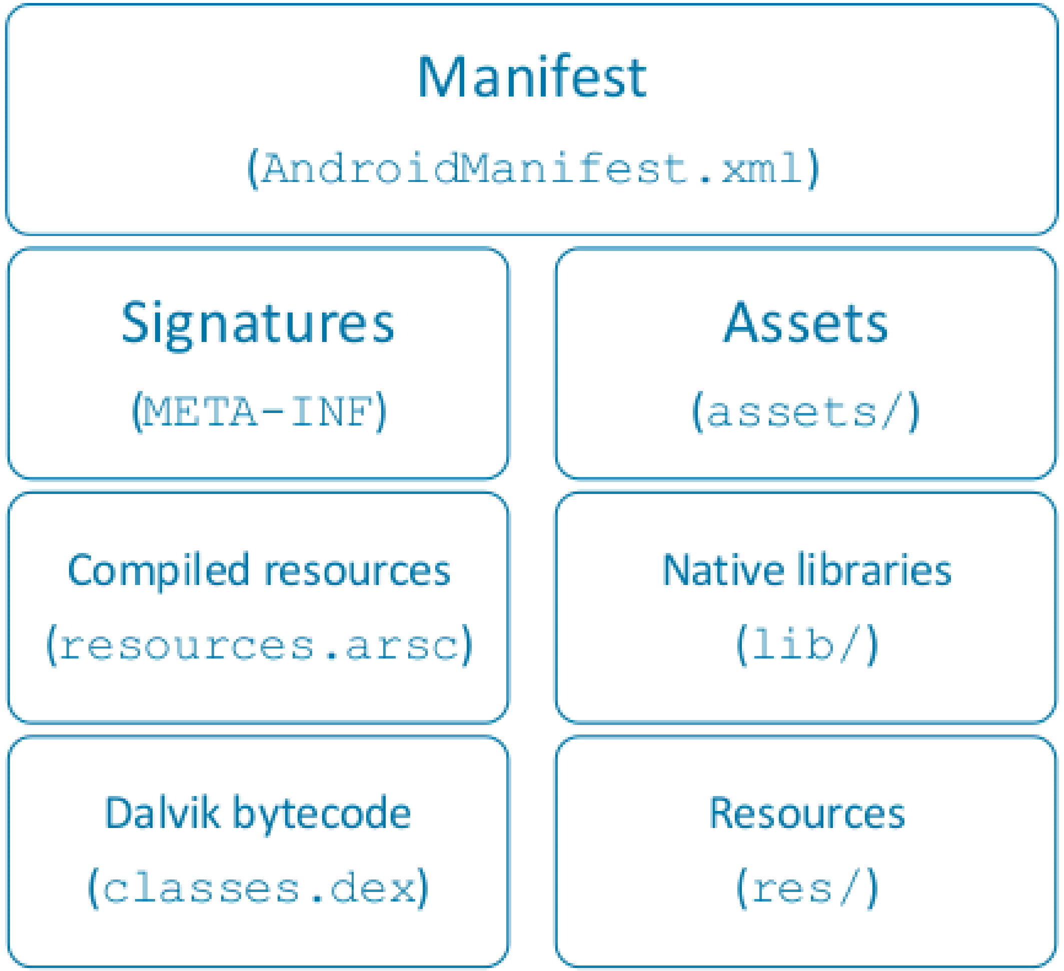




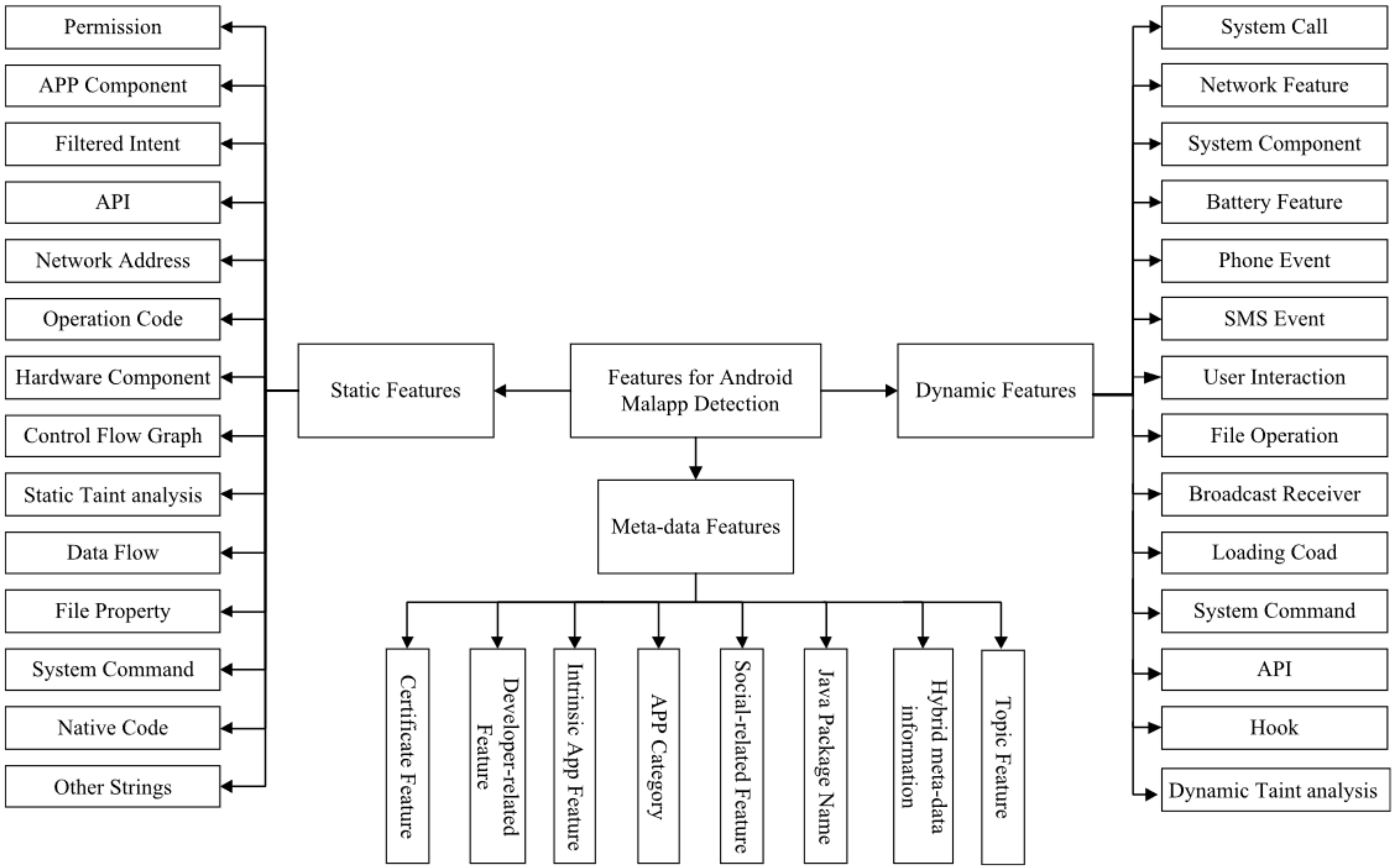


APK Structure





Android Features



W. Wang et al.: Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy, and Directions

Android Manifest File

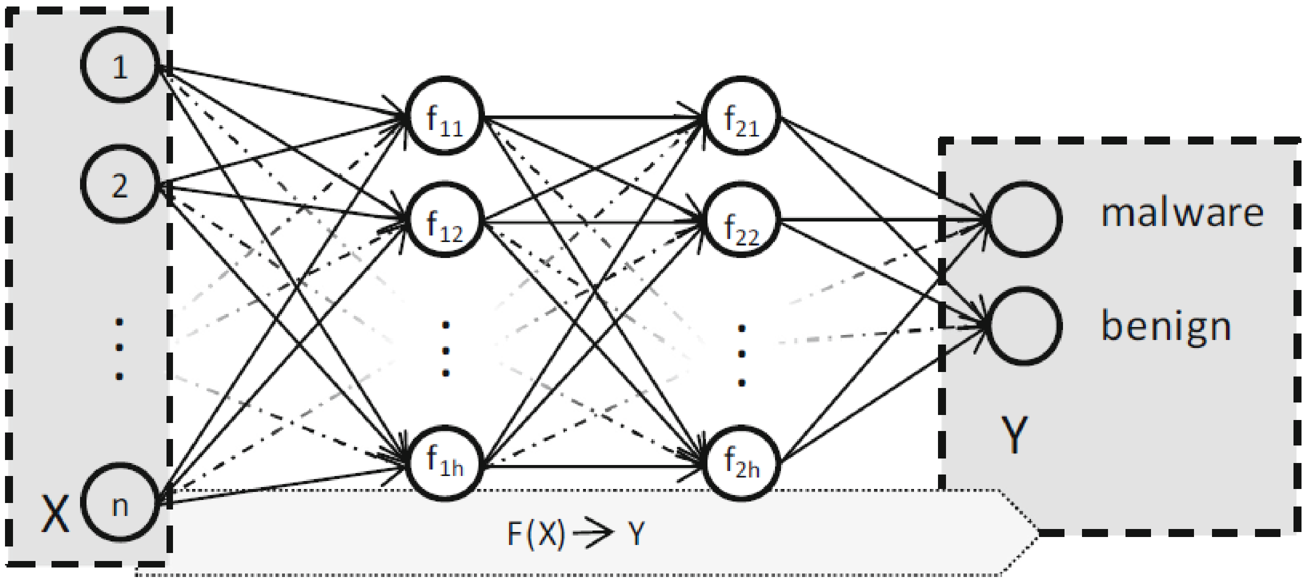
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.wujeng.data.android"
      android:versionCode="1"
      android:versionName="1.0">

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ControllerActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".StartupIntentReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <category android:name="android.intent.category.HOME" />
            </intent-filter>
    </receiver>
    <service android:name=".DataService"
             android:exported="true"
             android:process=":remote">
    </service>
    </application>
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.INTERNET">
    </uses-permission>
</manifest>
```

Features

- DREBIN data set
  - 129,013 APK
  - 123,453 benign
  - 5,560 malicious
    - 179 different malware families
    - August 2010 to October 2012
- Static features
- Feature classes from the manifest
- 8 feature classes
- 545,333 features
- binary value(feature is present in an application or not)

Android Froyo	Froyo	2.2 – 2.2.3	8	May 20, 2010
Android Gingerbread	Gingerbread	2.3 – 2.3.2	9	December 6, 2010
		2.3.3 – 2.3.7	10	February 9, 2011
Android Honeycomb	Honeycomb	3.0	11	February 22, 2011
		3.1	12	May 10, 2011
		3.2 – 3.2.6	13	July 15, 2011
Android Ice Cream Sandwich	Ice Cream Sandwich	4.0 – 4.0.2	14	October 18, 2011
		4.0.3 – 4.0.4	15	December 16, 2011
Android Jelly Bean	Jelly Bean	4.1 – 4.1.2	16	July 9, 2012
		4.2 – 4.2.2	17	November 13, 2012

DNN Model



Notation:

n = number features
h = number hidden units = 200
f = Activation function: rectified linear

X = input to network
Y = output of network
F = function learnt by neural network during training

$$\mathbf{F}_i(\mathbf{X}) = \frac{e^{x_i}}{e^{x_0} + e^{x_1}}\ , \quad x_i = \sum_{j=1}^{m_n} w_{j,i} \cdot x_j + b_{j,i}$$

Performance of the classifiers

Classifier/MR	Accuracy	FNR	FPR	MR	Dist.
Sayfullina et al. [32]	91%	0.1	17.9	–	–
Arp et al. [2]	93.9%	1	6.1	–	–
Zhu et al. [39]	98.7%	7.5	1	–	–
Ours, 0.3	98.35%	9.73	1.29	63.08	14.52
Ours, 0.4	96.6%	8.13	3.19	64.01	14.84
Ours, 0.5	95.93%	6.37	3.96	69.35	13.47

Crafting Adversarial Malware Examples

Algorithm 1. Crafting adversarial examples for Malware Detection

Input: $\mathbf{x}$, $\mathbf{y}$, $\mathbf{F}$, $\mathbf{k}$, $\mathbf{I}$

$\mathbf{x}^* \leftarrow \mathbf{x}$

$\Gamma = \{1 \dots |\mathbf{x}|\}$

while $\arg\max_j \mathbf{F_j}(\mathbf{x}^*) \neq \mathbf{y}$ and $||\delta_{\mathbf{X}}|| < \mathbf{k}$ **do**

  Compute forward derivative $\nabla \mathbf{F}(\mathbf{x}^*)$

  $i_{max} = \arg\max_{j \in \Gamma \cap \mathbf{I}, X_j = 0} \frac{\partial \mathbf{F}_y(\mathbf{X})}{\partial \mathbf{X}_j}$

  if $i_{max} \leq 0$ **then**

    return Failure

  end if

  $\mathbf{x}^*_{i_{max}} = 1$

  $\delta_{\mathbf{x}} \leftarrow \mathbf{x}^* - \mathbf{x}$

end while

return $\mathbf{x}^*$

Features in Adversarial Examples

Only 0.0004%, or 89, are used to mislead the classifier.
A quarter occurs in more than 1, 000 adversarially crafted examples

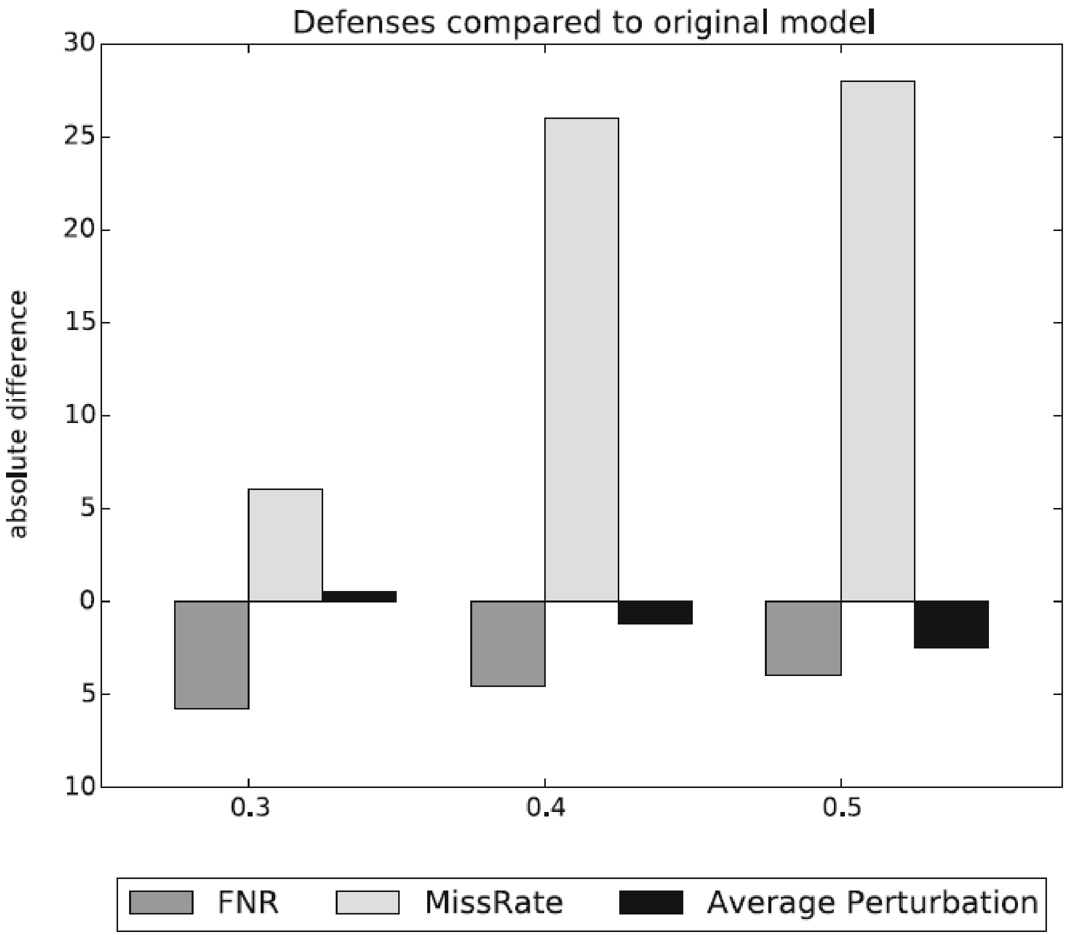
Feature	Total (0.3)	Total (0.4)	Total (0.5)
Activity	16 (3)	14 (5)	14 (2)
Feature	10 (1)	10 (3)	9 (3)
Intent	18 (7)	19 (5)	15 (5)
Permission	44 (11)	38 (10)	29 (10)
Provider	2 (1)	2 (1)	2 (1)
Service_receiver	8 (1)	6 (1)	8 (1)
$\sum$	99 (25)	90 (26)	78 (23)

Defensive Distillation

1. Given the original classifier F and the samples X, construct a new training data set D={(x,F(x)) | x∈X} that is labeled with F’s output at high temperature.

2. Construct a new neural network F′ with the same architecture as F

3. Train F′ on D.

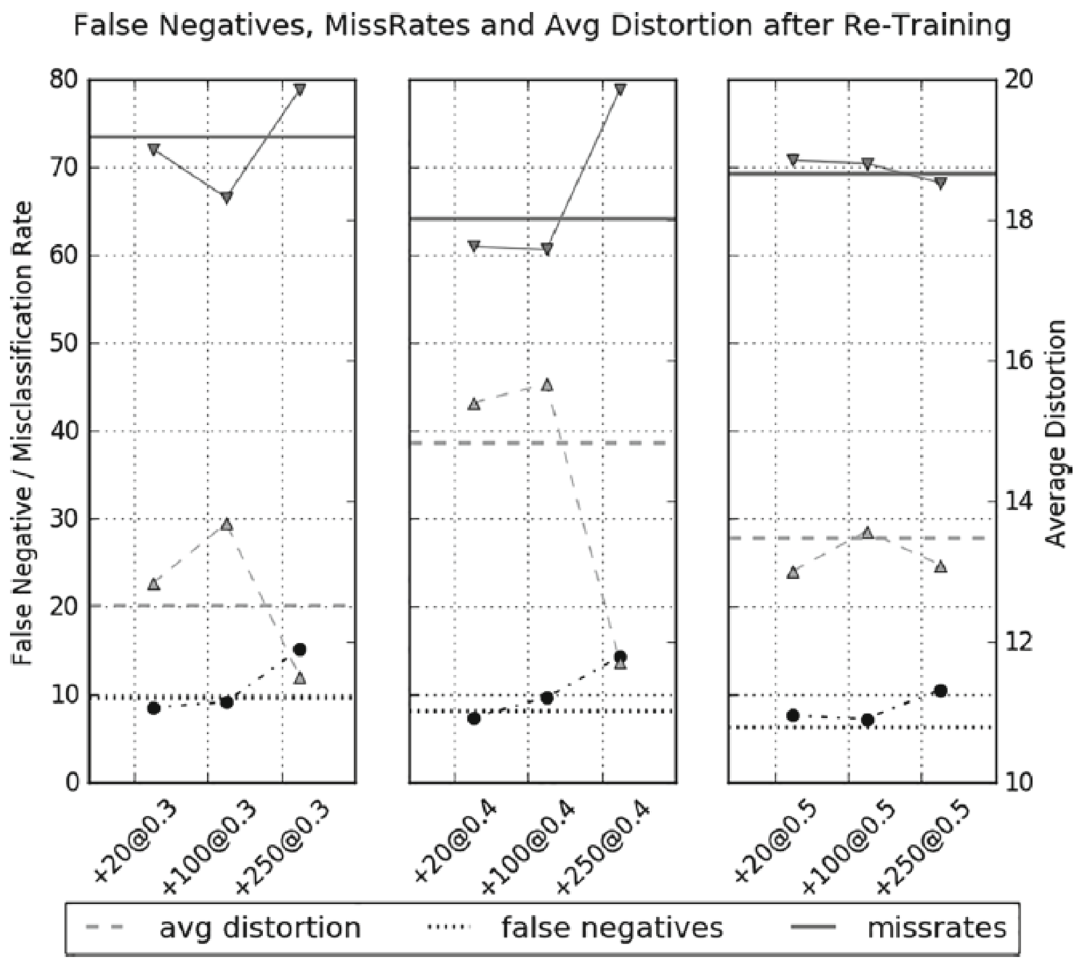


Adversarial Training

1. Train the classifier F on original data set D=B∪M, where B is the set of benign, and M the set of malicious applications

2. Craft adversarial examples A for F using the forward gradient method (n1=20, n2=100 and n3=250 additional adversarial examples)

3. Iterate additional training epochs on F with the adversarial examples from the last step as additional, malicious samples.



Comments

1. **Dataset**
   - Malware rate
2. **Selected features**
   - Static vs dynamic
   - manifest
3. **White box attack**
4. **Adversarial training**

Resources

[1] Grosse, K., Papernot, N., Manoharan, P., Backes, M., McDaniel, P. (2017). Adversarial Examples for Malware Detection.

[2] W. Wang et al.: Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy, and Directions

Questions

?

PERCEPTUAL ADVERSARIAL ROBUSTNESS: DEFENSE AGAINST UNSEEN THREAT MODELS

Hamidreza Amirzadeh
Ali Abdollahi

SPML – Presentation1
Spring 2023

Introduction

- **Key challenge in adversarial robustness:**
  - Lack of a precise mathematical characterization of human perception.

- **Current approaches:**
  - bound by L_2 or L_∞ distance, spatial perturbations, …
  - Main drawback: **No** transferable robustness.

- **Contribution of this paper:**
  - Propose adversarial training against the set of **all** imperceptible adversarial examples.
  - So we need a **suitable** perceptual distance metric.

True perceptual threat model (TPTM)

- **True perceptual distance $\mathbf{d}^*(\boldsymbol{x_1}, \boldsymbol{x_2})$ between images $\boldsymbol{x_1}, \boldsymbol{x_2}$:**
  - Measures how different two images appear to **humans.**
  - Perceptibility threshold ε^*:
  - $\mathrm{d}^*(x_1, x_2) \leq \varepsilon^* \Leftrightarrow$ images x_1, x_2 are indistinguishable from one another to **humans.**

- **True perceptual threat model:**
  - All adversarial examples $\tilde{x}$ which cause **misclassification** but are **truly imperceptible** different from x, i.e. $\mathrm{d}^*(x, \tilde{x}) \leq \varepsilon^*$.

- **Problem:**
  - $\mathrm{d}^*(.,.)$ can not be easily computed

Neural perceptual threat model (NPTM)

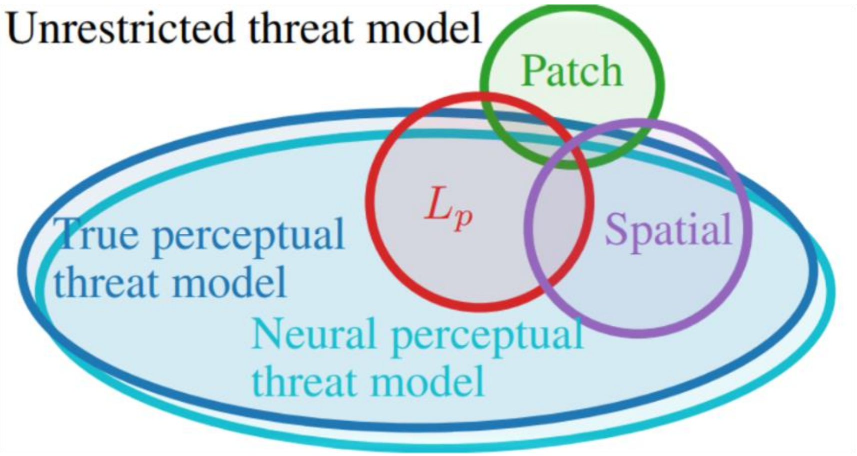
- True perceptual distance is not practical.
- Instead use an approximation: Neural perceptual distance.

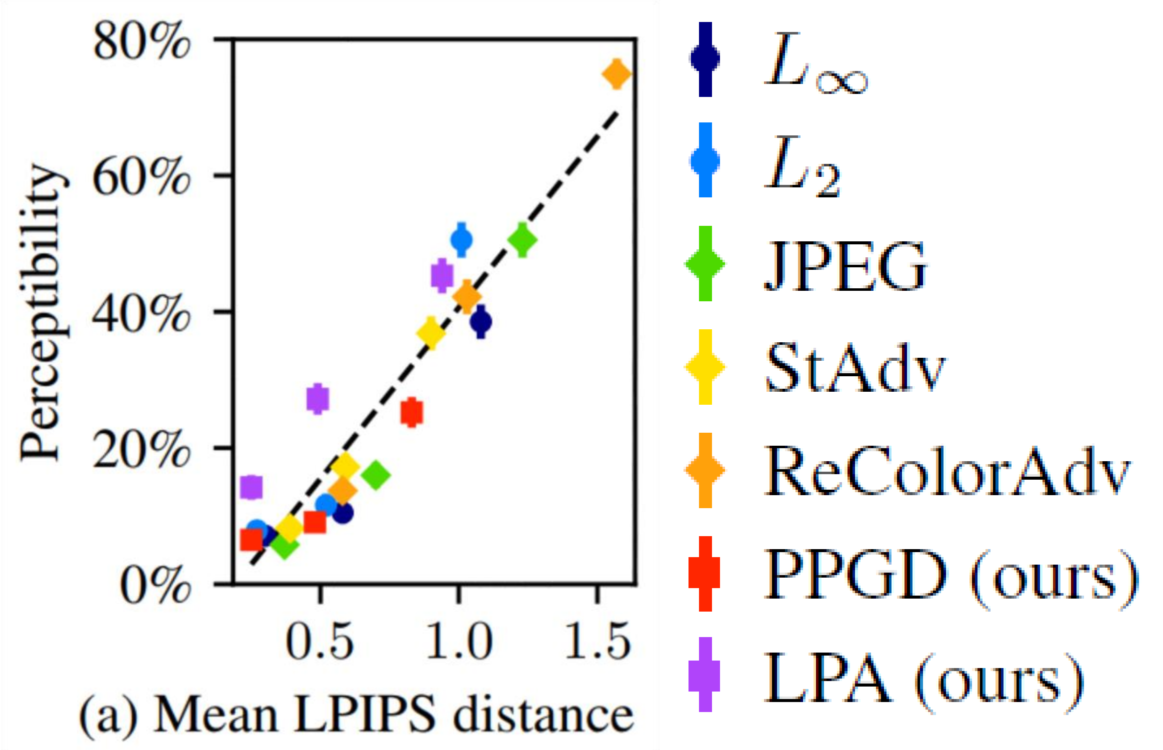
- **Learned Perceptual Image Patch Similarity (LPIPS) distance:**
  - Idea: Similarity between internal activations of a CNN for two images
  - Let g(.) be a **convolutional** image classifier with l **layers**
  - Two steps:
    1. Normalize activations across channel dimension $\Rightarrow \hat{g}_l(x)$
    2. Normalize activations by layer size and flatten into a single vector $\Rightarrow \emptyset(x)$

$$\phi(\mathbf{x}) \triangleq \left(\frac{\hat{g}_1(\mathbf{x})}{\sqrt{w_1 h_1}}, \ldots, \frac{\hat{g}_L(\mathbf{x})}{\sqrt{w_L h_L}} \right)$$

- **Neural perceptual threat model:**
  - All adversarial examples $\tilde{x}$ which cause **misclassification** but are **neuraly imperceptible** different from x, i.e. $\mathrm{d}^*(x, \tilde{x}) = \|\emptyset(x) - \emptyset(\tilde{x})\|_2 \leq \varepsilon^*$.

Neural perceptual threat model (NPTM)





- UTM:
- All adversaries causes misclassification

- LPIPS correlates well with human judgement
  - AlexNet

Perceptual adversarial attacks

- **Neural perceptual adversarial example with a perceptibility bound ε:**

$$f(\widetilde{\mathbf{x}}) \neq y \qquad \text{and} \qquad d(\mathbf{x}, \widetilde{\mathbf{x}}) = \|\phi(\mathbf{x}) - \phi(\widetilde{\mathbf{x}})\|_2 \leq \epsilon.$$

- **Constraint optimization:**

$$\max_{\widetilde{\mathbf{x}}} \quad \mathcal{L}(f(\widetilde{\mathbf{x}}), y) \qquad \text{subject to} \quad d(\mathbf{x}, \widetilde{\mathbf{x}}) = \|\phi(\mathbf{x}) - \phi(\widetilde{\mathbf{x}})\|_2 \leq \epsilon.$$

$$\mathcal{L}(f(\mathbf{x}), y) \triangleq \max_{i \neq y} \left(z_i(\mathbf{x}) - z_y(\mathbf{x})\right),$$

- **Note:**
  - f(.) and g(.) identical ⇒ *self-bounded attack*
  - f(.) and g(.) different ⇒ *externally-bounded attack*

- **Propose two attacks based on this formulation:**
  - Perceptual Projected Gradient Descent (PPGD)
  - Lagrangian Perceptual Attack (LPA)

PPGD attack

- Find a step δ to maximize $L(f(x+\delta), y)$ such that d($x+\delta$,x) = $\|\emptyset(x+\delta) - \emptyset(x)\|_2 \leq \eta$

- Approximate constraint optimization using first-order Taylor as follows:

$$\max_{\delta} \quad \hat{f}(\mathbf{x}) + (\nabla \hat{f})^{\top}\delta \qquad \text{subject to} \quad \|J\delta\|_2 \leq \eta.$$

- η : Step size
- $\hat{f}(x) = L(f(x+\delta), y)$
- J : Jacobian of $\emptyset(x)$
- J^{+} : pseudoinverse of J

- ***Lemma:*** Closed form solution to above is:

$$\delta^* = \eta \frac{(J^{\top}J)^{-1}(\nabla \hat{f})}{\|(J^{+})^{\top}(\nabla \hat{f})\|_2}.$$

- **Problem:**
    - Difficult to efficiently compute.

- **Idea:**
    - Approximately solve for δ^* using the *conjugate gradient method*.

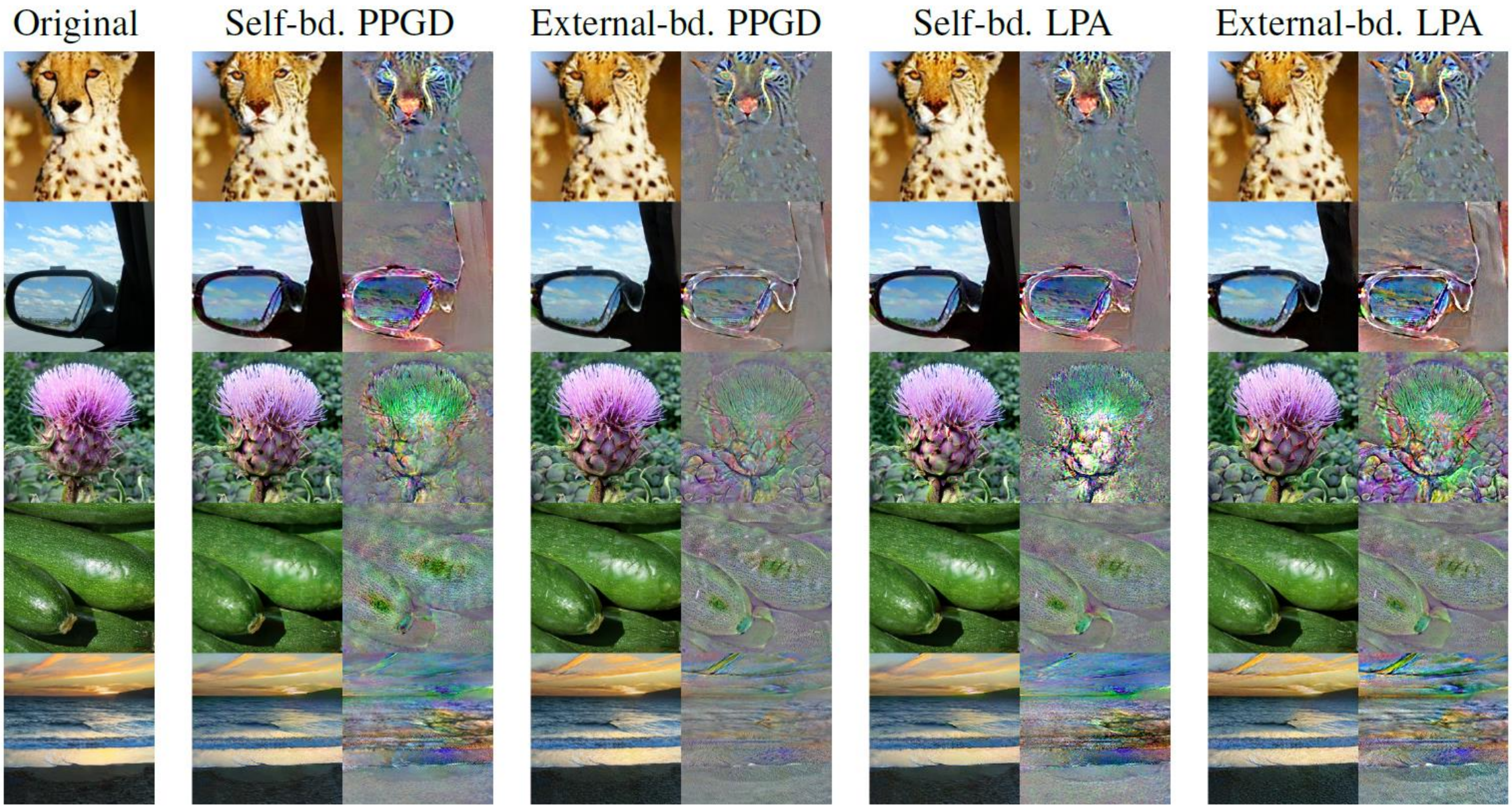
LPA attack

- Lagrangian relaxation of constraint optimization:

$$\max_{\widetilde{\mathbf{x}}} \quad \mathcal{L}(f(\widetilde{\mathbf{x}}), y) - \lambda \max\left(0, \|\phi(\widetilde{\mathbf{x}}) - \phi(\mathbf{x})\|_2 - \epsilon\right).$$

- Similar to C&W attack, adaptively change λ to find an adversarial example within the allowed perceptual distance.

- **Fast-LPA:** Will be used in adversarial training
  - Similar to LPA with two differences:
    1. Does not search for λ values
    2. Remove the projection step


Original
Self-bd. PPGD
External-bd. PPGD
Self-bd. LPA
External-bd. LPA

Perceptual adversarial training (PAT)

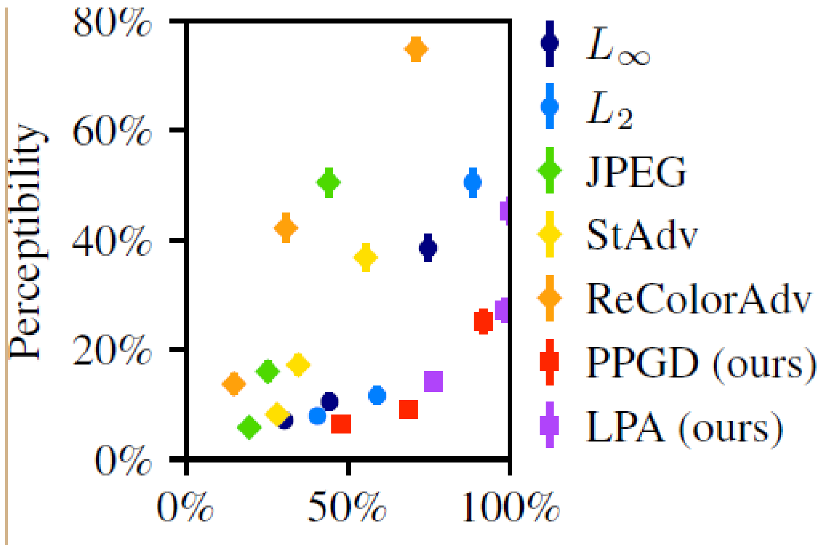
- Minimize the worst-case loss within a neighborhood of each training point x.

$$\min_{\theta_f} \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\max_{d(\widetilde{\mathbf{x}},\mathbf{x})\leq\epsilon} \mathcal{L}_{\mathrm{ce}}(f(\widetilde{\mathbf{x}}), y) \right].$$

- Neighborhood is bounded by the LPIPS distance.

- Dou to intractability of inner maximization:
    - Use Fast-LPA

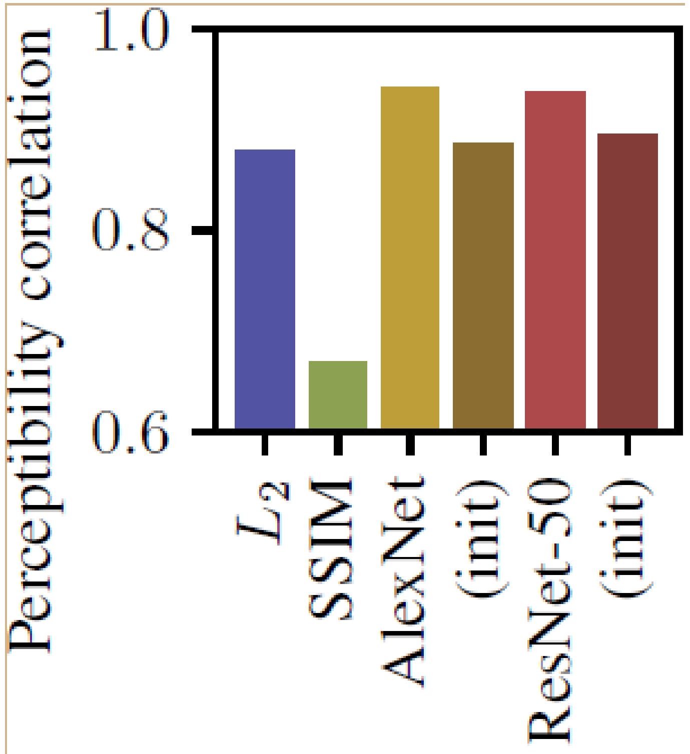
Evaluation



((b) Strength (succ. against AT)

- Perceptibility vs success rate of AT model against different attacks

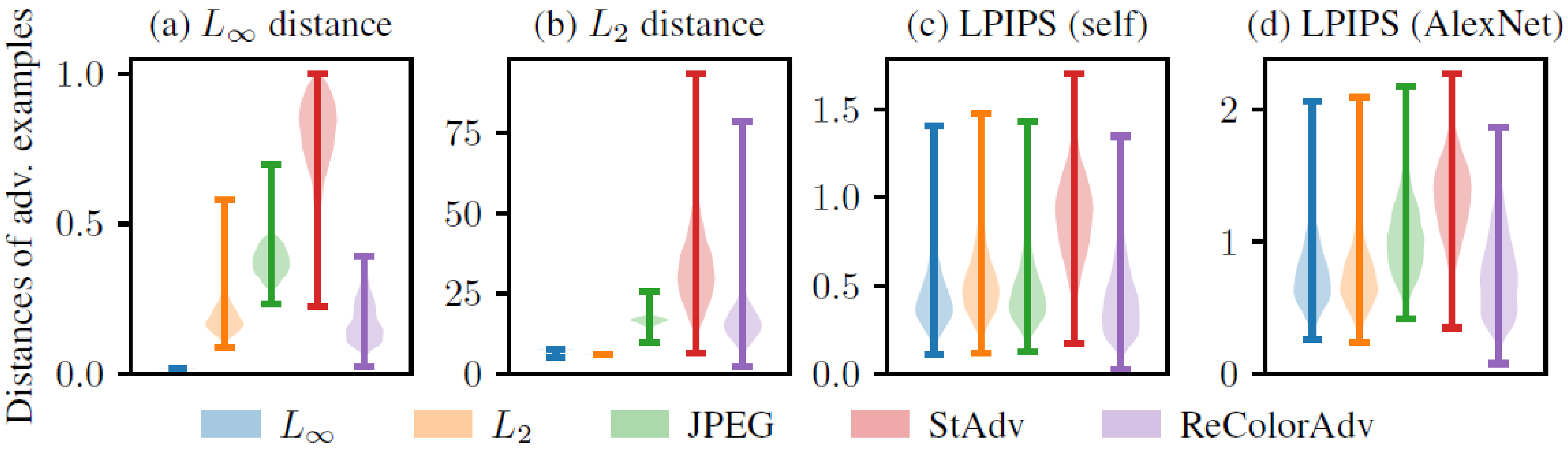
Evaluation



(c) Distance models

- Perceptibility vs distance models

Evaluation



(a) L_∞ distance
(b) L_2 distance
(c) LPIPS (self)
(d) LPIPS (AlexNet)
Distances of adv. examples
1.0
0.5
0.0
75
50
25
0
1.5
1.0
0.5
0.0
2
1
0
L_∞
L_2
JPEG
StAdv
ReColorAdv

Results

- Results on Cifar-10

Training	Union	Unseen mean	Narrow threat models: Clean	L_∞	L_2	StAdv	ReColor	NPTM: PPGD	LPA
Normal	0.0	0.1	94.8	0.0	0.0	0.0	0.4	0.0	0.0
AT L_∞	1.0	19.6	86.8	49.0	19.2	4.8	54.5	1.6	0.0
TRADES L_∞	4.6	23.3	84.9	52.5	23.3	9.2	60.6	2.0	0.0
AT L_2	4.0	25.3	85.0	39.5	47.8	7.8	53.5	6.3	0.3
AT StAdv	0.0	1.4	86.2	0.1	0.2	53.9	5.1	0.0	0.0
AT ReColorAdv	0.0	3.1	93.4	8.5	3.9	0.0	65.0	0.1	0.0
AT all (random)	0.7	—	85.2	22.0	23.4	1.2	46.9	1.8	0.1
AT all (average)	14.7	—	86.8	39.9	39.6	20.3	64.8	10.6	1.1
AT all (maximum)	21.4	—	84.0	25.7	30.5	40.0	63.8	8.6	1.1
Manifold reg.	21.2	36.2	72.1	36.8	43.4	28.4	63.1	8.7	9.1
PAT-self	21.9	45.6	82.4	30.2	34.9	46.4	71.0	13.1	2.1
PAT-AlexNet	**27.8**	**48.5**	71.6	28.7	33.3	64.5	67.5	**26.6**	**9.8**

Results

- Results on ImageNet-100

Training	Union	Unseen mean	Narrow threat models: Clean	L_∞	L_2	JPEG	StAdv	ReColor	NPTM: PPGD	LPA
Normal	0.0	0.1	89.1	0.0	0.0	0.0	0.0	2.4	0.0	0.0
L_∞	0.5	11.3	81.7	55.7	3.7	10.8	4.6	37.5	1.5	0.0
L_2	12.3	31.5	75.3	46.1	41.0	56.6	22.8	31.2	22.0	0.5
JPEG	0.1	7.4	84.8	13.7	1.8	74.8	0.3	21.0	0.5	0.0
StAdv	0.6	2.1	77.1	2.6	1.2	3.7	65.3	2.9	0.6	0.0
ReColorAdv	0.0	0.1	90.1	0.2	0.0	0.1	0.0	69.3	0.0	0.0
All (random)	0.9	—	78.6	38.3	26.4	61.3	1.4	32.5	16.1	0.2
PAT-self	**32.5**	**46.4**	72.6	45.0	37.7	53.0	51.3	45.1	29.2	**2.4**
PAT-AlexNet	25.5	44.7	75.7	46.8	41.0	55.9	39.0	40.8	**31.1**	1.6

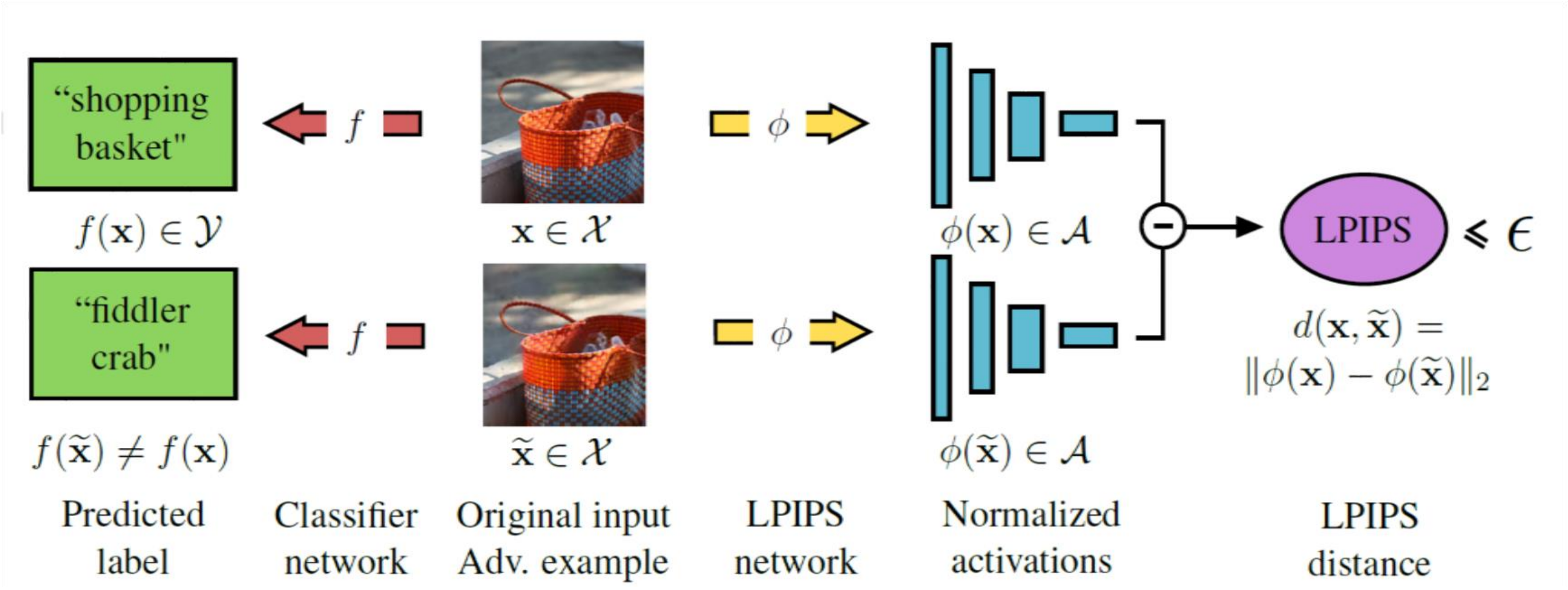
Conclusion

- Proposed NPTM realized by LPIPS distance

- Novel method for developing defenses against adversarial attacks that generalize to unforeseen threat models

- Weaknesses of the paper:
  - Difficult to optimize the LPIPS distance
  - LPIPS is not secure

Shcematic



“shopping basket"
“fiddler crab"
$f(\mathbf{x}) \in \mathcal{Y}$
$f(\tilde{\mathbf{x}}) \neq f(\mathbf{x})$
f
f
$\mathbf{x} \in \mathcal{X}$
$\tilde{\mathbf{x}} \in \mathcal{X}$
ϕ
ϕ
$\phi(\mathbf{x}) \in \mathcal{A}$
$\phi(\tilde{\mathbf{x}}) \in \mathcal{A}$
LPIPS
$\leqslant \epsilon$
$d(\mathbf{x}, \tilde{\mathbf{x}}) =$
$\|\phi(\mathbf{x}) - \phi(\tilde{\mathbf{x}})\|_2$
Predicted label
Classifier network
Original input Adv. example
LPIPS network
Normalized activations
LPIPS distance

References

- Perceptual adversarial robustness: Defense against unseen threat models Laidlaw, S Singla, S Feizi - arXiv preprint arXiv:2006.12655, 2020

- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and RobFergus. Intriguing properties of neural networks. In International Conference on Learning Representations,2014.

- Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In Proceedings of the IEEE International Conference on Computer Vision, pages 1369–1378, 2017.