Mitigating Adversarial Effects Through Randomization

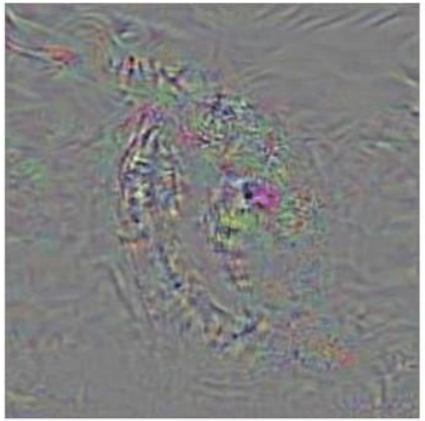
Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, Alan Yuille

paper: Cihang Xie, Jianyu Wang, Zhishual Zhang, Zhou Ren, Alan Yuille, Mitigating adversarial effects through randomization, *arXiv, 2017.*
code: https://github.com/cihangxie/NIPS2017_adv_challenge_defense

What is adversarial examples? (I)

king penguin

+



adversarial perturbation

=

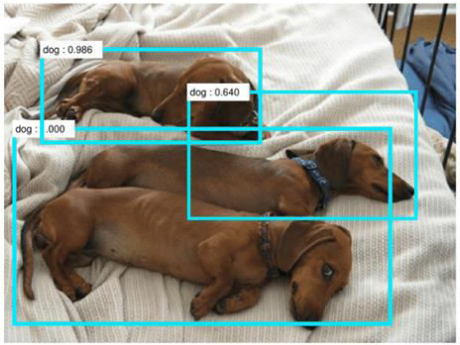
chihuahua

This figure shows the adversarial example in image classification

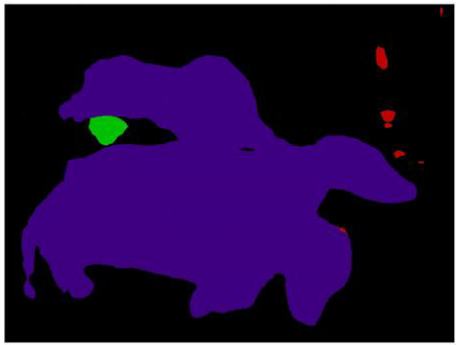
What is adversarial examples? (II)

Original Image

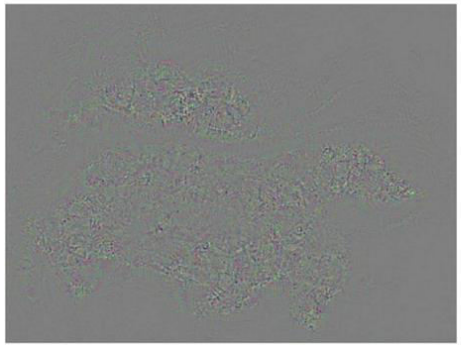
Original Image Detection



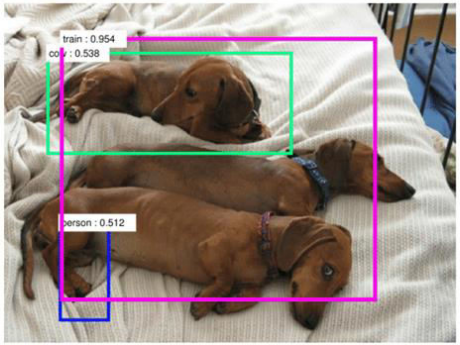
Original Image Segmentation



Adversarial Perturbation



Adversarial Image Detection



Adversarial Image Segmentation

This figure shows the adversarial example in object detection and semantic segmentation. For segmentation results, ***purple indicate dog, light green indicate train, green indicate sofa, pink indicate person***

Formulation of adversarial attacks

Let x denote the input image;

Let f denote the a classifier, e.g., a neural network;

Let l denote the adversarial label, i.e., $f(x) \neq l$

To find the adversarial perturbation r, we can solve the following problem

$$\textbf{min}\ c||r|| + \text{loss}(f(x+r), l)$$

$$\textbf{s.t.}\ \ x+r \in [0, 255]$$

where c is the parameter to control the importance of magnitude of adversarial perturbation, and ||•|| can be an arbitrary norm.

Terminology in adversarial attacks

- **Attack Rate**: the ratio of the number of adversarial examples that let the classifier fail over the total number of adversarial examples
- **Single-Step Attack**: performs only one iteration over the loss to generate adversarial examples
- **Iterative Attack**: performs several iterations over the loss to generate adversarial examples
- **White-Box Attack**: the network structure and parameters are known to the attacker
- **Black-Box Attack**: attacker does not the network parameters or network structures or both of them when performing adversarial attack

Popular Defense Methods

- Adversarial Training [1] / Ensemble Adversarial Training [2]
- Gradient Masking: defensive distillation [3]
- Ensemble Multiple Networks to build a defender

However, these methods can be broken when the network structure and parameters are known to the attackers (i.e., white-box attacks).

[1] Alexey Kurakin, Ian Goodfellow, and Samy Bengio, "Adversarial machine learning at scale", arXiv, 2016.
[2] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel, "Ensemble Adversarial Training: Attacks and Defenses", arXiv, 2017.
[3] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami, "Distillation as a defense to adversarial perturbations against deep neural networks", In IEEE Symposium on Security and Privacy, 2016.

Design Goal

1). hardly hurt the performance on clean image.

2). effective to different attack methods

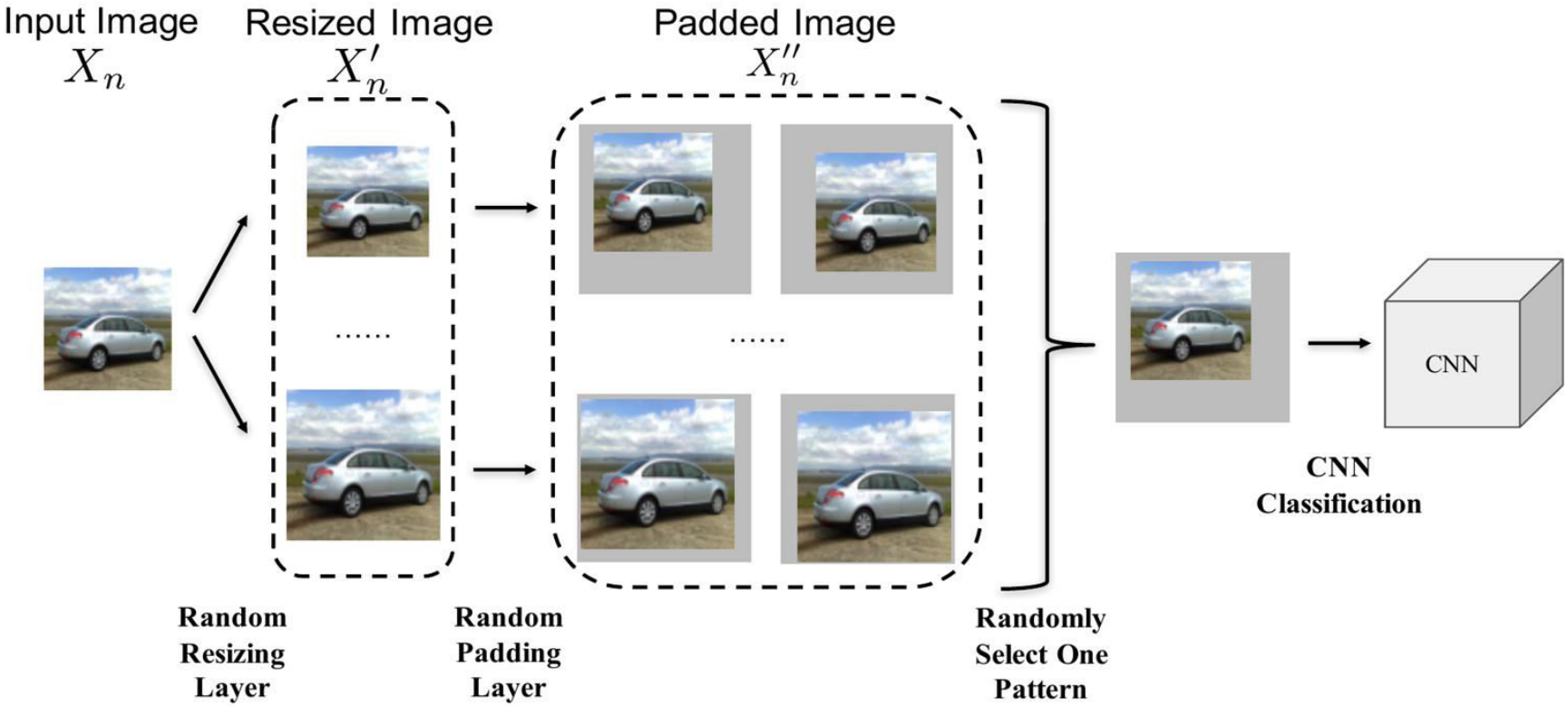
3). simple, e.g., no re-training / fine-tuning.

Our Solution - Randomization

Random Resizing Layer: Resize the original image to a larger size, i.e., to the size of Rnd x Rnd x 3.

Random Padding Layer: Pad the resized image to a new image with fixed size. For example, if we pad the resized image to the size 331 x 331 x 3, then the padding size at left, right, upper, bottom are [a, 331-Rnd-a, b, 331-Rnd-b].

Method Pipeline



Other low-level operations

- Randomly adding small random noise.
- Image filtering: linear or non-linear.
- Image compression: JPEG.

Experiments show little improvement combined with random resizing and padding.

Why it works?

1). No harm on clean images.

- The model trained on large-scale dataset, i.e., Imagenet, is to some extent robust to scale and padding.

2). Break the specific structure of adversarial noise especially for iterative attacks.

- For iterative attacks, the generated adversarial perturbation may be easily overfitted to the network parameter. An image transformation can break the structure.

Extensive Evaluation

Test dataset: 5000 image from Imagenet validation dataset (all classified rightly).

Single-Step Attack: FGSM [4]

Iterative attack: DeepFool [5], C&W [6]

Attack Scenario:

(a) vanilla attack: attackers do not know randomization layers.
(b) single-pattern attack: attackers know randomization layers and choose one specific pattern (resizing and padding) to attack.
(c) ensemble-pattern attack: attackers know randomization layers and choose multiple typical patterns to attack.

[4] Goodfellow I J, Shlens J, Szegedy C. “Explaining and harnessing adversarial examples”. arXiv, 2014.
[5] Moosavi-Dezfooli S M, Fawzi A, Frossard P. “Deepfool: a simple and accurate method to fool deep neural networks”. CVPR, 2016.
[6] Carlini N, Wagner D. “Towards evaluating the robustness of neural networks”. arXiv, 2016.

Top-1 accuracy under Vanilla attack

Models	Inception-v3		ResNet-v2-101		Inception-ResNet-v2		ens-adv-Inception-ResNet-v2	
	target model	defense model	target model	defense model	target model	defense model	target model	defense model
FGSM-2	33.2%	65.1%	26.3%	71.8%	65.3%	81.0%	84.4%	95.7%
FGSM-5	31.1%	54.5%	20.4%	54.3%	61.7%	74.1%	87.4%	94.5%
FGSM-10	33.0%	52.4%	20.4%	46.1%	61.2%	71.3%	90.2%	94.3%
DeepFool	0%	98.3%	0%	97.7%	0%	98.2%	0.2%	99.1%
C&W	0%	96.9%	0%	97.1%	0.3%	97.7%	0.9%	98.8%

Top-1 accuracy under Single-pattern attack

Models	Inception-v3		ResNet-v2-101		Inception-ResNet-v2		ens-adv-Inception-ResNet-v2	
	target model	defense model	target model	defense model	target model	defense model	target model	defense model
FGSM-2	35.1%	63.8%	29.5%	70.1%	71.6%	83.4%	86.3%	96.4%
FGSM-5	32.4%	53.9%	23.2%	52.3%	68.3%	78.2%	88.4%	95.4%
FGSM-10	34.7%	51.8%	22.4%	43.8%	66.8%	75.6%	90.7%	95.2%
DeepFool	1.1%	98.2%	1.7%	97.8%	0.6%	98.4%	1.0%	99.2%
C&W	1.1%	97.4%	1.7%	97.0%	0.8%	97.9%	1.6%	99.1%

Top-1 accuracy under Ensemble-pattern attack

Models	Inception-v3		ResNet-v2-101		Inception-ResNet-v2		ens-adv-Inception-ResNet-v2	
	target model	defense model	target model	defense model	target model	defense model	target model	defense model
FGSM-2	37.3%	41.2%	39.2%	44.9%	71.5%	74.3%	86.2%	88.9%
FGSM-5	31.7%	34.0%	24.6%	29.7%	65.2%	67.3%	85.8%	87.5%
FGSM-10	30.4%	32.8%	18.6%	21.7%	62.9%	64.5%	86.6%	87.9%
DeepFool	0.6%	81.3%	0.9%	80.5%	0.9%	69.4%	1.6%	93.5%
C&W	0.6%	62.9%	1.0%	74.3%	1.6%	68.3%	5.8%	86.1%

Top-1 accuracy under One-pixel Padding

Images are of size 330x330x3, our defense pad them to 331x331x3.
Possible patterns is 4, choose 3 to attack, and test on the remaining one

Models	Inception-v3		ResNet-v2-101		Inception-ResNet-v2		ens-adv-Inception-ResNet-v2	
	target model	defense model	target model	defense model	target model	defense model	target model	defense model
FGSM-2	36.4%	39.6%	29.8%	34.4%	71.3%	74.0%	88.2%	94.8%
FGSM-5	33.5%	36.2%	22.2%	26.2%	68.4%	71.0%	92.1%	94.4%
FGSM-10	34.5%	38.8%	21.3%	23.6%	67.4%	70.4%	93.7%	94.0%
DeepFool	0.9%	97.2%	0.9%	95.2%	0.9%	87.6%	1.5%	99.2%
C&W	0.8%	70.2%	0.9%	76.8%	1.0%	79.4%	2.4%	98.2%

Adversarial examples generated on one specific padding pattern is hard to transfer to a different padding pattern

Top-1 accuracy under One-pixel Resizing

Images are of size 330x330x3, our defense resize them to 331x331x3

Models	Inception-v3		ResNet-v2-101		Inception-ResNet-v2		ens-adv-Inception-ResNet-v2	
	target model	defense model	target model	defense model	target model	defense model	target model	defense model
FGSM-2	30.8%	56.2%	31.6%	44.6%	66.2%	75.0%	87.6%	97.2%
FGSM-5	31.2%	48.8%	25.6%	35.8%	61.4%	70.2%	91.2%	96.6%
FGSM-10	36.4%	51.0%	23.8%	32.6%	62.8%	68.2%	94.8%	95.2%
DeepFool	2.6%	99.4%	1.0%	98.6%	1.2%	97.4%	1.2%	99.4%
C&W	2.6%	97.8%	1.0%	94.8%	2.0%	94.8%	1.8%	99.6%

Adversarial examples generated on one specific resizing pattern is hard to transfer to a different resizing pattern

The Kaggle Submission

Base Model: ens-adv-Inception-Resnet-v2

(*This model is publicly available and almost all top attack teams consider this model in their attacks, thus we think we are doing defense under white-box attack*)

Randomization Parameter: (1) resizing the image to [310, 331)

(2) flipped the input image with p=0.5

(3) 30 randomization patterns are ensembled for the final prediction

Results: normalized score is 0.92, which is far better than using ens-adv-Inception-Resnet-v2 alone with score of 0.77.

Thanks!