African Masters Of Machine Intelligence
AMMI

Explaining and harnessing adversarial examples

autors: Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy

Presented by

ENGELBERT TCHINDE
ewamba@aimsammi.org

Supervised by

Tutors

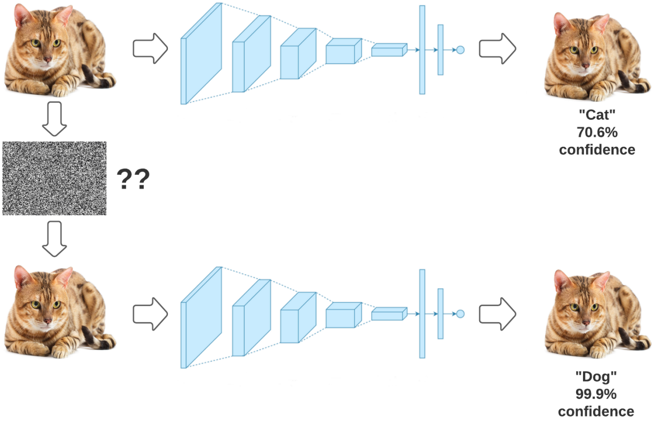
August 27, 2021

Overview

Motivation

- Models are not learning the true underlying properties of the data.
- The cause of adversarial examples is a mystery.
  - Extreme nonlinearity of deep neural networks
  - Insufficient model averaging
  - Insufficient regularization

Adversarial example



Adversarial example for linear model

- In many problems, the precision of an individual input feature is limited.
- Consider a linear model that take an input image $\mathbf{x}$.
- Image: 8 bits for pixel value. 256 pixel values (0-255).
- Perturbation η of each pixel of the image $\mathbf{x}$.
  - $\tilde{\mathbf{x}} = \mathbf{x} + \eta$,
  - $||\eta||_\infty < \epsilon$.
- Why minimization?
  - If we did not minimize η, we could just get a feature vector $\mathbf{x}$.
  - We want to keep the semantic (meaning) of the initial picture by applying so many noise.
- By considering the dot product between a weight vector w and an adversarial example $\tilde{\mathbf{x}}$,

$$\mathbf{w}^T\tilde{\mathbf{x}} = \mathbf{w}^T\mathbf{x} + \mathbf{w}^T\eta.$$

Adversarial example for linear model

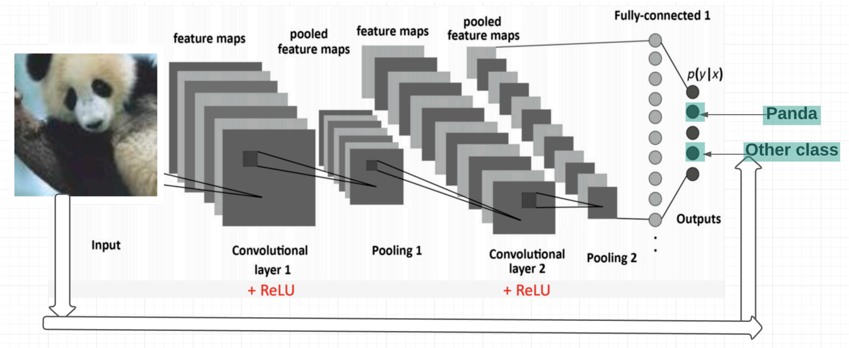
- Why we control $\mathbf{w}^T\eta$?
  - The adversarial perturbation causes the activation to grow by $\mathbf{w}^T\eta$.
- We define the perturbation to be the sign of the weight vector $\mathbf{w}$. Why?
- We are trying to maximize this dot product $\mathbf{w}^T\eta$ s.t. $||\eta||_\infty < \epsilon$
- The activation will grow by ϵmn,
  - $n = dim\, \mathbf{w}$
  - $m =$ average magnitude of an element of $\mathbf{w}$.
- *A simple linear model can have adversarial examples if its input has sufficient dimensionality.*

Linear perturbation for non-linear model

$$\boldsymbol{x}$$

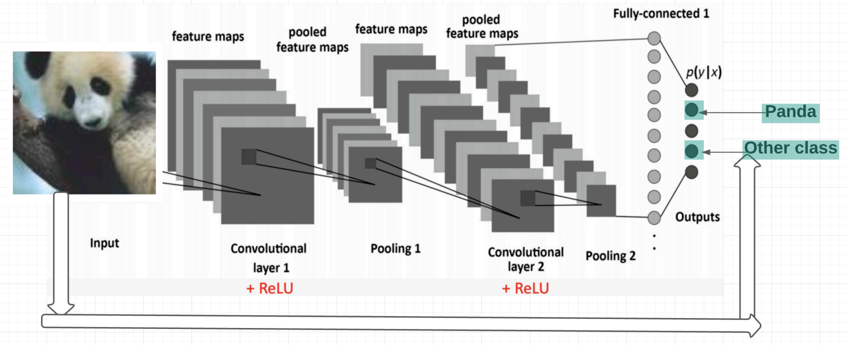
- Select a random real world image **x**: **panda.**

Linear perturbation for non-linear model



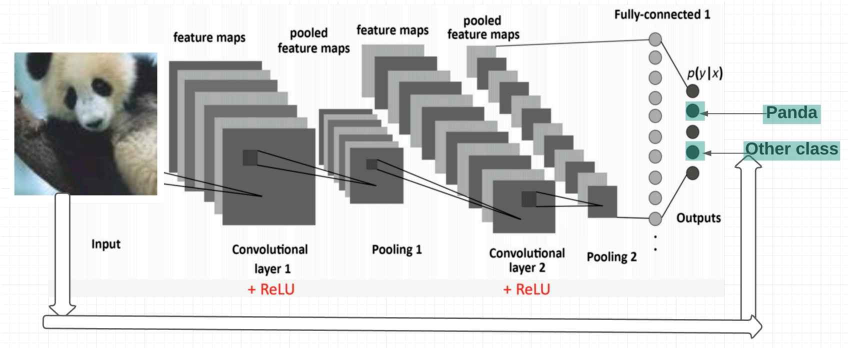
- Run the input image **x** into a ConvNet and get a correct classifier as a **panda**.
- Select a random output neuron in the output layer that is different from the true neuron that classifies **panda**.

Linear perturbation for non-linear model



- We apply **GD** to the input pixels of our **panda** in order to minimize the classification loss with respect to the newly neuron chosen class neuron.
- Instead of adjusting the network weights in order to optimize our classifier, we ajust the input pixels until they fool the network to make a wrong prediction.

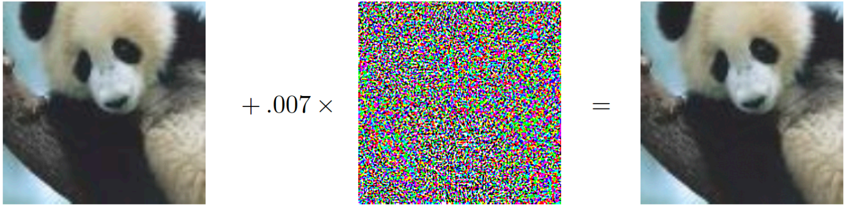
Linear perturbation for non-linear model



- The final trick is to make sure that our generated image looks as close as possible to the original one such that we can't see the difference.

Linear perturbation for non-linear model

- We can generate the perturbation η using a Fast Gradient Sign Methode (FGSM),
  - $\eta = \epsilon * sign(\nabla_x J(\theta, \mathbf{x}, y))$,
  - θ the parameter, **x** input vector, y target associated to **x**, J cost function.



$\boldsymbol{x}$ "panda" 57.7% confidence

$+ .007 \times$ $\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ "nematode" 8.2% confidence

$=$ $\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ "gibbon" 99.3 % confidence

Experiments

From paper:

epsilon	Error rate	Confidence	Activation	Dataset
0.25	99.9%	79.3%	Softmax	MNIST
0.25	89.4%	97.6%	Maxout	MNIST
0.10	87.15%	96.6%	Maxout	CIFAR-10

Our experiments:

epsilon	Error rate	Confidence	Activation	Dataset
0.25	67.33%	0.29%	LogSoftmax	MNIST
0.1 0	15.36%	0.39%	LogSoftmax	MNIST

epsilon	Error rate	Confidence	Activation	Dataset
0.25	63.79%	99.78%	Softmax	MNIST
0.1 0	14.07%	99.64%	Softmax	MNIST

End

THANK YOU !