Defenses against Adversarial Examples

Keller Jordan[1], Rene Gutierrez[2], Brett Gohre[3]

[1]Department of Computer Science
UCSC

[2]Department of Applied Mathematics & Statistics
UCSC

[3]Department of Physical & Biological Sciences
UCSC

CMPS290, Winter 2018

Motivation

Neural Networks at risk of being fooled

- Types of attacks:
  - White box Access to Neural Network model and weight configuration
  - Black box Only know the task of targeted Neural Network, archictecture and current weights unknown
- Alarming strategy:
  - Learn attacks on white box model trained on similar data to black box target
  - Attacks on white box carry over to black box, especially when ensembled (ref. Bo Li's slides)

Motivation

Attempts at defending

- Only good until attacker learns about them (ref. Bo Li's slides)
- Motivates attempt to use random preprocessing of input images
- Does being more data-efficient make attacks harder? CapsNet?
- Reflecting on reconstructed image from adversarial image could be sign of being fooled

Random Projection Matrix

Key Idea

Using a Random Projection Matrix unknown to the Attacker to modify the attack perturbation to defend against it.

Detection of Adversarial Examples using Reconstruction Error

Key Idea

The training loss function is augmented with L2 distance between original image and a reconstruction from the final output of a capsule network. The capsule network output is useful here since it outputs a vector that has more information about the input than a single scalar quantifying the existence of the class.

Multi-task training as Defense

Key Idea

MNIST and FashionMNIST have very different low-level features. Does joint training on FashionMNIST act as a regularizor that strengthens the MNIST classification against attacks? (Results will be featured in write-up).

Idea

- Some Black-Box Adversarial Attacks rely on adding noise-like small perturbations ϵ to an Input x.
- Can we transform the noise into a less powerful attack using a Random Projection Matrix Φ unknown to the attacker. That is, can we go from:

$$x + \epsilon \to \Phi x + \Phi \epsilon$$

- Does $\Phi\epsilon$ retains it's attacking power?

Method

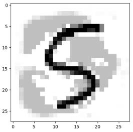
- We worked on the MNIST data set.
- We generated 2 sets of Adversarial Attacks based on 2 different architectures, Logistic Regression and Convolutional Neural Networks.
- In order to generate the Attacks, we added noise-like small perturbations using a trained network and then optimizing for misclassification, while trying to reduce the size of the perturbation.

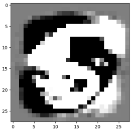
Method

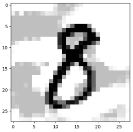
- The first set was created of a Logistic Regression architecture, consisting in the best attacks using as inputs the tests sets, obtaining 758 such attacks.
- The second set was similarly created on a CNN trained to achieve 95% accuracy on the validation set, obtaining 440 attacks.
- We tested 2 strategies, one Compression Matrix (from 784 to 196) and one Expansion Matrix (from 784 to 1960).

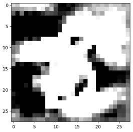
Methods

Figure: Examples of the Adversarial Attacks created under the 2 Architectures. The left side shows the attacks created with the Logistic Regression, while the right side was created using a CNN.


(a) Adversarial Attack for 3


(b) Noise


(c) Adversarial Attack for 3


(d) Noise

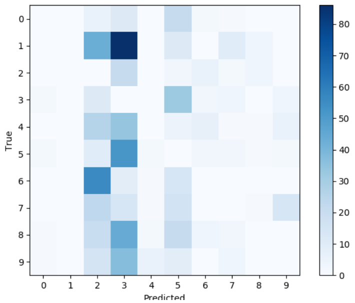
Results

Table: Accuracy for the 5 different methods, using the test set and the adversarial examples generated by Logistic Regression and CNN architectures.

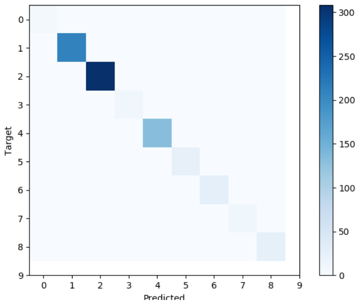
Method	*Test Set*	*Log Adversarial*	*CNN Adversarial*
Logistic Reg	92.00%	0.00%	3.41%
CNN 90%	89.82%	8.71%	5.23%
CNN 95%	94.64%	9.50%	9.50%
CNN 99%	98.63%	60.95%	11.36%
Com Log 196 dim	89.56%	0.00%	5.00%
Exp Log 1960 dim	91.07%	0.00%	3.64%

Results

Figure: Confusion Matrices for the True Target and Predicted Target for the Compression Matrix and Examples generated with the Logistic Regression architecture.



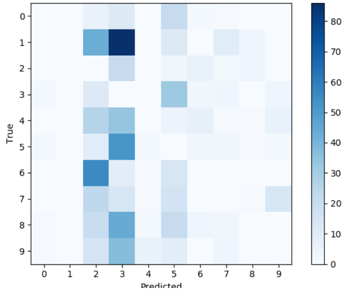
(a) Confusion Matrix for True Class



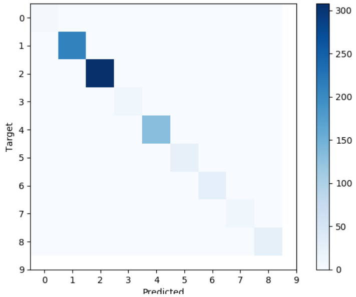
(b) Confusion Matrix for Target Class

Results

Figure: Confusion Matrices for the True Target and Predicted Target for the Expansion Matrix and Examples generated with the Logistic Regression architecture.



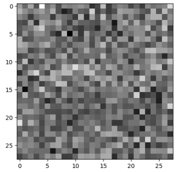
(a) Confusion Matrix for True Class



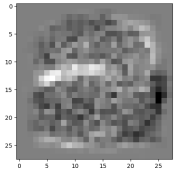
(b) Confusion Matrix for Target Class

Results

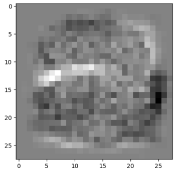
Figure: Weights for 2, for the Compression, Expansion and Regular Logistic Regression.



(a) Compression



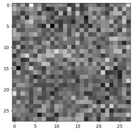
(b) Expansion

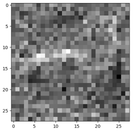


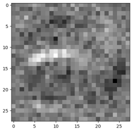
(c) Regular

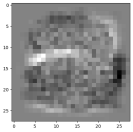
Results

Figure: Weights for 2, for the Compression t0 196, Compression to 392, Compression to 588 and Regular Logistic Regression.


(a) 196


(b) 392


(c) 588


(d) Reg

Optimizer Robustness

Optimizers Studied

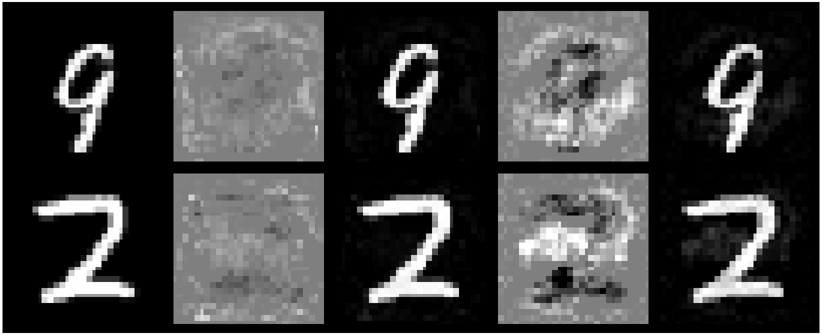
Method	SGD	EG
Rule	$w_{t+1} \doteq w_t - \eta \nabla L(w_t)$	$w_{t+1} \doteq w_t \, e^{-\eta \nabla L(w_t)}$

- Used extension of EG to $+/$- weights case for training

Procedure

- Train FC 784-100-10 using GD and EG$\pm$ on MNIST
- Run untargeted adversarial attack methods
  - Gradient Ascent (GA)
  - Fast Gradient Sign (FGS)
- Compare resulting models and adversarial examples
  - Number of iters to fool
  - Transferability of strong attacks
  - Average perturbation

Examples



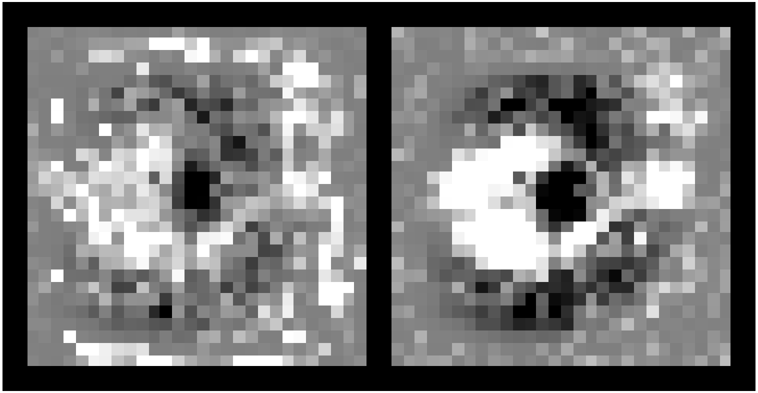
Attack Difficulty

Number of iterations to fool network

Method / Optimizer	SGD	EG
Gradient Ascent	$60.9(\pm 32.3)$	$85.1(\pm 40.5)$
Fast Gradient Sign	$52.0(\pm 26.1)$	$91.0(\pm 43.5)$

- A network is defined as "fooled" when its prediction changes

Average Perturbation



SGD (left), EG± (right)

Transferability Results

Probability of success on other optimizer

Method / Src→Dst	SGD→EG	EG→SGD
Gradient Ascent	67.4%	99.0%
Fast Gradient Sign	88.2%	99.8%

- Iterations held constant at 200 (should be comparably strong)

Results

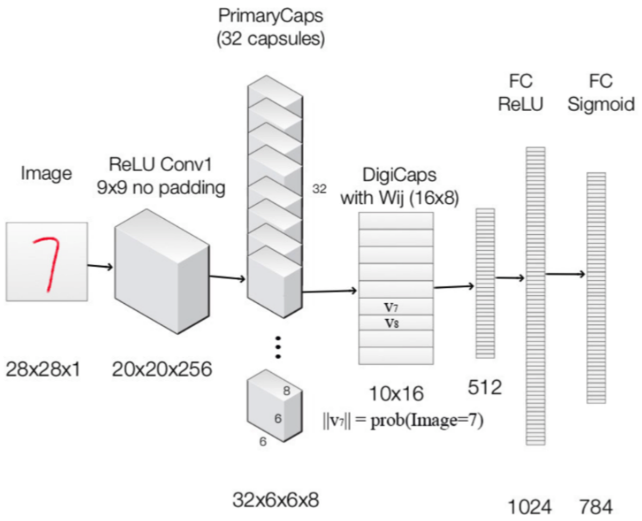
- Requires 1.5$\times$ stronger attacks to fool EG-trained model
- EG shows some robustness to attacks transferred from SGD
- SGD is not robust to attacks transferred to EG
- Attacks against EG make more sense w.r.t. expected structure of digit space

Defending using Reconstruction Error

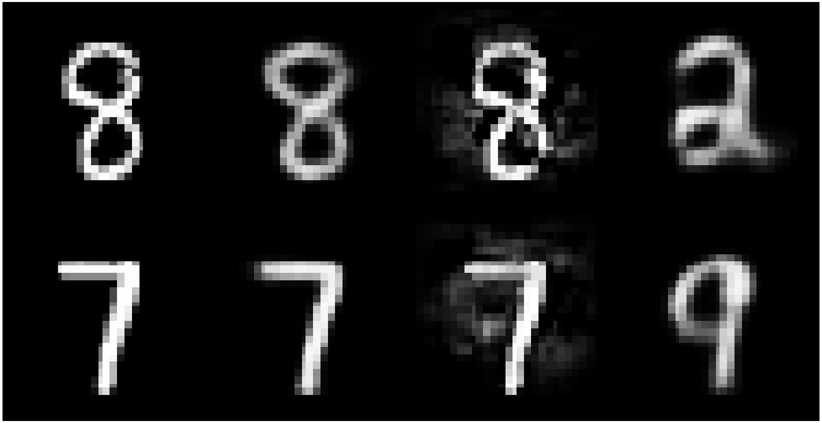
Basic idea

- Use an architecture that reconstructs input images (CapsNet)
- Model will reconstruct some element of decoder-space for fooled class
- Adversarial images are unlikely to be in this space
- Expect high reconstruction error (MSE)

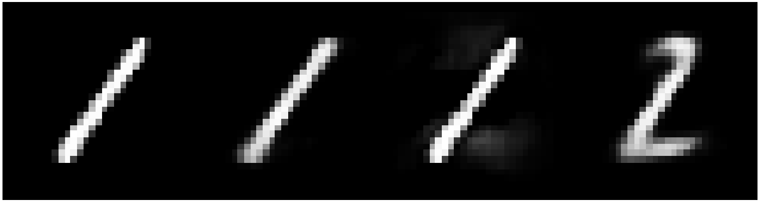
Capsule Network Refresher

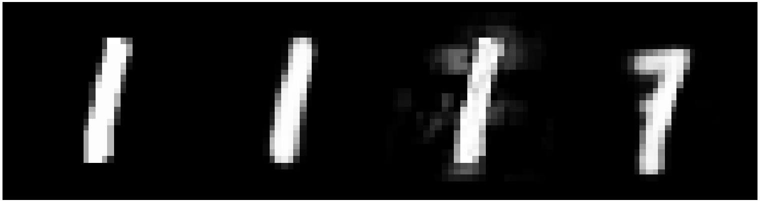


Reconstructions

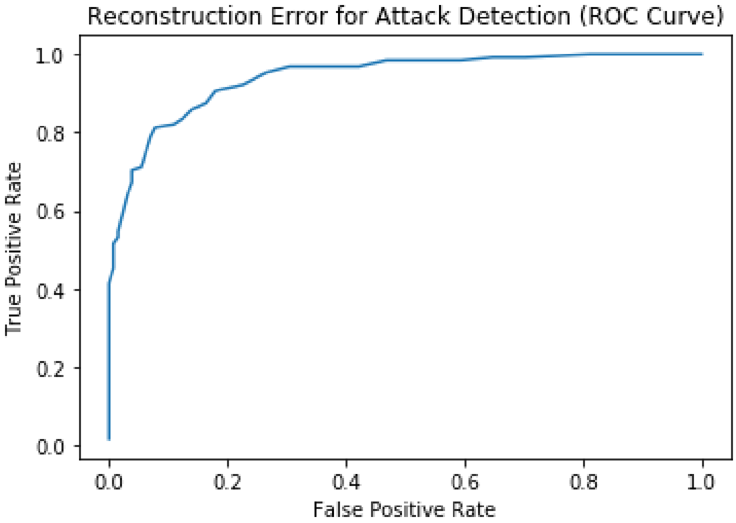


Reconstructions





Results


Reconstruction Error for Attack Detection (ROC Curve)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Results

- Method successfully detects $\sim$70% of of attacks with 5% false-positive rate
- Could be improved by better loss function
- Unknown vulnerability to white-box attacks
- Expect good black-box performance due to variability of decoders and loss functions