

Adversarial Examples for Eye-State Classification

Şefika EFEOĞLU

University of Potsdam

August 1, 2021

Outline

- Introduction
- Problem Setting
- Methodology
- Experiments and Results
- Conclusion

Introduction

- Deep Neural Networks achieve extreme accuracy on image classification tasks
- However, their performances depend on the amount of the training dataset.
- **Approach**: creating adversarial examples of the training dataset to increase the amount of the training dataset.
- **Objective** : Improve the robustness of Eye-State Classifier by adding the Adversarial Examples to the original training data. The adversarial examples are computed from the original training dataset.

Problem Setting

- In the case where a training dataset is small, the network model might not train enough and have a smooth model function.
- This insufficiently trained model does not perform well on the test data.

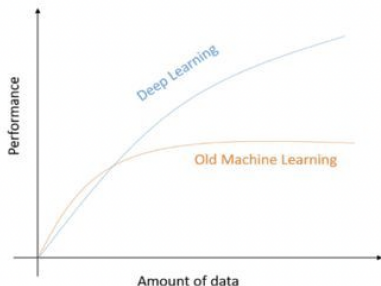


Figure: The performance of a deep learning model in terms of the amount of data ((see the corresponding article ²).

¹<https://www.kdnuggets.com/2021/06/computational-complexity-deep-learning-solution-approaches.html>

Problem Setting (cont.)

- The f model in Equation 1 is not affected by this r when it is smooth enough.
- The f can classify the input $(x + r)$ as l label. In this case, $x + r$ is a variant of x input.
- We try to investigate the usefulness of adding the negative examples like $x + r$ to the training data in order to improve the model's performance in this project.

$$f(x + r) = l \text{ and } f(x) = l \quad (1)$$

where x , r , l and f denote input, perturbation, label and model function, respectively. f has a continuous loss function as well.

Methodology

Eye-State Dataset

Open: 1500

Closed: 1500

Partially Open: 1376

NotVisible: 1346

Statistics about Eye States

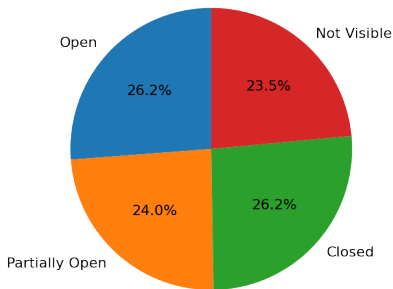


Figure: Eye-State data set consists of 5722 images, and the distribution of each eye state on the histogram.

- ➊ **Neural Network Models:** Residual Networks and Parseval Networks.
- ➋ train the models without Adversarial Examples.
- ➌ computes Adversarial Examples (AEs) by using AEs' **transferability**[1] from the model.
- ➍ Train the models by adding AEs to the original dataset.

Adversarial Examples

- 1 specialised inputs created with the purpose of confusing a neural network
- 2 cause misclassification
- 3 fool the networks identifying a given input.

Types of adversarial attack [2]:

- 1 Blackbox attack
- 2 WhiteBox attack
 - Fast Gradient Sign Method
 - Projected Gradient Descent
 - Deepfool etc..

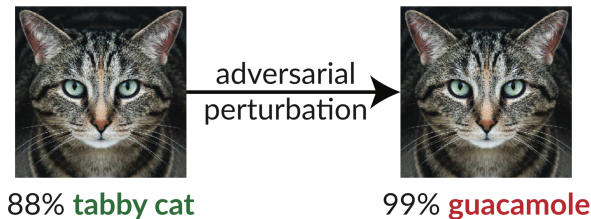


Figure: adversarial example of the cat image

Fast Gradient Sign Method (FGSM)

Definition

$$adv_x = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

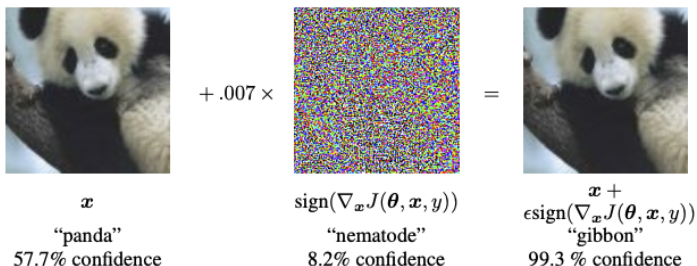


Figure: the example of how adversarial example of an image is obtained using Fast Gradient Sign Method.

Random Noise

The adversarial examples can be obtained by adding the random noise to the images. Adding the random noise is also known as weak attack [3].

Definition

Random Noise:

$$\eta \sim U(-\epsilon, \epsilon)$$

$$adversarial_example = image + \eta$$

The python code:

$$\eta = np.random.uniform(low = -\epsilon, high = \epsilon, size = image_shape)$$

Wide Residual Network[4]

Problem of Deep Neural Networks

- improving accuracy costs is expensive.
- training is a problem of diminishing feature reuse.
- very slow to train.

Wide Residual Network[4]

Problem of Deep Neural Networks

- improving accuracy costs is expensive.
- training is a problem of diminishing feature reuse.
- very slow to train.

Proposed solutions

- decrease depth.
- increase width of residual networks (Wide ResNet).

Neural Network Structure of the Wide ResNet

Table: shows the structure of Wide Residual Blocks. k and N denote width and depth factors used in Wide Residual Network, respectively.

group name	output size	block type = B(3,3)
conv1	32×32	$[3 \times 3, 16]$
conv2	32×32	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$

Parseval Networks

Objectives

Using the advantages of orthogonality and convexity constraints, improve the accuracy of the deep neural networks [5].

Additionally, it provides faster converges on learning curves.

Parseval Networks

Objectives

Using the advantages of orthogonality and convexity constraints, improve the accuracy of the deep neural networks [5].

Additionally, it provides faster converges on learning curves.

Two constraints below and parseval training are considered:

- Orthogonality constraint in linear/conv layer.
- Convexity constraint in aggregation layer.
- Parseval Training.

Wide Residual Network vs Parseval Network

Properties Name	Wide Residual Networks	Parseval Networks
Kernel Initializer	Gaussian	Orthogonal
Orthogonality Constraint	X	✓
Convexity constraint	X	✓

Table: shows that the properties of two different networks

Experiments and Results

Road Map

- Model Selection
- Training of Residual Networks without Adversarial Examples
- Training of Parseval Networks
- Training with Adversarial Examples
- Testing on the original dataset.

Model Selection (1)

Learning Rate (LR): 0.1, 0.01

Regularization Penalty (WD): 0.01, 0.001, 0.0001

Batch Size (BS): 64, 128, 256

Epoch: 50, 100, 150

Table: gives the hyperparameter tuning results.

Model Number	LR	# of Epoch	BS	WD	Loss (Avg)	Accuracy (Avg)
1	0.10	50.0	64.0	0.0001	0.874353	0.707679
2	0.10	100.0	64.0	0.0001	0.899651	0.732461
3	0.01	150.0	64.0	0.0001	0.937435	0.682373

Table: shows the effect of the width factor on the model's performance.

mean loss	mean acc	width factor (k)
0.874353	0.707679	1.0
1.088562	0.706981	2.0
1.253880	0.734729	4.0

Model Selection (2)

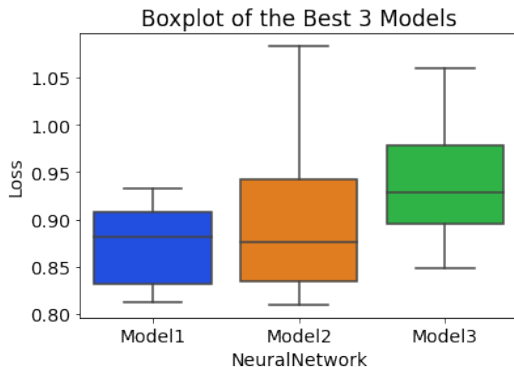


Figure: Parameter Selection with 10 Fold Cross Validation on ResNet16 architecture.

Learning Curves of the ResNet (Model1)

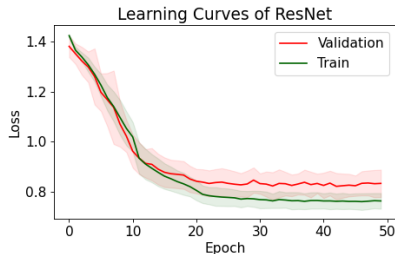
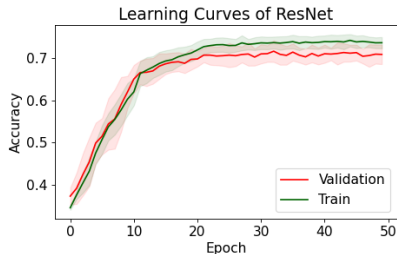


Figure: The learning curves of the optimized model with ten-fold cross-validation.

Training of Parseval Networks

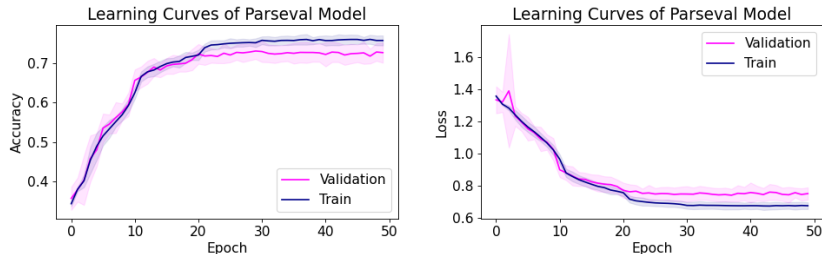


Figure: The learning curves of the Parseval version of the ResNet model.

Training of the ResNet with adversarial examples (FGSM)

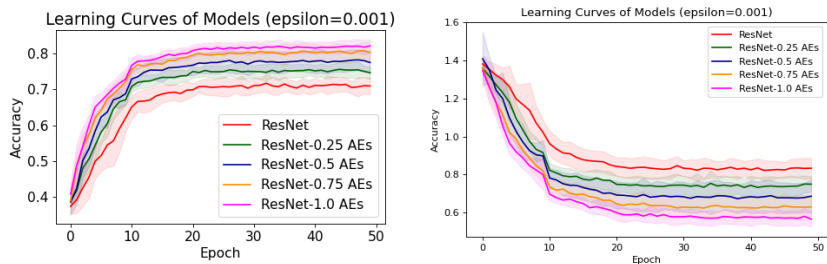


Figure: The models are trained by adding the Adversarial Examples (AEs) to the original training dataset. The AEs are computer by the fast gradient sign method (FGSM).

Training of the ResNet with adversarial examples (Random Noise)

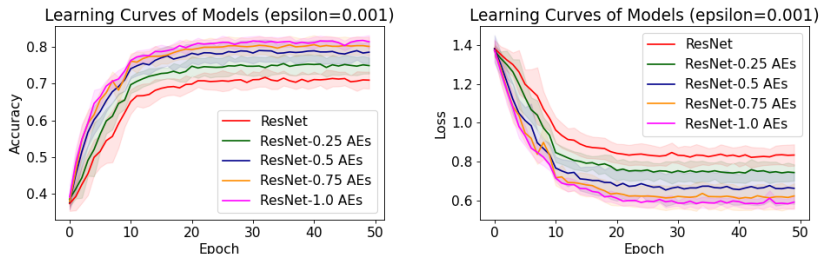


Figure: The models are trained by adding the Adversarial Examples (AEs) to the original training dataset. The AEs are computer by adding random noise to the original images.

Results of the models on the original test data-1 (FGSM)

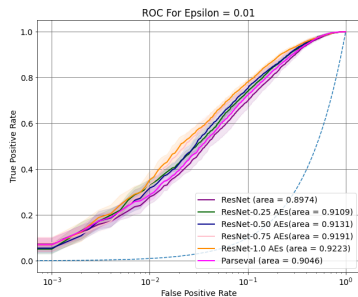
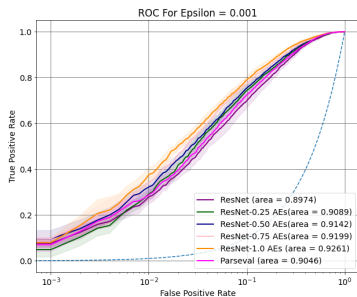


Figure: The Adversarial Examples (AEs) are computed in the figure on the left side with 0.001 epsilon value. The figure on the right side illustrates the ROC of the models when the epsilon value of the AEs is 0.01. The AEs are constructed by the fast gradient sign method.

Results of the models on the original test data-2 (Random Noise)

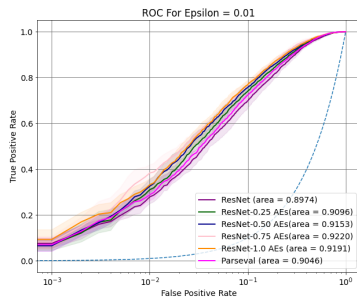
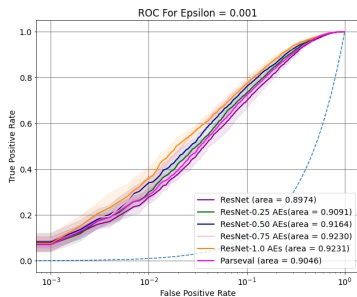


Figure: The ROC of the model trained by adding Adversarial Examples (AEs) which are computed by using random noise.

Results of the models on the original test data-3 (FGSM)

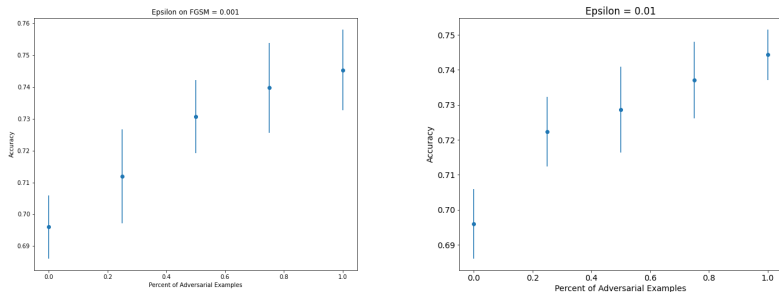


Figure: The figures show the models' performances which are trained by adding Adversarial Examples with different percentages. The figure on the left side show the accuracies when the Adversarial Examples' epsilon value is 0.001, and this epsilon value is 0.01 on the other figure (the right side).

Results of the models on the original test data-4 (FGSM)

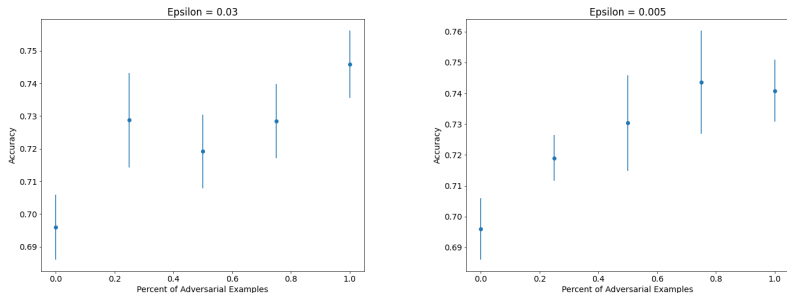


Figure: The models' performance results when epsilon values are 0.003 and 0.005

Results of the models on the original test data-5 (Random Noise)

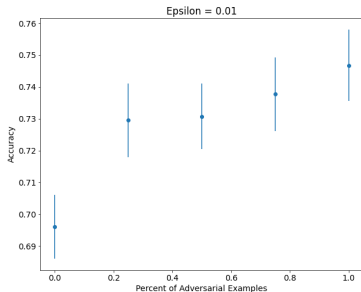
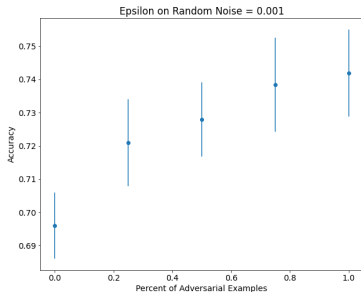


Figure: The performance results of the models when the Adversarial Examples are computed by using random noise.

Results of the models on the original test data-6 (Random Noise)

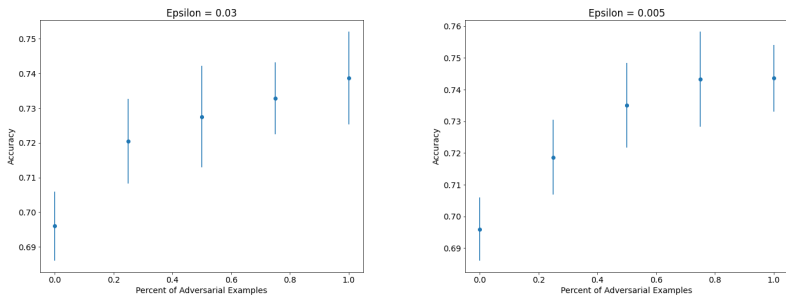


Figure: The performance results of the models when the Adversarial Examples are computed by using random noise.

All Results (FGSM)

Table: The results of Residual Network models, trained adding AEs to the training dataset original test dataset.

epsilon	% of AEs	mean acc	epsilon	% of AEs	mean acc
0.0	0.0	0.696	-	-	-
0.001	0.25	0.711867	0.01	0.25	0.722339
0.001	0.5	0.730716	0.01	0.5	0.728621
0.001	0.75	0.739791	0.01	0.75	0.737173
0.001	1.0	0.745375	0.01	1.0	0.744328
0.003	0.25	0.720593	0.03	0.25	0.728796
0.003	0.5	0.724084	0.03	0.5	0.719197
0.003	0.75	0.735253	0.03	0.75	0.728447
0.003	1.0	0.750785	0.03	1.0	0.745899
0.005	0.25	0.719023	0.005	0.75	0.743630
0.005	0.5	0.730366	0.005	1.0	0.740838

All Results (Random Noise)

Table: shows the accuracies of the models trained with the AEs computed by Random Noise.

epsilon	% of AEs	mean acc	epsilon	% of AEs	mean acc
0.0	0.0	0.696	-	-	-
0.001	0.25	0.720942	0.01	0.25	0.729494
0.001	0.5	0.727923	0.01	0.5	0.730716
0.001	0.75	0.738394	0.01	0.75	0.737696
0.001	1.0	0.741885	0.01	1.0	0.746771
0.003	0.25	0.730017	0.03	0.25	0.720419
0.003	0.5	0.730017	0.03	0.5	0.727574
0.003	0.75	0.742408	0.03	0.75	0.732810
0.003	1.0	0.748517	0.03	1.0	0.738743
0.005	0.25	0.718674	0.005	0.75	0.743281
0.005	0.5	0.735079	0.005	1.0	0.743630

Conclusion






- Basic Residual Network is enough for this classification task.
- Neural Networks can be made smooth by adding adversarial examples to the training dataset.
- Robustness was improved by adding adversarial examples.
- The amount of the adversarial examples in the training dataset does not effect the model's performance adversely.
- However, the epsilon value of the adversarial examples might effect the model's performance.
- Overall, we can confirm that adding adversarial examples to the training data improves the deep learning model's performance.

Repository:

https://github.com/sefeoglu/adversarial_examples_parseval_net

Thank you.

References

-  J. Zhang and C. Li, “Adversarial examples: Opportunities and challenges,” *IEEE Transactions on Neural Networks and Learning Systems*, p. 1–16, 2019. [Online]. Available: <http://dx.doi.org/10.1109/TNNLS.2019.2933524>
-  A. Nazemi and P. Fieguth, “Potential adversarial samples for white-box attacks,” 2019.
-  A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” 2018.
-  S. Zagoruyko and N. Komodakis, “Wide residual networks,” 2017.
-  M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, “Parseval networks: Improving robustness to adversarial examples,” 2017.

Orthogonality Constraint

Definition

- an optimization algorithm on the manifold of orthogonal matrices
- another name is Stiefel Manifold

$$R_{\beta}(W_k) \leftarrow \frac{\beta}{2} \left\| W_k^T W_k - I \right\|_2^2$$

is expensive after each gradient update step.

- after every main gradient update, second update is applied to make the algorithm more efficient

Orthogonality Constraint

Definition

- an optimization algorithm on the manifold of orthogonal matrices
- another name is Stiefel Manifold

$$R_{\beta}(W_k) \leftarrow \frac{\beta}{2} \left\| W_k^T W_k - I \right\|_2^2$$

is expensive after each gradient update step.

- after every main gradient update, second update is applied to make the algorithm more efficient

$$W_k \leftarrow (1 + \beta) W_k - \beta W_k W_k^T W_k$$

Convexity Constraint in Aggregation Layer

Definition

- In Parseval Networks, aggregation layers output a convex combination of their inputs.
- To ensure that Lipschitz constant at the node n is such that

$$\Lambda_p^n \leq 1$$

euclidean projection is applied below

$$\alpha^* = \arg \min_{\gamma \in \Delta^{K-1}} \|\alpha - \gamma\|_2^2$$

Parseval Training

Algorithm 1: Parseval Training

```
 $\Theta = \{W_k, \alpha_k\}_{K=1}^{k=1}, e \leftarrow 0$ 
while  $\{e \leq E\}$  do
    Sample a minibatch  $\{(x_i, y_i)\}_{i=1}^B$  .
    for  $k \in \{1, \dots, K\}$  do
        Compute the gradient ;
         $G_{W_k} \leftarrow \nabla_{W_k} l(\Theta, \{(x_i, y_i)\})$ 
         $G_{\alpha_k} \leftarrow \nabla_{\alpha_k} l(\Theta, \{(x_i, y_i)\})$ 
        Update the parameters:
         $W_k \leftarrow W_k - \epsilon \cdot G_{W_k}$ 
         $\alpha_k \leftarrow \alpha_k - \epsilon \cdot G_{\alpha_k}$ 
        if hidden layer then
            Sample a set S of rows of  $W_k$ 
            Projection:
             $W_s \leftarrow (1 + \beta)W_s - \beta W_s W_s^T W_s$  .
             $\alpha_k \leftarrow \operatorname{argmin}_{\gamma \in \Delta^{K-1}} \|\alpha_{K-\gamma}\|_2^2$ 
        end
    end
     $e \leftarrow e + 1$ 
end
```
