Detection and Properties of Adversarial Noise

Sharon Qian
Harvard University
sharonqian@g.harvard.edu

Irina Tolkova
Harvard University
itolkova@g.harvard.edu

Abstract

While machine learning algorithms have shown impressive results, there have been a large number of publications highlighting the lack of robustness in these models. Specifically, various adversarial algorithms have been created to apply small pixel perturbations to images to fool classifiers. This has begun a cycle of new adversarial noise generators and new techniques to protect against adversarial data. Through an experiment with multilayer perceptrons over the MNIST dataset, we show that we can train a classifier to detect adversarially corrupted samples generated by three different algorithms with high accuracy. We also argue that given an adversarial algorithm, we can learn interesting properties of the decision boundaries of the original classifier.

1 Introduction

Recently, there has been a great deal of research on generating adversarial images to fool machine learning algorithms, which has highlighted the lack of robustness in neural networks [12, 9, 8, 11]. In these cases, the classical stationarity assumption of machine learning is violated, causing performance to deteriorate on later examples [2]. Part of the concern is driven by the knowledge that, unlike random noise, adversarial noise has been shown to be damaging even at very low magnitudes (undetectable to the human eye). Our work primarily focuses on identifying images subject to random or adversarial noise, and evaluating the noise budget at which corruption becomes problematic. Additionally, we explore the result of repeatedly adding adversarial noise to the original image and using the adversarial algorithm to provide insight into the decision boundaries of the original classifier.

2 Background

Previous work has shown the robustness of various machine learning algorithms to noisy labeled examples. In [1], it was shown that PAC-learnable algorithms can be generalized to handle a certain amount of random noise. However, even in the presence of random noise, a sample size can be chosen to be sufficient for learning. Previous work has also shown that the noisy version of the perceptron problem (with random label noise) is PAC-learnable [5]. An analogous analysis has been done for the robustness of support vector machines to label noise [3]. Our work touches on PAC-learnability, but focuses on detecting the presence of noise added to the feature space.

While most work in this field has centered on increasing the robustness of classifiers against adversarial noise, there have also been multiple studies aiming to identify adversarial samples in the testing dataset. For instance, [7] study the position of samples relative to the decision boundary, and show that adversarial samples can be detected by using a different measurement of classification uncertainty based on kernel density estimates and Bayesian distance metrics. Some studies protect against adversarial attacks by checking for consistency in classification between a data point and a processed data point; in particular, [10] applies scalar quantization and a spatial smoothing filter to test images, and uses classification confidences for the original and modified samples to build a detection filter.

Finally, there are several recent studies that are closest to our line of work. Metzen et al. (2017) generate adversarial images for the CIFAR (and some of ImageNet) datasets, and combine the original and adversarial images to train a binary “detector" network. They find high classification accuracy for detecting adversarial images. Furthermore, the work shows high transferability of the “detector" across different noise generation algorithms. Finally, the authors also describe a possible counterattack, and a method for defending against the counterattack. Other studies [4] suggest that modified loss functions are sufficient to bypass these defenses.

3 Types of Adversarial Attacks

We compare three different types of adversarial attacks that are based on the gradient of the neural network. By adding a small amount of noise to the original image, these attacks are able to trick the classifier with high confidence.

For the discussion of these algorithms, we consider a set of inputs x and labels y, used to train a classifier $f(x)$ to minimize a cost $J(\theta, x, y)$, where θ denotes the parameters of the classifier.

Box-constrained L-BFGS

In their initial description of adversarial perturbations, Sgezedy et al., posed the problem of constructing an adversarial example from a point x as an optimization problem. As an initial formulation, the authors aimed to minimize the magnitude of a perturbation r, subject to the constraints that the modified point $x + r$ remained a valid image, and was misclassified by the network. While this could be a highly nonconvex problem, the authors approximate the solution with box-constrained L-BFGS, a general optimization algorithm from the family of quasi-Newton solvers. Following this work, this method was rarely used in practice, as it depended on an computationally expensive line search. However, it inspired many new algorithms for generating adversarial examples, which will be discussed below. In our project, we implemented the following three adversarial attacks.

3.1 Untargeted Fast Gradient Sign

In 2014, Goodfellow et al. developed the fast gradient sign method (FGSM) algorithm. The original version of this algorithm consisted of one step of a fixed length ϵ down the gradient of the loss with respect to the original label.

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

This method inspired many variations; for instance, the one-step descent could be done iteratively with a smaller step size until the boundary is crossed.

3.2 Targeted Fast Gradient Sign

Another variation on FGSM, targeted Fast Gradient Sign creates a targeted attack, which allows the adversary to specify the label of the desired misclassification class by replacing the gradient with respect to the original label by the negative gradient with respect to the target label:

$$\eta = -\epsilon \text{sign}(\nabla_x J(\theta, x, y_{target}))$$

Later variants apply momentum to the update step, such that the step is updated with a decaying sum of the previous gradients in addition to the current gradient. In fact, the first-place winner of the NIPS 2017 competition for Targeted and Non-Targeted Adversarial Attacks was an FGSM-based algorithm, which calculated FGSM with momentum across an ensemble of models [6]. These variations show that even conceptually simple attacks can be effective, transferable, and virtually undetectable.

3.3 DeepFool

Proposed by Frossard et al., DeepFool finds the closest distance to the boundary by iteratively linearly approximating the classifier. The update becomes

$$\eta = -f(x)\frac{\nabla f(x)}{||\nabla f(x)||_2^2}$$

This algorithm has been shown to find a smaller perturbation than FGSM, and can be extended to minimize distance metrics based on different p-norms.

Examples of these three adversarial attacks are shown in Fig. 1. All three attacks are able to misclassify the original image with high confidence.

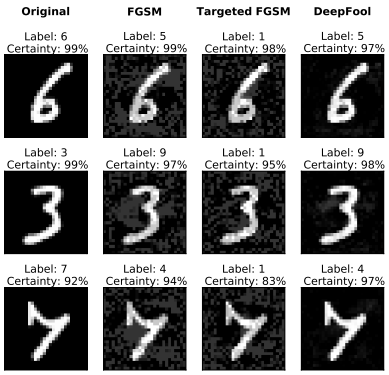

Figure 1: Examples of FGSM, Targeted FGSM (with target = 1), and DeepFool adversarial noise added to MNIST digits.

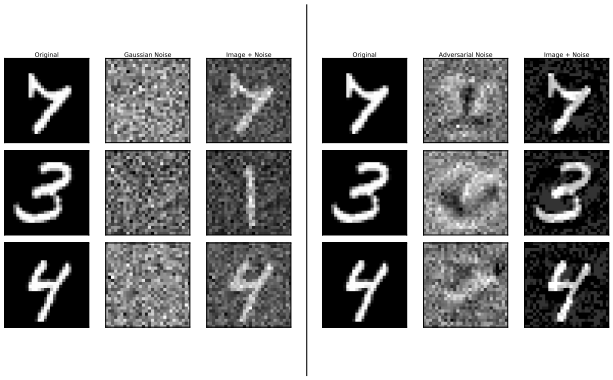

Figure 2: Left: Examples of Gaussian noise added. Right: Examples of adversarial noise added.

4 Adversarial vs. Random

To understand the properties of adversarial attacks, we compared the impact of test-time adversarial and random noise on classification accuracy. For our experiment, we trained a multilayer perceptron (MLP) on the MNIST digit dataset. Then, we added Gaussian noise centered at 0 with a given standard deviation to the test dataset, and measured the classification accuracy. We repeated this measurement for standard deviation varying from 0 to 1. Finally, we generated adversarial test images using one-step FGSM. By varying the step size ϵ, we could measure the classification accuracy across different noise budgets. Only a subset of the test dataset was used in this experiment due to the long runtime. Examples of the Gaussian noise added and adversarial noise added, along with the resulting ”noisy" figures, are shown in Fig 2.

The comparison of classification rates across noise magnitude is shown in Fig 3. From these results, it is clear that adversarial noise is much more damaging than Gaussian noise at equal noise budgets.

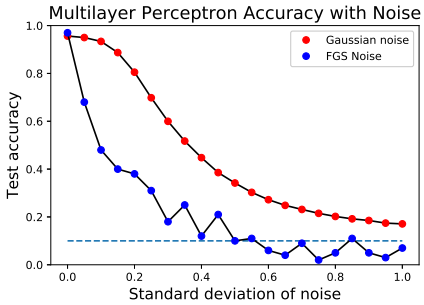

Figure 3: Classification accuracy of MNIST classifier for test data corrupted with Gaussian noise and adversarial noise of varying magnitude.

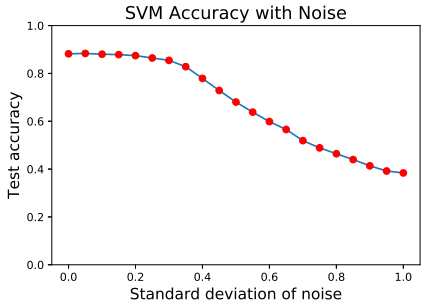

Figure 4: Classification accuracy of SVM for test data corrupted with Gaussian noise of varying magnitude.

Comparison to SVM

We repeated the above Gaussian noise experiment with an SVM. Only 10% of the testing and training dataset was used for computational purposes. Fig 4 shows the classification results across varying noise budgets. The results are similar, though the accuracy of the SVM appears to decay slower with noise magnitude.

5 Detection of Adversarial Noise

Now that we have seen the increased impact of adversarial noise compared to Gaussian noise, we would like to see if we can automatically detect it with a binary classifier. Specifically, after training an MLP on MNIST, we construct a dataset where 50% of the images are corrupted with adversarial noise and the other 50% are the original images. Then, we train a new binary detection classifier to differentiate between clean and adversarial images, and measure its classification accuracy on a similarly constructed test set.

We repeat this experiment for the three adversarial algorithms previously discussed: the Fast Gradient Sign Method, Targeted Fast Gradient Sign, and DeepFool. As a baseline, we repeat the experiment with random (Gaussian) noise of mean and standard deviation equal to the mean and standard deviation of the adversarial noise generated by each of the three algorithms. The results of this experiment are shown in Table 1.

In Table 1, we can see that, consistent with previous experimental results, Deep Fool adds a smaller amount of noise than FGSM. As a result, the equivalent amount of Gaussian noise is difficult for the MLP to detect without going into extensive hyperparameter tuning. For all three algorithms, the detection rate of the adversarial noise is much higher than the detection rate of the corresponding Gaussian noise of the same noise budget.

Table 1: Detection accuracy of various adversarial noise attacks and corresponding detection accuracy of Gaussian noise of the same noise budget.

Adversarial Attack	μ_{noise}	σ^2_{noise}	**Detection Rate of Adversarial**	**Detection Rate of Gaussian**
FGSM	0.0102	0.0531	100%	99.9%
Targeted FGSM	0.00416	0.0622	100%	80%
Deep Fool	0.00207	0.00313	99.0%	57.5%

This trend is also reflected in Fig 5 where the detection rate of FGSM adversarial noise is higher than the detection rate of Gaussian noise for smaller noise budgets.

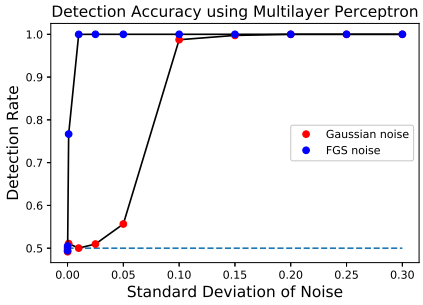

Figure 5: Detection accuracy using MLPs is very effective. For small noise budgets, the detection accuracy of adversarial noise is higher than that of Gaussian noise.

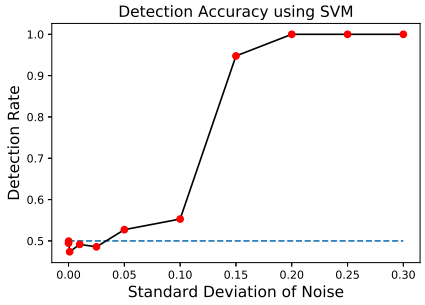

Figure 6: Gaussian noise detection accuracy using SVMs. When $\sigma > 0.15$, the detection accuracy is very high.

Comparison to SVM

We repeat the Gaussian noise experiment with SVMs. To train the SVM in a computationally feasible amount of time, we used 10% of the MNIST training and test sets. The detection rate results for SVMs in Fig 6 show a very similar trend for detection accuracy as the multilayer perceptron in Fig 5. Both models show a large increase in detection accuracy when the noise budget exceeds 0.1 standard deviation. This is interesting because after the noise budget is below a certain threshold, both SVMs and MLPs have difficulty detecting the presence of Gaussian noise.

6 Iterative Adversarial Attacks

To explore the question if adversarial attacks can provide information on the decision boundary of a black box classifier, we use a particular adversarial attack algorithm and run multiple iterations of it on an image.

We run multiple iterations of the untargeted FGSM on the MNIST dataset (formulation in Algorithm 1). After multiple iterations, the resulting image is extremely grainy and indistinguishable to the human eye. However, the neural network is able to classify these images as the incorrect digit with extremely high confidence. This experiment shows that the neural network is extremely sensitive to the adversarial attack and is able to switch its predictions when adversarial noise is repeatedly added.

Overall, we found that iteratively applying Fast Gradient Sign with $\epsilon = 0.04$ resulted in an image that looks similar to white noise (Fig 7). A collection of manually tested examples were shown to be statistically indistinguishable from white noise by the Ljung-Box test, though we did not repeat this analysis on a large scale. However, we found it interesting that, while the classification probabilities resulting from this algorithm varied, they would occasionally result in a very high confidence classification. This suggests the question – can an image of white noise be classified with high confidence?

Algorithm 1: k-Iterative Adversarial Attack

Data: Clean sample $\mathbf{x}_i$ with true label $\mathbf{y}_i$
for $i = 1$ *to* k **do**
  for $j = 1$ *to steps* **do**
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon * \text{sign}\nabla J(\mathbf{x}_i, \mathbf{y}_i)$
  Check $\mathbf{y}_i \neq f(\mathbf{x}_i)$
  $\mathbf{y}_i \leftarrow f(\mathbf{x}_i)$
return $\mathbf{x}_i$

Algorithm 2: Boundary Walk

Data: Clean sample $\mathbf{x}_i$ with true label $\mathbf{y}_i$
for $i = 1$ *to* k **do**
  while $\mathbf{y}_i \neq f(\mathbf{x}_i)$ **do**
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon * \text{sign}\nabla J(\mathbf{x}_i, \mathbf{y}_i)$
  $\mathbf{y}_i \leftarrow f(\mathbf{x}_i)$
return $\mathbf{x}_i$

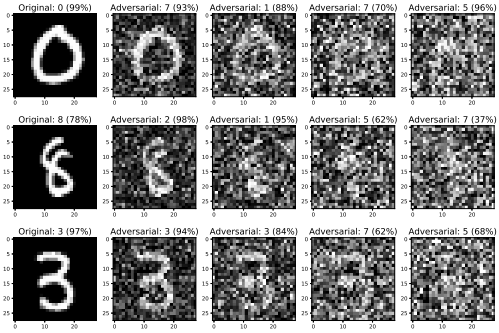

Figure 7: Examples of iterative untargeted adversarial noise being added to MNIST using Algorithm 1. From Left to Right: Original image, every 5th iteration of adversarial noise added.

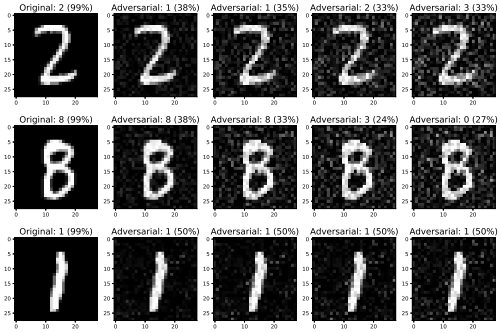

Figure 8: Examples of Algorithm 2 applied to three example images. Images are shown after every 200 iterations of the outer loop.

To verify this, we constructed a large dataset (50000 samples) of “images" of uniform noise, ran the MLP classifier for MNIST, and found the probability associated with the most likely class. The density histogram in Fig 9 shows these probabilities. The histogram peaks at just over 44%

confidence. Notably, we find that over 1.5% of random uniform noise is given a classification with over 97% confidence, and over 6.7% of random uniform noise is given a classification with over 90% confidence. This experiments suggests a lack of robustness of the network to regions of the feature space that are not representative of training samples.

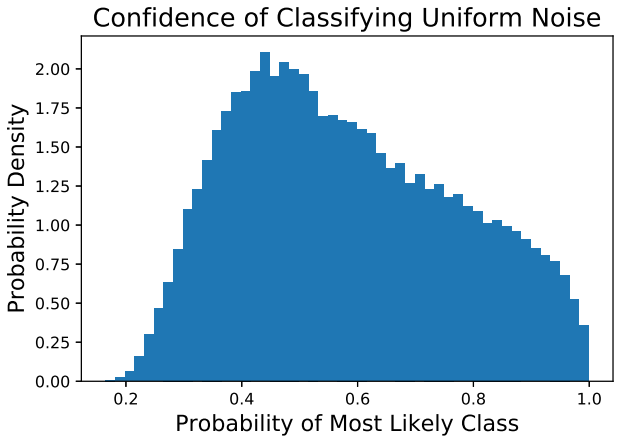

Figure 9: Histogram of highest confidence of classifications of a dataset generated from a random uniform distribution between 0 and 1.

In addition, we experimented with a different variation of the iterative walk algorithm, in which the argument of the gradient was switched as soon as the boundary was crossed (formulation in Algorithm 2). While the previous algorithm took large steps and landed into the interior of classification regions (therefore resulting in a high classification accuracy), the motivation of this experiment was to stay close to the boundary, so the step size was set to $\epsilon = 0.002$. Since this experiment was largely exploratory, we ran it manually with randomly selected initial images. Fig 8 shows the evolution of example images from this algorithm, every 200 outer iterations. In addition, after each iteration of the algorithm (just after a boundary was crossed), we classified the output image with the original classifier. This classification confidence over iterations, colored by class, is shown in Fig 12 for the three example image paths.

Unsurprisingly, the images in Fig 8 do not diverge as quickly into noise as the previous algorithm, as the step size is much lower. However, it is interesting to note the behavior of the classification confidence throughout the outer iterations. In each figure, the confidence falls rapidly over the first few iterations. Then, all examples in Fig 12 show temporary plateaus, during which the classification is limited to a select number of classes. For instance, in the top plot, the plateau occurs around 50% confidence, shared between classes 7 and 0; in the middle plot, the plateau occurs around 33% confidence, shared between classes 3, 2, and 0; and in the bottom plot, the image hovers around 33% (3 classes) before falling to about 25% (4 classes) and then to 20% (5 classes). Within each plateau, the confidences between the classes seem to converge slightly to each other. When examining this algorithm over a greater number of iterations (up to 10000), the final images look like indistinguishable noise (similarly to Algorithm 1) and are classified with very low confidence.

By construction, this algorithm is meant to move along the boundaries of the classifier. We hypothesize that within each plateau, it walks along a high-dimensional boundary shared by a subset of classification regions, but continues to move further from the parts of the feature space corresponding to more meaningful input values. Since there is definite structure to the paths followed by example images, and discrete sections when new class labels occur on the path, this work may be useful for better understanding the boundary structure of networks, and extracting a lower-dimensional representation of the decision boundaries.

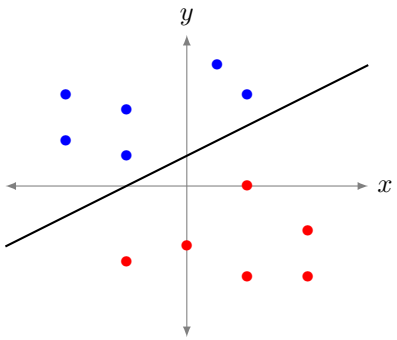

Figure 10: Toy example of binary classifier with linear decision boundary.

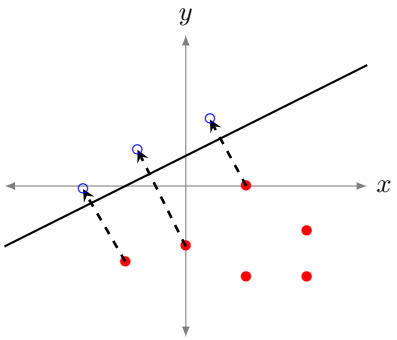

Figure 11: Toy example of adversarial noise being added to one class, pushing the points over the decision boundary.

7 Sample Complexity of Adversarial Noise

We want to formalize the complexity of detecting adversarial noise in the framework of PAC learning. The objective in PAC learning is to produce a classifier that, with probability $1-\delta$, has an error rate of at most ϵ. To achieve this, the learning algorithm is supplied with m number of i.i.id training samples and their correct classifications. Determining the minimum $m(\epsilon, \delta)$ of training samples necessary to achieve the objective would give us the model's sample complexity.

If we know the sample complexity of a black-box linear classifier to be m, we would like to find a relationship with respect to m', the sample complexity of detecting adversarial noise.

Given the adversarial model, we provide intuition on why the sample complexity m' of learning adversarial noise is bounded by the sample complexity of the attacked linear classifier m. To add adversarial noise, the adversarial algorithm must know the hyperplane of the linear classifier (Fig 10). To attack a sample point, the adversarial algorithm must then move the point across this hyperplane (Fig 11). Since the adversarial noise only moves the point to the other side of the boundary, the sample complexity of detecting adversarial noise at most the sample complexity of the linear classifier. Furthermore, $m' << m$ because the adversarial algorithm pushes the point to the nearest point on the boundary to flip the label. This action reduces the space spanned by the sample points.

8 Conclusion

Many papers have proposed various methods of generating adversarial noise that can fool a neural network classifier. In this paper, we demonstrate that it is much easier to detect an adversarial attack on an image compared to Gaussian noise. We hypothesize that this is mainly due to the structure of adversarial noise, which is calculated as a gradient loss. This gives us reason to believe that adversarial noise may not be so harmful as described in literature. In addition, by examining the iterative variant of adversarial noise generation, we are able to infer some characteristics about the decision boundaries near an image. Perhaps by knowing an adversarial algorithm, we could learn a subset of the decision boundaries of a black-box classifier.

9 Future Work

There are multiple directions for continuing this work. First, we are interested in more formally analyzing the ability to detect adversarial images in a PAC-learning framework, specifically by considering the sample complexity of the original classifier as an upper bound of the sample complexity of the binary detection classifier. This line of work may show that the problem of adversarial noise is not as harmful as literature would suggest if we are able to easily detect it.

Second, the MNIST dataset may not be the most representative of the statistical properties of adversarial noise and our generated images, as there is limited variation in image structure. To evaluate the transferability of our results, we would like to extend the prior experiments to more

complex datasets such as ImageNet. We expect to see similar experimental results even with more complex input data.

Furthermore, we would like to use the iterative variants of FGSM to understand the decision boundaries of the original classifier especially in the multiclass case. In particular, if we stop the attack as soon as a decision boundary is crossed, we may be able to extract information about the decision boundary from the adversarial algorithm.

Acknowledgments

We would like to thank Yaron Singer and Adam Breuer for enlightening conversation and feedback throughout the process of this project. We would also like to the thank the rest of the CS282R class for helpful comments and sharing their own projects and ideas throughout the semester.

References

[1] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[2] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[3] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, pages 97–112, 2011.

[4] N. Carlini and D. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *ArXiv e-prints*, May 2017.

[5] Edith Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 514–523. IEEE, 1997.

[6] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting Adversarial Attacks with Momentum. *ArXiv e-prints*, October 2017.

[7] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting Adversarial Samples from Artifacts. *ArXiv e-prints*, March 2017.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[9] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[10] B. Liang, H. Li, M. Su, X. Li, W. Shi, and X. Wang. Detecting Adversarial Examples in Deep Networks with Adaptive Noise Reduction. *ArXiv e-prints*, May 2017.

[11] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. 11 2016.

[12] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

10 Appendix

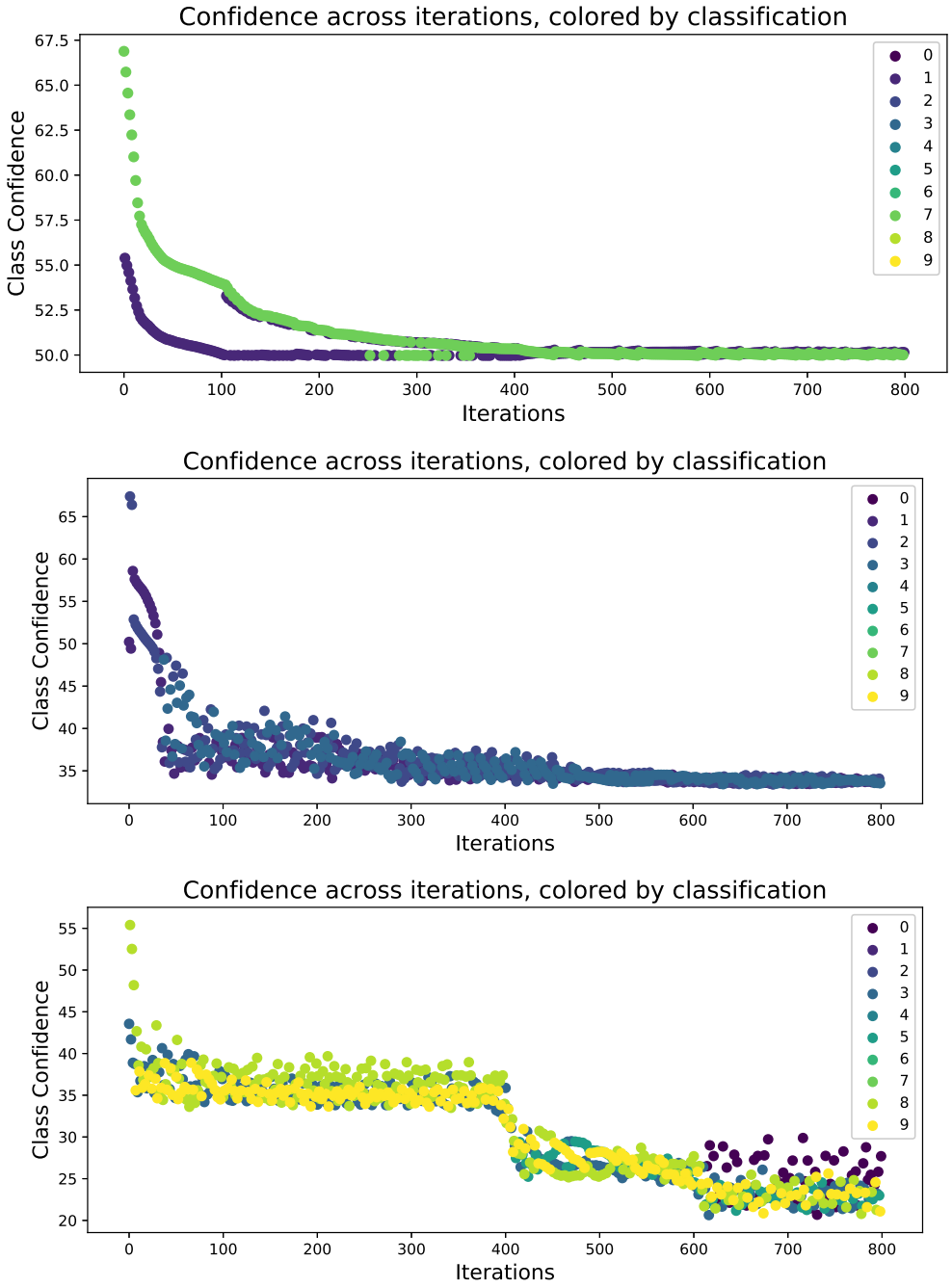

Figure 12: Classification confidence of the MNIST multi-layer perceptron over images generated on each outer iteration of Algorithm 2. The points are colored by the predicted class.