Adversarial Machine Learning

Ivelin Bratanov
ivelin.bratanov@mail.mcgill.ca
260535395

Cody Mazza-Anthony
cody.mazza-anthony@mail.mcgill.ca
260405012

Sorin Muchi
sorin.muchi@mail.mcgill.ca
260575810

***Abstract*— In safety-sensitive contexts such as autonomous vehicles, the success of machine learning applications depends on a thorough vetting of their resistance to adversarial data. This paper explores the field of adversarial machine learning by looking at recent research done in this area and combining their various approaches. This is done by stress-testing a CNN classifier on the MNIST digit database under adversarial conditions. We compare performance on two different validation sets (with and without adversarial examples), in the two different scenarios where the model is trained on a dataset with and without adversarial examples. Adversarial examples are generated using the Fast Gradient Sign Method and Jacobian-based Saliency Map Approach. Additionally, several approaches for increasing robustness are applied to the model, such as adversarial training and anomaly detection using Robust PCA.**

Our results are on par with the current literature, training on adversarial examples does reduce misclassification rates to a certain extent, and other filtering techniques such as robust PCA alone do not significantly improve accuracy.

I. INTRODUCTION

Machine learning algorithms are becoming increasingly popular in a multitude of applications, and many are responsible for financial assets, information security, and even human lives. These models are susceptible to many different forms of attacks that can compromise their ability to function properly. As well, they can even be manipulated to produce unwanted outputs, which can lead to devastating consequences. One of these, known as adversarial attacks, consist of input data specifically designed to be misclassified by the model. Essentially, these adversarial examples are intentionally worst-case perturbations of real-world examples for which the model outputs an incorrect answer with high confidence. The example can even be crafted to give a specific output which is beneficial to the adversary.

Quite often, these perturbations are perceptually indistinguishable, and difficult to detect and prepare for. What is scarier still, is that an adversarial example that was designed to be misclassified by a model is often also misclassified by other, different models trained on completely different data. This means that it is possible to generate adversarial examples and perform a mis-classification attack on a system without access to the underlying model.

As the goal of this project is to investigate safe and responsible methods for machine learning, we believe that exploring adversarial example generation and defense mechanisms is not only relevant, but also incredibly important for ensuring that models are secure and resistant to external influence. One does not need to look far for real-world applications of adversarial attacks: crashing autonomous vehicles, hacking into biometric authentication systems, and bypassing illegal content filters.

There are a multitude of different approaches for generating adversarial examples. The "fast-gradient sign method" and "Jacobian-based saliency map approach" are two of the most discussed and well-documented.

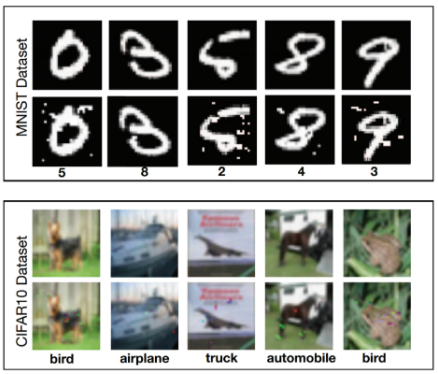

Fig. 1. Set of legitimate and adversarial samples for two datasets: For each dataset, a set of legitimate samples, which are correctly classified by DNNs, can be found on the top row while a corresponding set of adversarial samples (crafted using [1]), misclassified by DNNs, are on the bottom row. (source: [2])

On the defensive side of things, a common approach for combating adversarial attacks is to include adversarial examples in the training set, which is explored in this report. Additionally Robust Principal Component Analysis can be used for anomaly detection, which we explore in this paper. Other techniques used for anomaly detection are the use of autoencoders for nonlinear dimensionality reduction, as well as PCA-GRID with Median Absolute Deviation (MAD) as a distance measure, which we will address in the discussion section.

II. RELATED WORK

Although it is currently a very popular field for experts in both academia and industry, adversarial machine learning is relatively new and there is still much to be discovered. Machine learning experts in both academia and industry are currently working hard to develop robustness techniques for protecting against adversarial attacks.

Two of the primary papers consulted in this report are concerned with generating adversarial examples from existing data. Goodfellow et al. [3] explore the fast-gradient sign

method (FGSM) and determine that adversarial attacks are particularly effective against neural networks due to their linear nature. They are able to successfully reduce the test set error of a maxout network by training it on adversarial examples. Papernot et al. [1] use the Jacobian-based saliency map approach (JSMA) to achieve a 97% adversarial success rate on a deep neural network while only modifying a tiny part of each real-world example - on average 4.02% of the input features. The researchers who coined the above-mentioned methods additionally built an open-source library [4] which assists in the creation of adversarial examples, and which we discuss in the implementation section below.

In 2016, Kurakin et al. [5] use FGSM, among other methods, to demonstrate that even a large number of printed adversarial images that resemble the original to the human eye, when fed to an Inception v3 image classification neural network through a cell-phone camera, are misclassified. Furthermore, none of the image transformations designed as defense mechanisms were able to recognize 100% of adversarial attacks. These findings have profound implications for machine learning models that operate in the physical world.

As mentioned earlier, it has been shown that adversarial examples generalize across models trained to perform the same task. Szegedy et al. [6] were one of the first to demonstrate this, by training models with different architectures on different training sets and observing that they misclassify the same perturbed input. This means an attacker can train their own model, generate adversarial examples against it, and then deploy those adversarial examples against a model they do not have access to.

Until recently, adversarial attacks were assumed to at least need access to the model's training data in order to be successful, yet earlier in 2017 even this was disproven when Papernot et al. [7] developed a "black-box" DNN that only had access to the outputs produced by the target model. It was found that this method still managed to fool this target model. The team achieved 84.24% misclassification on a model hosted on MetaMind, and even higher rates on Google and Amazon using JSMA on the MNIST dataset.

Thus far, no fully effective mechanism for defending against adversarial attacks has been found, although the two approaches for increased model robustness with the best results thus far are adversarial training and defensive distillation for DNNs. However, even for these methods, Papernot et al. [7] show that the black-box attack mentioned above is effective at overcoming the defense and achieving misclassification.

Szegedy et al. [6] first demonstrated that adversarial training, which consists training the model on different combinations of regular and adversarial data, is effective at reducing misclassification. Additionally, Goodfellow et al. [3] have shown that adversarial training can improve a model's accuracy on non-adversarial examples, as illustrated in Figure 2.

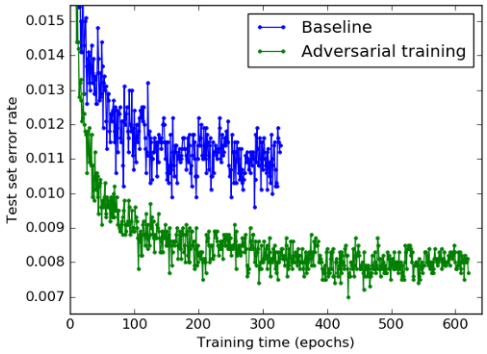

Fig. 2. Reduced error rate on a model subject to adversarial training - even on non-adversarial test set examples (source: [8])

Defensive distillation for DNNs, developed by Papernot et al. [2] consists of extracting class probability vectors produced by an initial DNN or an ensemble of DNNs. They subsequently trained a second DNN with the same architecture as the first on the probability vectors of the first. The team demonstrates that defensive distillation successfully reduces the success rate of adversarial example crafting (from 95% to 0.5%) against a DNN trained on the MNIST dataset, without reducing the model's classification accuracy.

In addition to training on adversarial data, other defense mechanisms such as Robust Principal Component Analysis (Robust PCA) have been discussed in the literature. Robust PCA has been proposed as an effective tool for anomaly detection in general [9], and for images in particular [10]. Variations of Robust PCA have been used both for extracting meaningful information from image data (ex. foreground detection) [11], as well as for isolating anomalous data from regular data [12].

III. PROBLEM DEFINITION

The purpose of this paper is to explore the question: How easy is it to generate effective adversarial attacks, and can we defend against them? Figure 2 below offers a clear visual representation of the problem at hand. Each of the adversarial examples are still easy to classify by the human eye, but misclassified with high confidence by a DNN.

The problem question is explored using the MNIST dataset, as well as multiple adversarial datasets, generated using FGSM and JSMA and described below.

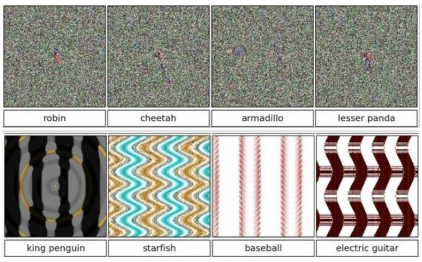

Fig. 3. These images are classified with $> 99.6\%$ confidence as the shown class by a Convolutional Network. (source: [2])

A. MNIST Dataset

The MNIST imageset is very commonly used in adversarial machine learning research, due to its large set of images for what is seen now as a relatively simple classification task. The large number of images has allowed the machine learning community to reach particularly low error rates for digit classification (0.23 with a CNN [13]). Additionally, the low dimension of the images allows for reduced computation time during learning and adversarial example generation. The dataset consists of a training set of 60,000 28x28 black and white images, and a test set of 10,000 images. Features were encoded as a 28x28 matrix, representing each pixel in the image and a binary value for black/white.

B. Adversarial Dataset

FGSM and JSMA were used to generate adversarial datasets for training. These perturbed images are stored in a numpy array, and consist of the same dimensions and feature structure as the normal MNIST dataset.

Using FGSM, which is more efficient than JSMA, a training set of 10,000 images, and a test set of 1000 images were generated. Additionally, although JSMA proved to be much more computational intensive, a 10000-image training set was generated and a test set of 1000 images was also generated. We performed simulations on 1000 training examples and test set for both of the adversarial methods. The size of the training and testing set were chosen to probe on where to focus our efforts because of the lack of computational resources. We also performed simulations on 10000 training examples and 1000 test examples respectively (Table I).

Computational restrictions discussed in the implementation section below limited us to performing robust PCA on our result set of 1000 images for JSMA and FGSM respectively.

IV. ALGORITHM SELECTION AND IMPLEMENTATION

A. CNN

Using a convolutional neural network for this problem was the obvious choice, due to its high success rates for image classification tasks, and its particularly effective classification accuracy on the MNIST dataset. Also, adversarial training has been found to be most effective with DNNs.

The CNN was set up using Keras [14] and TensorFlow [15], and trained on each of the datasets over 5 epochs with a learning rate of 0.1. The number of epochs was changed from 5 to 10 for certain result sets.

B. Cleverhans Library

Cleverhans [4] is a Python library developed by OpenAI, and specifically, the researches who developed the FGSM and JSMA methods for generating adversarial examples. The library is meant to assist in the generation of adversarial examples. We used code from the Cleverhans tutorials section which includes methods to craft both JSMA and FGSM adversarial examples.

An additional library for assisting with adversarial example crafting called Deep-Pwning was discovered, though not explored due to time restrictions.

C. Jacobian-Based Saliency Map Approach

JSMA consists of generating a Jacobian matrix relating the output class and input features of the model, indicating how much impact each feature has on each output class. Then, a saliency map is generated to identify the best features for the adversarial goal. This goal being to misclassify an example in a chosen target class, while applying the smallest number of changes to the input features of the given example. Thus, JSMA specifically helps with crafting examples with a misclassification output goal, rather than simply causing the model to make a wrong prediction.

The mathematical approach to generating an adversarial example X^* from a normal example X consists of solving the optimization problem below.

$$\arg\min_{\delta_X} \|\delta_X\| \quad \text{s.t. } \mathbf{F}(\mathbf{X} + \delta_\mathbf{X}) = \mathbf{Y}^* \tag{1}$$

$X^* = X + \delta_X$ is the adversarial example, Y^* is the target adversarial output, and $\|\delta_X\|$ the norm of the perturbation vector.

The Jacobian matrix represents the forward derivative of the function learned by the CNN. The Jacobian matrix is then used to construct the saliency map (see Figure 4) which indicates which features to include in perturbation δ_X.

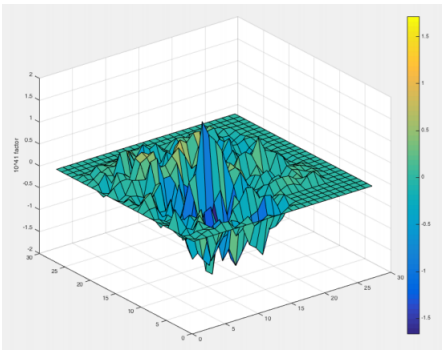

Fig. 4. Saliency map of a 784-dimensional input, corresponding to the 28x28 image pixel alignment. Large absolute values correspond to features with a significant impact on the output when perturbed. (source: [16])

In order to generate the adversarial dataset, we compute the Jacobian matrices for each target, then iterate over all samples and perturb all target classes. We then train the CNN on the training set (where appropriate) and compute the misclassificaiton rate for each test set example.

D. Fast Gradient Sign Method

The fast gradient sign method introduces a tiny perturbation to the features of a real-world example which amounts to an overall large change in the prediction when multiplied by the weight vector of a high-dimensional model. This tiny perturbation on its own is so small, that the adversarial

example itself is almost indistinguishable from the original one.

The core part of FGSM is computing a perturbation vector η where the adversarial example is $x^* = x + \eta$. θ represents the parameters of the model, x the input to the model, and y the target associated with x. $J(\theta, x, y)$ is the cost used to train the neural network, and ϵ, the predetermined bound meant to keep η sufficiently small. The full equation is shown below.

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \tag{2}$$

The required gradient is computed using backpropagation. This is done for each normal example x, and we are able to generate a larger number of training examples (10,000) due to the lower computation cost.

E. Robust Principal Component Analysis

The use of Robust PCA for anomaly detection was inspired by [12]. Our implementation was based on the variant of robust PCA proposed by Candes et al. [11].

In order to avoid misclassification due to the adversarial nature of the input, we would like to implicitly generalize our model by extracting the principal components of these images, and ignore most perturbations. Therefore we decompose input images I into

$$I = L_0 + S_0 \tag{3}$$

where L_0 and S_0 are each a resuling low-rank and sparse matrix, respectively. We compute L_0 and S_0 using the Principal Component Pursuit (PCP) estimate [11], defined as

$$\min \|L\| + \lambda \|S\|_1 \text{ s.t. } L + S = I \tag{4}$$

where $\|L\| := \sum_i \sigma_i(L)$ denotes the nuclear norm of the matrix L, that is, the sum of the singular values of L, and $\|S\|_1 = \sum_{i,j} |S_{i,j}|$ denotes the ℓ_1-norm of S seen as a long vector in $\mathbf{R}$. We then ignore the resulting low rank matrix L, and only consider the sparse matrix S as our input. This type of preprocessing is to be applied to all inputs, for both training and testing sets.

V. TESTING AND VALIDATION

We now use our experimental set-up to answer the questions presented at the beginning of this paper: (1) How easy is it to craft effective adversarial examples? (2) Can we defend our models against such malicious examples? The primary findings can be found in Table III in the appendix.

A. Crafting Adversarial Examples

As previously described in the literature related to adversarial machine learning, our findings strongly prove the assertion that crafting adversarial examples is not only possible but highly effective. We find that a Convolutional Neural Network trained on only non-adversarial examples from the 10-class MINST dataset, performs significantly worse than random when tested against both the Jacobian-based saliency map (7.69% accuracy) and the fast gradient sign method (5.60% accuracy).

Although the JBSM approach to crafting adversarial examples is computationally expensive, but very effective, the FGSM method seems to be significantly faster, and equally dangerous. Furthermore, our findings indicate that crafting, and using such adversarial examples in practice, without previous knowledge about a given model, is almost trivial [17], [18], [19], and can be used even without access to the underlying model in order to induce potentially devastating results [7].

B. The Adversarial Training Defense (1000 examples)

However, training on adversarial inputs seems to be improving the model accuracy when facing adversarial examples, and therefore render the model significantly more robust. For example prediction accuracy increases from 7.69%, when training on 1,000 non-adversarial examples to 14.29% when training on a test set of 100 non-adversarial and 900 JBSM examples, and testing on 1,000 JBSM examples. Similarly, accuracy increases from 5.60%, when training on 10,000 non-adversarial examples to 66.93% when training on 100 non-adversarial examples and 900 FGSM examples, and testing on 1000 FGSM examples.

Furthermore, the fact that this 10/90 split is showing the most promising results is highly indicative that a larger training set could produce significant improvements. Based on our preliminary findings, and the current literature we have strong reason to believe that using larger datasets would significantly improve model's generalizability, and resistance to adversarial examples [1], [3], [16]. We show this in tables I and II for the FGSM method, where we improve a model's accuracy from 4.8% to 90.6% when training on 10,000 examples for 10 epochs. However, due to constrained computational resources, and the high computational requirements of the JBSM adversarial example generation method, we were not able to experiment with significantly larger datasets.

C. The Adversarial Training Defense (10000/1000 training/test examples)

The results of these simulations are found in Table I and Table II in the Appendix section(end of the report).

Training on a larger test set can greater improve accuracy on adversarial examples. This is clearly shown when training on FGSM data set only. The accuracy on FGSM test set increases from 54.5% accuracy to 90.4%. However, JBSM examples seem to be very difficult to interpret. The accuracy does not appear to increase on the JBSM test set.

When interpreting the 10/90 split, FGSM accuracy increases for all three options in the test set especialy on the normal and FGSM test sets. Although 10/90 split for JBSM increases the accuracy on the normal test set and FGSM test sets, it does not increase accuracy on the JBSM test set.

Another interesting finding is that training entirely on 10,000 adversarial FGSM examples actually produced nearly the same accuracy as 10,000 unperturbed examples (87.29% vs. 87.69%). This indicates that there is not a tradeoff

when adversarial examples are used instead of real-world examples, and thus adversarial examples in training will not reduce the accuracy of our models. This supports the findings in [3] [6].

Increasing the number of epochs in our simulations was shown to improve accuracy when training with FGSM examples only. When training on JBSM examples, increases were observed on JBSM and normal test sets, but not the JBSM test set.

D. Other Defense Mechanisms

Our findings indicate that the naive use of Robust PCA to adversarial inputs does not significantly improve prediction accuracy, or reduce classification error, and that a more sophisticated approach would be required instead. Indeed, our implementation of Robust PCA based on [11] only increased our prediction accuracy from 7.6% to 12.6% on an adversarial test set of size 1,000, generated using JBSM on the MIST dataset. Therefore it seems that Robust PCA alone is neither sufficient, nor adapted to protecting our models against adversarial examples. Instead other methods such as autoencoders, and distillation should be used (cf. Discussion).

VI. DISCUSSION

The findings of our work tend to indicate that indeed, adversarial example generation, especially in the case of FGSM, is relatively easy to accomplish, and quite effective at forcing a CNN model with 87.7% classification accuracy down to 5.6%, which is significantly worse than random guessing at approximately 10%. In terms of defense, adversarial training does increase the accuracy of models under adversarial conditions, though not by as much as was expected. This might be due to the relatively small training sets used, and the small number of epochs. The most significant improvement found using adversarial training was on a training set using 10% real-world examples and 90% FGSM adversarial examples, from 5.60% accuracy to 66.93%. When increasing the size of our training set we can observe that adversarial training on FGSM is very effective, whereas with JBSM adversarial training is not nearly as effective. This behaviour should be further investigated at a later time.

One major drawback of our approach was the small number of images used for training. As mentioned earlier, more computation resources would have allowed us to generate more FGSM and JSMA examples allowing us to achieve results which are more indicative of trends and the benefits of adversarial training. Persisting our adversarial examples in a .npy file and re-using these same examples for each new experiment, allowed us to avoid having to regenerate adversarial examples every time.

In the ideal case, our team would have liked to have access to 30,000 adversarial examples of both JMSA and FGSM, allowing us to train entirely on those images, as well as analyze the effects of various ratios of normal and adversarial images in the train set. It would be interesting to see whether a combined adversarial training set, consisting of a mix of unperturbed examples with both FGSM and JSMA examples, would outperform the other methods attempted, and what ratio of each type of example would lead to the lowest misclassification.

Additionally, training on a larger number of epochs would allow us to reach a higher classification accuracy on the models subject to adversarial training. Once again, this would have allowed us to really observe the positive impacts of adversarial training.

Another issue which particularly slowed down the implementation was the lack of documentation for the Cleverhans black-box adversarial example generation library. This made setting up and using the library non-trivial, since many dependencies were not documented, nor was some of the functionality this project required.

We would also like to further investigate the use of several other methods for anomaly detection. Robust statistical methods, notably ANTIDOTE[12], a version of PCA, using the Median Absolute Deviation as a robust measure of dispersion, has shown encouraging results for network traffic anomaly detection and could potentially make a promising candidate for anomaly detection in the context of image recognition as well. Autoencoders, which have also been used for anomaly detection, through the literature [20], [21] could be helpful for both flagging adversarial examples, as well as for reducing the dimensionality of the input, and therefore reducing the adversarial attack surface. Similar to the generating of adversarial examples, applying robus PCA was also very computationally intensive and restricted our training set to 1000 examples.

Throughout the course, we have felt that the main bottleneck of interesting results has been computational capacity. The main motivation is to adequately learn cloud platforms (AWS, Google cloud, etc.) to go from idea to realization as quickly as possible. Fast iteration, we feel is one of the main components to succeed in the machine learning field. It is safe to say that we are all motivated to run our models more effectively.

Additionally, our team would like to explore other known models for adversarial defense and model robustness, as well as attempt new methods which have not yet been explored. In particular, it would be interesting to compare defensive distillation for DNNs with both FGSM and JSMA to our results for adversarial training. A combination of the two methods, in the form of a CNN trained on both normal and adversarial examples, distilled into a second CNN, may have promising results. Thus far, we have not found any attempts to use SIFT/SURF for image classification in adversarial settings. This would be interesting to explore, since SIFT/SURF is focused on identifying core elements of an image, referred to as keypoints, rather than looking purely at individual pixels. This may prove to be an effective approach for classifying images which appear similar to real-world examples with only small perturbations, because as opposed to only looking at the image pixel by pixel, SIFT/SURF allow to generalize by region and thus might minimize the overall effect of small adversarial perturbations.

Additionally, SIFT/SURF keypoints are most often transformed into features using KMeans clustering, which could further minimize the effect of adversarial perturbations, since image keypoints are clustered together by looking at relative overall distance.

It is possible to see that the field of adversarial machine learning is relatively new and there is still much to be discovered. Machine learning experts in both academia and the industry are currently working hard to develop robustness techniques for protecting against adversarial attacks, as they are of utmost importance for many applications of machine learning.

With the introduction of adversarial examples, it is clear that machine learning models, particularly in image classification, still have a long way to go to catch up to the human eye and really "see" what is presented in front of them. We would like to end with a quote from Goodfellow et al.: "The existence of adversarial examples suggests that being able to explain the training data or even being able to correctly label the test data does not imply that our models truly understand the tasks we have asked them to perform."[3]

VII. Statement of Contributions

- Ivelin: Research on FGSM, JSMA, explored related works and implementation options, wrote report
- Cody: Implementation of CNN with TensorFlow & Keras, generation of adversarial data with Cleverhans, testing of model under various adversarial conditions
- Sorin: Research and redaction on adversarial defense mechanisms, helped with redacting the report

We hereby state that all the work presented in this report is that of the authors.

References

[1] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE, 2016.

[2] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 582–597, IEEE, 2016.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[4] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "cleverhans v1.0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2016.

[5] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[7] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, ACM, 2017.

[8] I. Goodfellow, "Deep learning adversarial examples - clarifying misconceptions," July 2015.

[9] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," tech. rep., DTIC Document, 2003.

[10] C. Croux, P. Filzmoser, and M. R. Oliveira, "Algorithms for projection–pursuit robust principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 87, no. 2, pp. 218–225, 2007.

[11] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.

[12] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pp. 1–14, ACM, 2009.

[13] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3642–3649, IEEE, 2012.

[14] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015.

[15] Google, "Tensorflow: An open-source software library for machine intelligence,"

[16] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE, 2016.

[17] P. Tabacof and E. Valle, "Exploring the space of adversarial images," in *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp. 426–433, IEEE, 2016.

[18] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[19] R. Shokri, M. Stronati, and V. Shmatikov, "Membership inference attacks against machine learning models," *arXiv preprint arXiv:1610.05820*, 2016.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[22] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[23] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25, ACM, 2006.

[24] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58, ACM, 2011.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[26] I. Caswell, A. Nie, and O. Sen, "Exploring adversarial learning on neural network models for text classification,"

[27] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 387–402, Springer, 2013.

[28] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[29] B. Miller, A. Kantchelian, S. Afroz, R. Bachwani, E. Dauber, L. Huang, M. C. Tschantz, A. D. Joseph, and J. D. Tygar, "Adversarial active learning," in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pp. 3–14, ACM, 2014.

[30] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[31] N. Papernot and P. McDaniel, "On the effectiveness of defensive distillation," *arXiv preprint arXiv:1607.05113*, 2016.

[32] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[33] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[34] F. Pedregosa, G. Varoquaux, and e. a. Gramfort, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

APPENDIX

TABLE I
SUMMARY OF MAIN RESULTS

Training/Testing	Normal (1K)	JBSM (1K)	FGSM (1K)
Normal (10K examples)	87.69%	7.69%	5.60%
10K JBSM examples	11.9%	12.4%	12.29%
10K FGSM examples	87.29%	11.2%	90.4%
10K 10/90 split JBSM	53.5%	10.7%	17.2%
10K 10/90 split FGSM	73.7%	9.1%	88.79%

TABLE II
SUMMARY OF MAIN RESULTS (10 EPOCHS)

Training/Testing	Normal (1K)	JBSM (1K)	FGSM (1K)
Normal (10K examples)	93.1%	9.8%	4.8%
10K 10/90 split JBSM	70.8%	12.5%	14.6%
10K 10/90 split FGSM	79.89%	9.2%	90.6%

TABLE III
RESULTS: TESTING ACCURACIES ON THE NORMAL, ADVERSARIAL JBSM AND ADVERSARIAL FGSM TEST SETS WHEN TRAINING ON DIFFERENT TRAINING SETS.

Training/Testing	Normal (1K)	Adversarial JBSM (1K)	Adversarial FGSM (1K)
Normal (1K examples)	47.80%	9.59%	9.80%
Adversarial (1K JBSM)	10.99%	9.00%	14.10%
Adversarial (1K FGSM)	13.90%	9.29%	54.5%
Normal + Adversarial (1K total 70/30 split JBSM)	41.40%	11.68%	10.49%
Normal + Adversarial (1K total 30/70 split JBSM)	32.20%	12.99%	16.70%
Normal + Adversarial (1K total 70/30 split FGSM)	42.29%	9.69%	32.30%
Normal + Adversarial (1K total 30/70 split FGSM)	30.80%	9.79%	50.29%
Normal + Adversarial (1K total 90/10 split JBSM)	54.80%	10.59%	15.50%
Normal + Adversarial (1K total 10/90 split JBSM)	11.80%	14.29%	13.10%
Normal + Adversarial (1K total 90/10 split FGSM)	65.90%	9.59%	19.39%
Normal + Adversarial (1K total 10/90 split FGSM)	18.50%	8.89%	66.39%
Normal (1K examples) PCA on test set	10.80%	9.30%	11.10%
Normal (1K examples) PCA on both sets (train/test)	11.70%	11.70%	11.70%
Normal + Adversarial FGSM (1K total 50/50) PCA on both sets(train/test)	11.60%	12.60%	12.60%
Normal + Adversarial JBSM (1K total 50/50) PCA on both sets(train/test)	11.60%	12.60%	12.60%