NUWC-NPT Technical Memo 23-042

Neural Network Anomaly Detection

Sherida T. Jacob
Undersea Warfare Combat Systems Department

31 August 2023

Naval Undersea Warfare Center Division
Newport, Rhode Island

ADMINISTRATIVE INFORMATION

This technical memo was prepared under the NUWC Division Newport project "Neural Network Anomaly Detection," principal investigator Sherida T. Jacob (Code 2511). The sponsoring activity is the NUWC Division Newport Internal Investment Program, program manager Derek K. Potvin (Code 00X).

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
31-08-2023	Technical Memo	01-10-2022	30-09-2023

4. TITLE AND SUBTITLE

Neural Network Anomaly Detection

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)

Sherida T. Jacob

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Naval Undersea Warfare Center Division Newport 1176 Howell Street Newport, RI 02841-1708	TM 23-042

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Naval Undersea Warfare Center Division Newport 1176 Howell Street Newport, RI 02841-1708	NUWC	

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Deep neural networks (DNNs) are used across various domains and have the potential to reduce operational costs, increase productivity, and improve safety and security. Although DNNs are used with a high rate of success, they are inherently vulnerable to adversarial examples (AEs), which are inputs to a trained DNN that are modified so that the DNN is deceived into misclassifying the inputs. If this vulnerability is exploited, the DNN can produce unexpected results that may adversely affect public health and safety. In this research, analytical approaches identified image-based AEs. DNN models were created using the Tensorflow library. The hidden layer activations of “normal” and adversarial inputs were created using images from the Canadian Institute for Advanced Research-10 classes dataset. The Uniform Manifold Approximation and Projection code was used to visualize how AEs differed from normal inputs relative to class manifolds. The Stochastic Gradient Descent One Class Support Vector Machine anomaly detection algorithm was applied to the hidden layer activations to detect AEs. Resulting key insights into how DNNs interpreted AEs will lead to mathematical strategies to identify AEs.

15. SUBJECT TERMS

adversarial examples, anomaly detection algorithm, convolutional neural network, deep neural network, neural network, One Class Support Vector Machine, UMAP, Uniform Manifold Approximation and Projection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
(U)	(U)	(U)	SAR	15

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
SHERIDA T. JACOB	401-832-4776

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

ADA	anomaly detection algorithm
AE	adversarial example
CIFAR-10	Canadian Institute for Advanced Research-10
CNN	convolutional neural network
DNN	Deep neural network
FGSM	Fast Gradient Sign Method
KNN	K-nearest neighbors
MNIST	Modified National Institute of Standards and Technology
MSE	mean squared error
NUWC	Naval Undersea Warfare Center
ReLU	rectified linear unit
SGDOneClassSVM	Stochastic Gradient Descent One Class Support Vector Machine
Tgted_Adv_Plane	targeted adversarial airplane
UMAP	Uniform Manifold Approximation and Projection
Untgted_Adv_Plane	untargeted adversarial airplane

1. INTRODUCTION

A neural network is a series of algorithms designed to imitate the functions of a human brain. The nodes in a neural network consist of weights and biases and act like neurons. These nodes form the "hidden layer(s)" of the network and serve as learning parameters that are modified as the network is trained. The hidden layers enable the neural network to model nonlinear, complex relationships—while building on previous knowledge—in order to process and interpret large amounts of complex data. Neural networks "learn" when the nodes interact to modify their values to produce the desired output. Once a neural network is trained to learn the input-output relationships of a dataset, it can then make generalized assessments of new data when first seen. A deep neural network (DNN) is a neural network with at least two hidden layers [1]. Multiple hidden layers allows the neural network to improve the generalization of the model, which can lead to improved accuracy of the outputs.

DNNs are used in many aspects of national security and modern warfare. Facial recognition systems at border entry points, unmanned vehicles such as the Razorback unmanned underwater vehicle, the Reaper unmanned aerial vehicle, and the tracked Robot 10-ton to support troops on the battlefield are just a few examples. DNNs are a benefit to military operations because they have the potential to reduce operational costs, increase productivity, and improve safety and security.

However, although DNNs are able to perform tasks such as pattern recognition, clustering, and classification with a high level of success, researchers have found that DNNs do not understand the underlying characteristics of the data [2]. Additionally, researchers do not have a clear understanding of how neural networks make decisions. DNNs operate like a "black box."

The hidden layer neurons do not perform dedicated functions to complete a task. Each neuron in the hidden layer receives data from all of the outputs of the previous layer. Weights and biases that are gradient descent optimized are applied to the data received. A nonlinear activation function (for example, convolution, pooling, sigmoid, or rectified linear unit (ReLU)) is then applied to the sum of all the data; the result is the output of the neuron. The interdependency of the neurons and the complex representations of the data produced by the nonlinear functions hinder research efforts to understand the factors that contribute to a given decision. In fact, it is possible for different neural networks with the same topology and input data but different weights and biases to produce the same decisions.

The inability of DNNs to comprehend the underlying characteristics of the data coupled with the inability of researchers to detail how decisions are determined by the DNNs—the black box effect—make DNNs susceptible to the deleterious effects of adversarial examples.

Adversarial examples (AEs) are replicas of correctly categorized inputs that contain distortions. These distortions would not trick a human, but they can trick the network into misinterpreting inputs and producing unexpected—and inaccurate—outputs. Initial research by Szegedy et al. [3], Goodfellow et al. [2], and Su et al. [4] explored AEs and ways to create them. In [3], the authors discovered that neural networks were vulnerable to AEs. They also found that an AE generated by one DNN could be misidentified by another DNN, as long as both DNNs were

designed to perform the same task. This phenomenon was observed even if the other DNN was trained on a different subset of the training data or had a different architecture.

In [2], the authors developed the fast gradient sign method, which exploits the gradient of the neural network to create AEs from images. In this method, the original image is altered by adding or subtracting a small error to each pixel. See Figure 1 from [2]. The authors also surmised that "linear behavior in high-dimensional spaces is sufficient to cause adversarial examples" [2]. In DNNs, each neuron is activated on a linear combination of its inputs; therefore, when a slight change is made to each input, it leads to a large change in the network's output. This change causes the input to be misinterpreted.

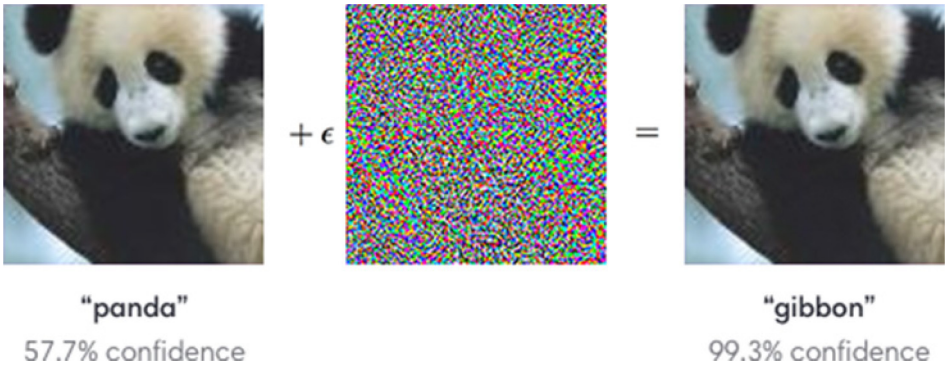

Figure 1. An Example of Using the Fast Gradient Sign Method (FGSM) to Misidentify an Image with High Confidence [2]

In [4], the authors proved that an AE could be created by changing a single pixel in an image. The authors used differential evolution to determine which pixel to change and how to change it in order to create an AE.

To protect against AEs, researchers use three primary methods: adversarial training, defensive distillation, and detection. In [2], the authors found that adding AEs to training data improved generalization performance by minimizing classification error rates. Adversarial training improved the resistance of a DNN to AEs better than typical machine learning regularization techniques such as dropout or weight decay.

In Papernot et al. [5], the authors explored using distillation to protect DNNs from AEs produced with the FGSM on datasets from the MNIST[1] and CIFAR-10[2] datasets. In defensive distillation, the outputs of one DNN are used as training data for another DNN. The class labels for the outputs of the second DNN are represented by probability distributions indicating the likelihood of the input being in each class, instead of "hard labels" (for example, 0 or 1, cat or dog). The first DNN is then re-trained on the output of the second DNN. This process improves the

[1] Modified National Institute of Standards and Technology database.
[2] Canadian Institute for Advanced Research-10 database.

generalization performance of the DNN because the probability distributions provide more information to classify the input than hard labels.

In Lu et al. [6], the authors used a radial basis function support vector machine to analyze the state of the internal layers of the DNN to determine whether an input was adversarial. In Feinman et al. [7], the authors used kernel density estimation to detect AEs, since AEs have a different density distribution than the original inputs.

Although adversarial training, defensive distillation, and anomaly detection algorithms (ADAs) are useful tools in the defense against AEs, they are not infallible. There are limitations to all AE defense mechanisms researched to date. Adversarial training can mitigate a DNN's vulnerability to AEs if the attack AE is known to the DNN. However, the high-dimensional space that a DNN allocates to a class might be larger than the space occupied by the training data for the class. The effort needed to create the optimal amount of training AEs for every possible input in order to make a DNN invulnerable to AEs is prohibitive.

Defensive distillation can be defeated with a strong Carlini and Wagner attack. This attack uses binary search to determine the appropriate optimization constraint needed to misclassify an input. As for ADAs, each one uses different methodologies to recognize anomalies; therefore, each algorithm has different limitations on its ability to detect anomalies.

This research expands upon the research conducted by Pocos et al. [8]. In [8], the authors used the MNIST and CIFAR-10 datasets to analyze the hidden layer activations of AEs with varying norm constraints (norm constraints restrict the complexity of the DNN). They used the K-nearest neighbors algorithm (KNN) to approximate the manifold onto which the AEs were projected. The KNN algorithm classifies new data points based on how similar they are to the classes in the training dataset. They found that, while AEs tend to get closer to the incorrect class at later layers in the network, some AEs do not leave the proximity of the manifold of the correct class. They also found that AEs generated using different norm constraints invoke various effects in the neural network; therefore, the cause of the misclassification can differ.

The research described here explores the idea of using the One Class Support Vector Machine ADA to analyze the hidden layer activations of a trained DNN. The intent of the analysis is to compare "normal" inputs and AEs to gain insight into how AEs are interpreted by a DNN. As DNNs are integrated into current and future military operations, it is important to mitigate the threat of AEs. Developing the ability to detect an AE will be an important part of the mitigation process.

2. EXPERIMENTAL SETUP AND RESULTS

2.1 EXPERIMENTAL SETUP

To conduct this research, a 22-layer (20 hidden layers) convolutional neural network (CNN) was constructed in Python using Keras and Tensorflow. Dropout and max-pooling were used to regularize the network. The CNN was then trained on the CIFAR-10 dataset and achieved an

83% classification accuracy, without overfitting. The CIFAR-10 dataset consists of 60,000 images from 10 classes of animals and machines. This dataset was then split into five datasets:

- Two training datasets of 22,500 images each
- A validation dataset of 5,000 images
- An adversarial dataset of 5,000 images
- A test dataset of 5,000 images

After the model was trained using the first training dataset, the hidden layer activations in the test dataset were captured, and the activations from each layer were plotted using UMAP. The mean squared error (MSE) algorithm was used to create untargeted AEs, and the sparse categorical cross entropy algorithm was used to create targeted AEs.

The hidden layer activations of the targeted and untargeted AEs were then captured. Untargeted AEs are inputs that are altered to be misclassified as any class other than the original class; targeted AEs are inputs that are altered to be misclassified as a specific class. After this was completed, Tensorflow and Keras were used to implement the Stochastic Gradient Descent One Class Support Vector Machine (SGDOneClassSVM) algorithm.

2.2 OBSERVATIONS AND RESULTS

For the first part of this research, UMAP plots of the CIFAR-10 images in the test dataset were created to observe how the classes were oriented on the manifold. Figure 2 shows UMAP plots of activated hidden layers 1 and 4, and Figure 3 shows similar plots of activated hidden layers 10 and 18. The test data in the scatter plots are color coded by class.

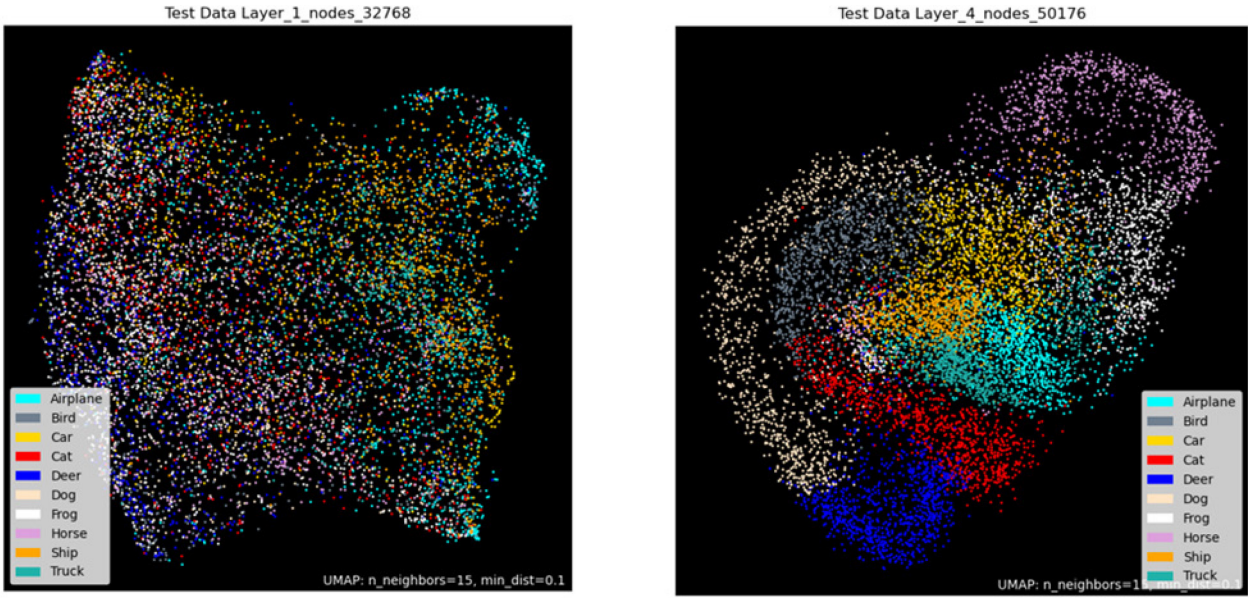

Figure 2. UMAP Plots of Hidden Layer Activations of CIFAR-10 Images: (left) Layer 1 and (right) Layer 4

It was observed that, as the data progressed through the network, the delineation between the class manifolds increased and the clusters of class data points tightened. In Figure 2, for example, the airplane class data that are fairly diffuse in hidden layer 1, are clustered in hidden layer 4. The plots serve as a visual demonstration that the CNN could adequately classify CIFAR-10 images.

After this task was completed, the MSE was used to create untargeted AEs. Then, a sparse categorical cross entropy algorithm was used to create targeted AEs from the same subset of CIFAR-10 images. The images in Figure 4 are examples of untargeted adversarial images. The CNN categorized an image of a deer as a frog (left side of Figure 4) and an image of a cat as a truck (right side of Figure 4) even though it correctly categorized the images before the adversarial image creation process altered them.

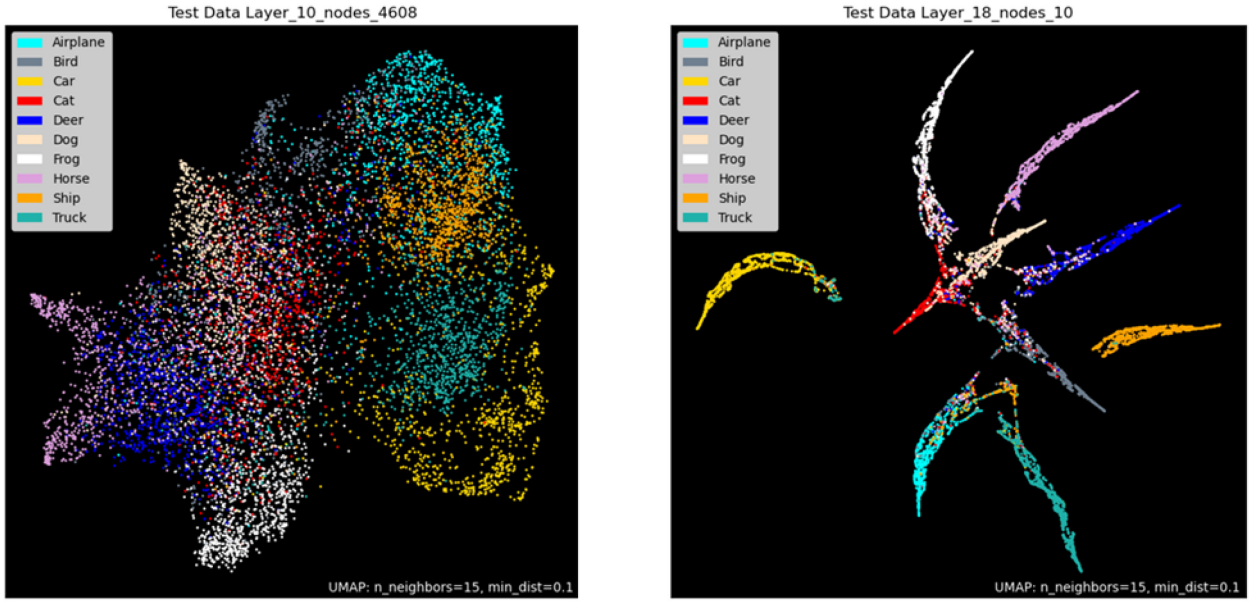

Figure 3. UMAP Plots of Hidden Layer Activations for CIFAR-10 Images: (left) Layer 10 and (right) Layer 18

Figure 4. Untargeted Adversarial Images Created Using the MSE Algorithm

Once all the adversarial images were created, the hidden layer activations of the adversarial images in the secondary training data were plotted in UMAP. Figures 5 and 6 show UMAP visualizations of where the AEs were oriented on the manifold in relation to the CIFAR-10 classes. Figure 5 plots the targeted and untargeted adversarial hidden layer activations for the set of CIFAR-10 images in layer 1. Figure 6 plots similar data for layer 4.

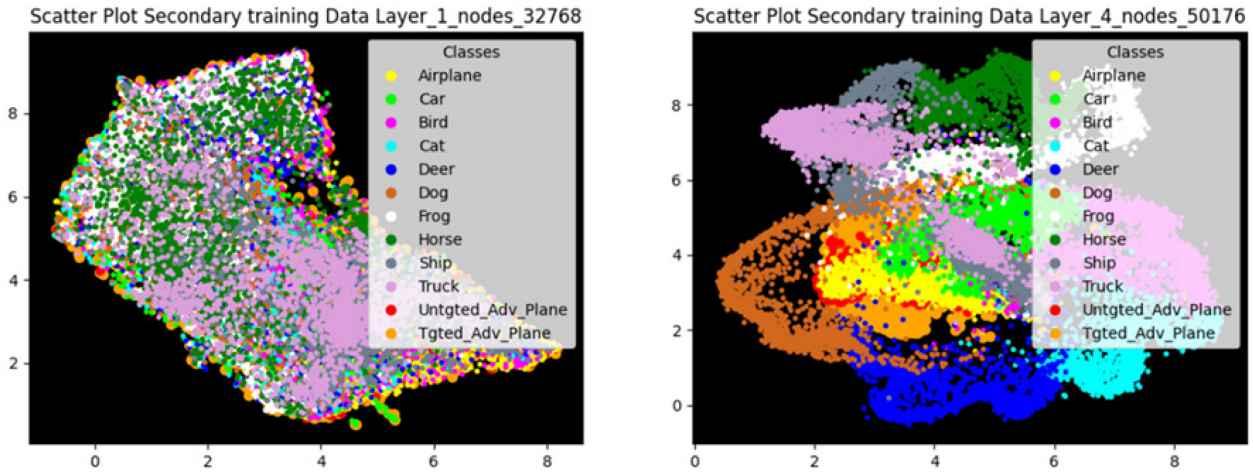

Figure 5. UMAP Plots of Targeted and Untargeted Adversarial Hidden Layer Activations for CIFAR-10 Images: (left) Layer 1 and (right) Layer 4

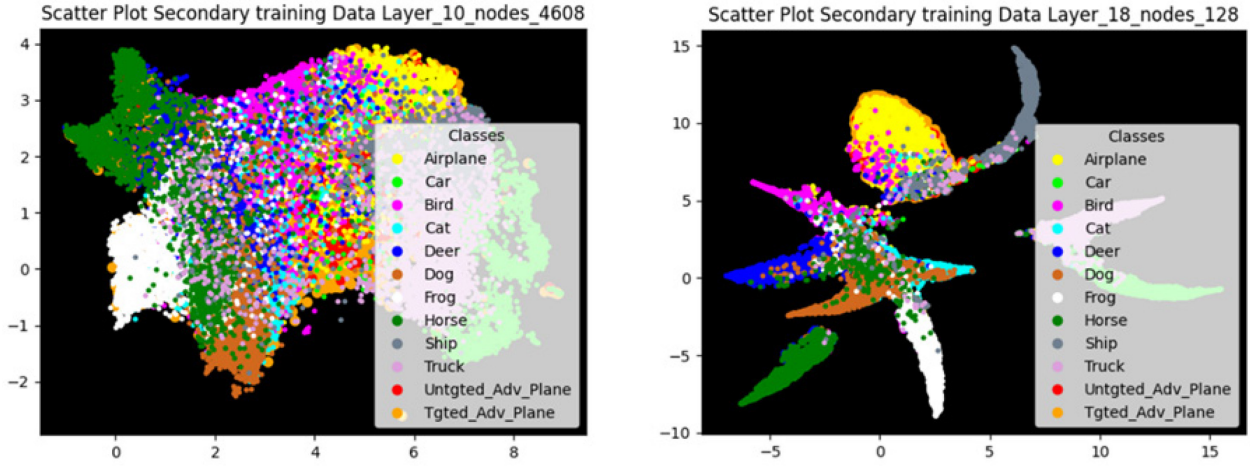

Figure 6. UMAP Plots of Targeted and Untargeted Adversarial Hidden Layer Activations for CIFAR-10 Images: (left) Layer 10 and (right) Layer 18

The images show that, as the CNN learned more about the images in the later layers (Layers 10 and 18), it predominantly categorized and grouped the AEs with actual airplane images. This is in contrast with the clustering in the earlier layers, for example, in layer 4 where the untargeted and targeted AEs were intermingled closely around the car, ship, and truck classes.

At this point, the SGDOneClassSVM code was implemented in Python. The hidden layer activations from the second set of training data were used to train the SGDOneClassSVM ADA. The trained ADA was then implemented on the targeted airplane AEs and captured the predicted image class. The UMAP plots were used to capture how well the ADA accurately detected the targeted airplane AEs as anomalous.

Figures 7 and 8 show visualizations of how well the targeted airplane AEs were detected by the SGDOneClassSVM.

Notice in Figure 7 that the SGDOneClass SVM ADA did not detect the AEs with a high rate of success (in layers 1 and 5). In Figure 8 (layers 11 and 21), the detection accuracy of the SGDOneClass SVM ADA improved as the DNN learned more about the data in the later layers.

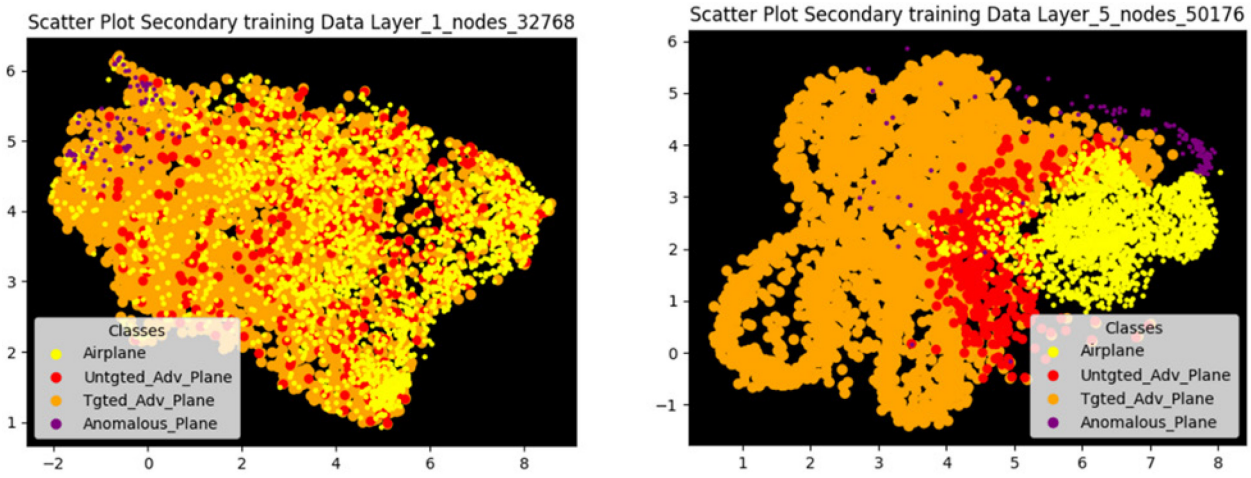

Figure 7. UMAP Plots of Accurate Predictions of Targeted Airplane AEs: (left) Layer 1 and (right) Layer 5

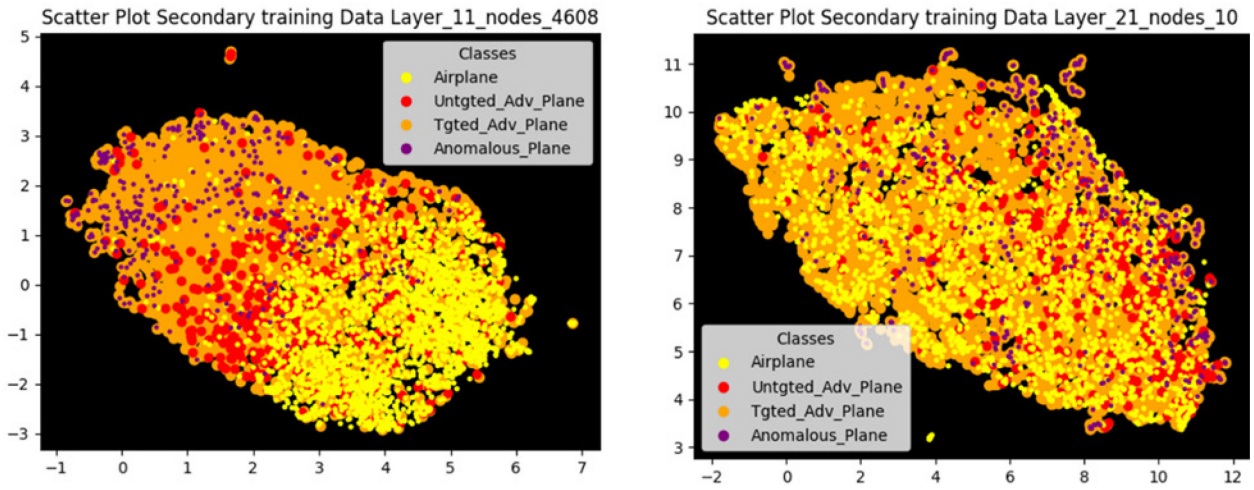

Figure 8. UMAP Plots of Accurate Predictions of Targeted Airplane AEs: (left) Layer 11 and (right) Layer 21

Figure 9 shows the detection accuracy when the results of the DNN output layer (layer 22) are analyzed. Notice that the SGDOneClass SVM ADA was able to detect even more of the AEs.

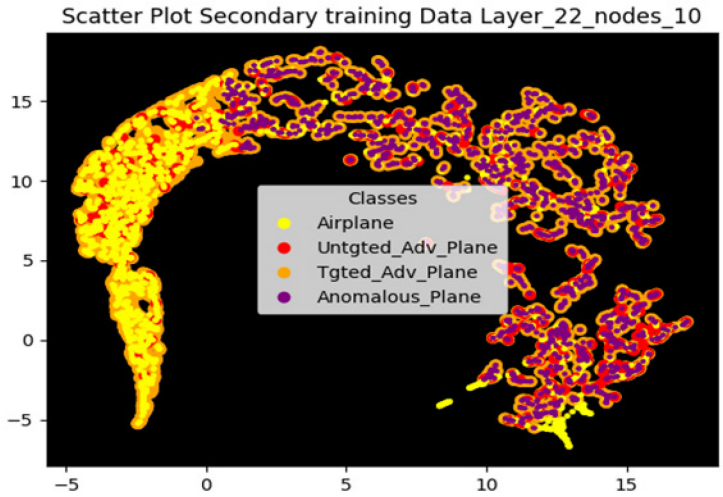

Figure 9. UMAP Plot of Accurate Predictions of Targeted Airplane AEs for CNN Output Layer 22

3. CONCLUSIONS AND FUTURE WORK

This research provided insight into how DNNs interpret AEs compared to normal inputs. A 22-layer CNN was created and trained on the CIFAR-10 dataset to achieve an 83% classification accuracy. Targeted and untargeted AEs were then created from the CIFAR-10 dataset, and images that tricked the network into misidentifying the images of airplanes were analyzed.

It was observed that the network did not recognize the AEs as adversarial, which was expected based on the background research. However, it was not expected that the network would cluster the AEs so closely to the same general region of the class manifold as the actual airplane images for as many layers as it did. The AEs were expected to cluster (a) within a region adjacent to the actual airplane class manifold and (b) primarily for the later layers (example: layers 18–21). The predictions of a DNN are based on the information learned in each layer. The network was expected to cluster the AEs within their original class manifolds in the earlier layers because the malicious perturbations (the distortions added to the original image to make the AE) were not expected to impact how the network made associations and interpreted the data at such an early stage.

The predictions made by the SGDOneClassSVM algorithm were also surprising in that the ADA was expected to detect AEs within the earlier layers, as opposed to the later layers. Also, the ADA was not expected to identify AEs with a high level of accuracy in the output layer, given the low level of accuracy observed in the earlier layers.

For future work, the plan is to quantify the prediction accuracy of the SGDOneClassSVM ADA, implement the One Class Support Vector Machine, Local Outlier Factor, and Isolation Forest ADAs, and compare the accuracy of their detection performances in an attempt to gain further insight on the mathematical strategies that could be used to identify AEs.

REFERENCES

[1] R. Pramoditha, "Two or More Hidden Layers (Deep) Neural Network Architecture," *Medium*, Jan. 1, 2022 [Online]. Available: https://medium.com/data-science-365/two-or-more-hidden-layers-deep-neural-network-architecture-9824523ab903

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

[3] C. Szegedy et al., "Intriguing properties of neural networks," arXiv preprint, arXiv:1312.6199 [cs.CV], 2013.

[4] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[5] Š. Pócoš, I. Bečková, and I. Farkaš, "Examining the Proximity of Adversarial Examples to Class Manifolds in Deep Networks," in *Artificial Neural Networks and Machine Learning – ICANN 2022. Lecture Notes in Computer Science,* E. Pimenidis, P. Angelov, C. Jayne, A. Papaleonidas, M. Aydin, Eds, vol 13532. Springer, Cham, 2022. https://doi.org/10.1007/978-3-031-15937-4_54.

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

[7] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and Rejecting Adversarial Examples Robustly," in *Proc. IEEE Int. Conf. Computer Vision*. IEEE, 2017.

[8] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," arXiv preprint, arXiv:1703.00410 [stat/ML], 2017.