Towards Adversarially Robust DDoS-Attack Classification

Michael Guarino*, Pablo Rivas†, *Senior, IEEE*, and Casimer DeCusatis*, *Fellow, IEEE*
*School of Computer Science and Mathematics
Department of Computer Science, Marist College
†School of Engineering and Computer Science
Department of Computer Science, Baylor University
Email: Pablo_Rivas@Baylor.edu

***Abstract*—On the frontier of cybersecurity are a class of emergent security threats that learn to find vulnerabilities in machine learning systems. A supervised machine learning classifier learns a mapping from x to y where x is the input features and y is a vector of associated labels. Neural Networks are state of the art performers on most vision, audio, and natural language processing tasks. Neural Networks have been shown to be vulnerable to adversarial perturbations of the input, which cause them to misclassify with high confidence. Adversarial perturbations are small but targeted modifications to the input often undetectable by the human eye. Adversarial perturbations pose risk to applications that rely on machine learning models. Neural Networks have been shown to be able to classify distributed denial of service (DDoS) attacks by learning a dataset of attack characteristics visualized using three-axis hive plots. In this work we present a novel application of a classifier trained to classify DDoS attacks that is robust to some of the most common, known, classes of gradient-based and gradient-free adversarial attacks.**

***Index Terms*—adversarial robustness, convolutional neural networks, cybersecurity, DDoS attacks, honeypots, hive plots**

I. Introduction

In quarter 1 of 2020 Amazon Web Services (AWS) reported they defended against a distributed denial of service (DDoS) attack with a peak traffic volume of 2.3 Tbps [1]. At the time this was the largest reported DDoS attack in history just surpassing GitHub's 2018 DDoS attack of 1.7 Tbps. DDoS are characterized as a flood illegitimate network traffic with the intent to exhaust system resources and therefore render the system unable to fulfil legitimate requests. DDoS attacks are considered one of the most serious cybersecurity threats to businesses as they can result in significant and costly damage to IT infrastructure [2]. In a 2018 White House report the United States Council of Economic Advisers estimated that malicious cyber activity cost the U.S. economy between $57 and $109 billion in 2016 [3]. Over the past decade there has been an exponential growth in the volume of DDoS attacks and an increasing severity in the devastation that these attacks cause [4]. With the advent of IoT (internet of things) the annual volume of botnet IoT-based DDoS attacks is expected to grow significantly. The overwhelming nature of DDoS attacks motivates the necessity of automated systems to help cybersecurity professionals analyze trends in network traffic to detect such attacks as they are developing so they may be stopped before they can do significant damage.

Over the past decade Machine learning algorithms have become an important part of many critical applications such as credit card fraud detection, online customer support systems, and email spam filtering. Machine learning algorithms use data to learn important features for making predictions or generating analogies between data points. Neural Networks are machine learning algorithms that have achieved state of the art performance on most computer vision, audio, and natural language processing tasks as well as state of the art performance in many other domains. In our previous work we show that Convolutional Neural Networks (CNNs) are able to classify DDoS attacks from network traffic represented as 3-axis hive plots with strong performance [5]. Neural Networks are very powerful models; however, they have been shown to be vulnerable to adversarial perturbations of the input which cause them to misclassify with high confidence. Adversarial perturbations are small but targeted modifications to the input which cause a trained neural network classifier to fail and pose a risk to cybersecurity applications relying on Neural Networks.

Given the value automated network traffic monitoring systems could provide and the demonstrated ability of Neural Networks to successfully distinguish between normal network traffic and DDoS attacks [5] we believe it is important to work towards reducing the effect that adversarial perturbation have on these machine learning models. In this work we present a novel application of a classifier trained to classify DDoS attacks that is robust to several common known classes of gradient-based [6], [7], [8] and gradient-free attacks [9].

II. Related Work

Given the great success of deep learning on tasks in computer vision, audio, and natural language processing increasing attention is being paid to understanding and improving security vulnerabilities of these algorithms. A widely known vulnerability of neural networks is their susceptibility to adversarially perturbations of the input [6], [10], [11], [12], [13]. Adversarial perturbations are small but targeted modifications to the input

978-1-7281-9656-5/20/$31.00 ©2020 IEEE

often undetected by the human eye, which cause them to misclassify with high confidence.

There have been several successful methods shown to improve the robustness of neural networks against adversarial attacks. A popular approach used by [14], [7] is to incorporate examples perturbed by known adversarial attacks into the training process.

Machine Learning models are iteratively fed batches of data from the larger training dataset which they use to approximate the underlying data generation distribution [15]. By allowing adversarially perturbed data to be part of the training process the model learns a representation of the input that includes attacked data and is therefore robust to the attack.

Another strategy seen in the literature is to preprocess the inputs before sending them through the neural network. [16], [17], [18] investigate preprocessing strategies to discretize and transform the input using a novel application of thermometer encoding, bit-depth reduction, total variance minimization, and image quilting techniques. The motivation here is by adding non-differentiable seemingly random transformations the adversary has a more difficult time learning to find effective attacks.

Much interesting work has been done using applications of manifold theory. A manifold is a point in high-dimensional space that resembles euclidean space. The manifold hypothesis states that data concentrates on lower-dimensional manifolds in the larger input data space and a classification algorithm's job is to learn to distinguish between these manifolds [15]. Some works project input data onto a manifold and work to fix gaps in manifolds which adversarial attackers seem to exploit [19], [20].

Many of the defense strategies against adversarial perturbation presented in the literature do not appear to substantially protect against attack [21], [22], [23], [24] except [7] and to a greater extent [25].

Much of the current literature on adversarial attacks and defenses use MNIST and CIFAR10 datasets to prove the efficacy of their methods. These are computer vision datasets consisting of images of handwritten digits and real world objects respectively and both contain 10 different classes. In this work we transfer domains to cybersecurity and apply the approach of [7] as a method to provide adversarially robust DDoS-attack classification to several common types of gradient-based and gradient-free adversarial perturbations through adversarial training.

III. Dataset

Honeypots are decoys deployed to augment traditional network perimeter defenses that are often designed to gather information about the tactics and motives of black hat attackers. In previous work under support from NSF CAREER Award IIS-1149372 [26], Marist College Security Operations Center (SOC) research students captured network traffic using a set of honeypots during periods where cyber attacks were taking place. Marist College expanded such work by developing its own set of honeypots which have been used to study attack patterns [27]. This attack data was used to create hive plot visualizations of the network traffic patterns for analysis. Hive plot visualizations are heavily used in computational biology research for tasks such as comparing gene regulation and protein-protein interaction [28], [29]. Hive plots are also used in other applications that involve large networks where directly working with a graph structure could be computationally intractable and therefore are a great solution for the use case of analyzing DDoS attack network traffic patterns involving hundreds of machines. To create the Marist DDoS-attack dataset network traffic was recorded over a two minute time period and evenly divided into 8 different timesteps (samples shown in Fig. 1) from the initial starting point of the attack to the completion of the attack. No great care was taken to determine the optimal number of timestamps to use; however, future work that leverages time to improve the confidence of their predictions maybe find value in this. The hive plots in the Marist DDoS-attack dataset have three axes. The first (top) axis plots the time elapsed since the beginning of the attack where the center is the moment the attack started, the second (right) axis is the source IP address of the attacker, the third (left) axis is the country corresponding to the source IP address of the attacker.

The Marist DD0S-attack dataset has two classes describing if the hive plot of the network traffic is that of a DDoS-attack or normal legitimate network traffic. This dataset contains 2,000 total examples and the class distribution is balanced meaning there are 1,000 8-timestep sequences of legitimate network traffic and 1,000 8-timestep sequences of DDoS-attack patterns.

IV. Methods

A. Problem Setup

We consider the task of making a DDoS-attack classifier robust to adversarial perturbations using a dataset consisting of hive plot images. The hive plots visualize attack patterns over time from source to target IP address. In prior work by [5] this dataset has been used to perform binary classification of DDoS-attacks using a Convolutional Neural Network (CNN) specifically the ResNet34 [30] architecture with great success.

CNNs were first introduced in [31] and have become the workhorse of modern computer vision. CNNs work by running a filter over the input applying the cross correlation operator then extracting the most meaningful feature representations using pooling operations, and finally these feature representations are passed through a fully connected neural network which performs the classification. These are some of the basic operations often chained together in a variety of ways by different CNN architectures to solve various computer vision problems or problems assuming that patterns exist invariant to scale, shift, rotation, and spatial factors. Building off [5] we use the ResNet34 [30] and MobileNetV2 [32] architectures as our baseline classifier which performs binary classification of DDoS-attacks. These architectures were chosen because they perform well and have received heavy adoption from the computer vision community. These CNN architectures serve

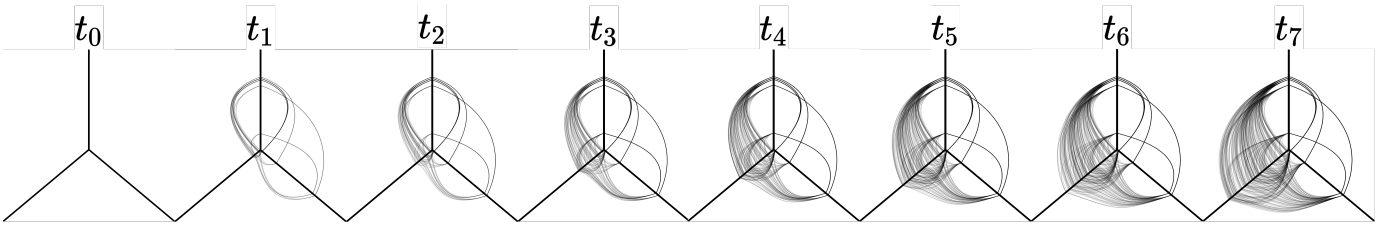


Fig. 1. Starting from the left (t_0) this is an example of DDoS-attack network traffic collected over a two minute period and evenly divided into 8-timestep sequence and visualized in hive plots. The final timestep (t_7) is on the right of this figure. Note that the hive plots in the Marist DDoS-attack dataset have three axes where the top axis shows time elapsed since the beginning of the attack, the right axis is the source IP address of the attacker, the left axis is the country corresponding to the source IP address of the attacker.

as a great place for evaluating the efficacy of a method to improve robustness to adversarial perturbations although many other architectures would be perfectly suited as well.

Supervised Machine Learning aims to learn an underlying function to map input data x to some associated label y. This is accomplished by feeding input data x into the supervised machine learning model to obtain the predicted output $\hat{\mathbf{y}}$ and then measuring the error against y. Once the error is obtained the model parameters θ are updated by some optimization strategy (stochastic gradient descent) in a direction to reduce the error. The training process iterates through these steps with the newly found model parameters seeking to find model parameters that produce the lowest possible error. Neural Networks use the backpropagation algorithm [33] which calculates the partial derivative of the loss with respect to the learned parameters by recursively applying the chain rule of calculus through every layer of the network yielding a Jacobian at every layer. Gradients are obtained by taking the Jacobian-Gradient product. The gradient is a vector that points in the direction of steepest descent. These gradients are then used to move the parameters θ of the Neural Network in a direction such as to lower the error.

To formalize the problem of making a classifier robust to adversarial perturbation (attack) let's first consider a standard classification task with an underlying data distribution D over examples $\boldsymbol{x} \in \mathbb{R}^d$ and their corresponding labels $y \in \{c_1, c_2\}$ whose goal is to minimize the loss $L(\boldsymbol{\theta}, \boldsymbol{x}, y)$ where θ is the set of model parameters [7]. We can minimize the loss in terms of its expectation:

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[L(\boldsymbol{x}, y, \boldsymbol{\theta})] \tag{1}$$

Our goal is to find the model parameters θ that minimize the risk of successful attack [7]. An attacker takes an example $\boldsymbol{x}$ belonging to class c_1 as input and learns to find $\boldsymbol{x}^{\text{adv}}$ such that $\boldsymbol{x}^{\text{adv}}$ is very similar to x but the trained classifier incorrectly classifies $\boldsymbol{x}^{\text{adv}}$ as belonging to class c_2 where $c_2 \neq c_1$ [7]. In order to specify an attack model (adversary) we introduce a set of allowable perturbations $\mathcal{S} \subseteq \mathbb{R}^d$ that formalizes the manipulative power of the adversary and $\mathcal{S}$ is chosen such that it is perceptually similar to the original hive plot image [7]. In order for our classifier to become resistant to adversarial perturbations (attacks) we allow samples from the data distribution D to be perturbed by an adversary before being sent into the loss function giving rise to this saddle point problem:

$$\min_{\boldsymbol{\theta}} \rho(\boldsymbol{\theta}), \tag{2}$$

where

$$\rho(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\boldsymbol{\theta}, \boldsymbol{x}+\delta, y)\right]. \tag{3}$$

The saddle point can be viewed as a composition of an inner maximization and outer minimization problem. The inner maximization problem aims to find an adversary that can maximize the loss via the attacking neural network and the outer minimization problem is to find model parameters θ such that the overall loss is minimized despite the best efforts of the inner maximization problem (adversary) [7].

The saddle point problem formulated above should establish a clear view of the goals of this paper. Building on the work of [5] and using techniques from [7] we aim to learn a DDoS-attack classifier robust to a wide variety of adversarial perturbations (attacks).

The adversarial perturbations selected for use in this work were chosen based on their popularity in the literature and an attempt to capture a wide variety of different mechanics used in their attack strategies. We consider two different general classes of attack strategies: gradient-based attacks and gradient-free attacks.

Gradient-based attacks exploit the use of the gradient in machine learning algorithms that use gradient based learning such as Neural Networks. Gradient-based attacks can be thought of as white-box attacks in that the attacker has complete access the model it is attacking. Gradient-based attacks considered in this work are the Gradient Sign Attack [6] and Projected Gradient Descent Attack algorithms [7]. Gradient-free attacks do not need access to the gradient used during the training process and rather rely solely on targeted transformations to the input for their attacks.

Gradient-free attacks created adversarial perturbations without any internal knowledge of the model they are attacking. These attacks can be seen as black-box attacks treating the model as an orcale that is feed a perturbed input and receives the corresponding output. In our work we consider one gradient-free attack: the Single Pixel Attack [9].

B. Gradient Sign Attack

The Gradient Sign Attack also known as the "Fast Gradient Sign method" (FGSM) [6] uses the gradient obtained during backpropagation to create an adversarial perturbation. FGSM uses the gradient of the loss with respect to the input to form the inner maximization of the loss in the saddle point problem introduced above. The perturbation used in FGSM can be formulated as

$$\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} L(\boldsymbol{\theta}, \boldsymbol{x}, y)\right) \tag{4}$$

where $\boldsymbol{x}$ is the input, y is the associated label, ϵ is a coefficient, bound to $\|\boldsymbol{\eta}\|_\infty < \epsilon$, used to scale the size of the perturbation, and $L(\boldsymbol{\theta}, \boldsymbol{x}, y)$ is the associated loss. This perturbation is then added to the original input $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$ to create the adversarial perturbation used to attack the classifier. The intuition behind this method is that the gradient of the loss with respect to the input image allows for a measure of how each pixel contributes to the loss value and therefore allows each pixel to be adversarially perturbed in a direction to maximize the loss. This method can be considered a simple one-step scheme in that it only considers current examples in the formulation of the perturbation.

C. Projected Gradient Descent Attack

The Projected Gradient Descent attack (PGD) introduced in [7] is a gradient-based white box attack very similar to FGSM; however, unlike FGSM this method is iterative and calculates a new gradient every iteration and adds a smaller perturbation to each point in the given batch of examples. This is formulated as:

$$\boldsymbol{x}^{t+1} = \Pi_{\boldsymbol{x}+S}\left(\boldsymbol{x}^t + \alpha \operatorname{sign}\left(\nabla_{\boldsymbol{x}} L(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)\right) \tag{5}$$

where $\boldsymbol{x}$ is the input, y is the associated label, α is a constant used to scale the affect of the gradient on the perturbation, t is the current iteration, S is the set of allowable perturbations, and $L(\boldsymbol{\theta}, \boldsymbol{x}, y)$ is the associated loss. One view of PGD is that it is a cumulative form of FGSM; taking into account examples from previous iterations.

D. Single Pixel Attack

The Single Pixel attack is a gradient-free, black box attack first introduced in [9]. This method relies solely on a targeted perturbation to a single pixel. A critical pixel is a perturbed pixel that causes a trained nerual network to misclassify. In our work we consider Algorithm 1 of the Single Pixel Attack introduced in [9] although there have been recent improvements to the Single Pixel Attack theory in [34]. Algorithm 1 of the Single Pixel Attack in [9] introduces a perturbation function $\mathrm{PERT}(I, p, x_1, x_2)$ which takes image I as input, a perturbation parameter $p \in \mathbb{R}$, a location (x_1, x_2), and outputs an image $I_p^{(x_1,x_2)} \in \mathbb{R}^{\ell \times w \times h}$ that can be defined as:

$$I_p^{(x_1,x_2)}(b,u,v) \stackrel{\text{defn}}{=} \begin{cases} I(b,u,v) & \text{if } x_1 \neq u \text{ or } x_2 \neq v \\ p \times \operatorname{sign}(I(b,u,v)) & \text{otherwise.} \end{cases} \tag{6}$$

This means the image $I_p^{(x_1,x_2)} = \mathrm{PERT}(I, p, x_1, x_2)$ will be exactly the same as image I for every pixel except the pixel at $(*, x_1, x_2)$ which will have a value of $p \times \operatorname{sign}(I(\star, x_1, x_2))$ [9]. As part of Algorithm 1 (RandAdv) presented in [9] uses a simple search procedure to pick the location (x_1, x_2) in image I and applies the $PERT$ function defined above to obtain the perturbed image $I_p^{(x_1,x_2)}$. The perturbed image is then sent through the trained classifier. If the perturbed image produces a misclassification a critical pixel has been identified.

All implementation was done using the PyTorch python API [35] and all adversarial attacks were implemented in [36]. PyTorch was selected as the deep learning framework because of it's easy to use dynamic computation graph, automatic differentiation (autograd), ability to accelerate computation on the GPU, and high quality implementations of ResNet34 and MobileNetV2. AdverTorch was selected as the adversarial toolbox used for the experiments in this work over the popular foolbox [37] and cleverhans [38] because AdverTorch attack implementations appear to be closer to the original papers than foolbox and cleverhans only provides TensorFlow implementations. PyTorch Ignite [39] was used abstract out the training loop and general machine learning boilerplate code such as model checkpointing (saving model states throughout the training process), generating performance metrics, and logging summaries of model progress.

V. Experiments

In this section we present the findings of this investigation into creating a DDoS-attack classifier robust to several methods of adversarial perturbation on the Marist DDoS-attack dataset. In order to demonstrate the efficacy of the method applied we used several popular different neural network architectures. This is done to show that regardless of neural network architecture and type of adversarial perturbation this method for creating a robust classifier works. The criteria used to determine if a model is considered adversarially robust is that the model's performance will not suffer when adversarial perturbations are present.

We start by first training a classifier on the Marist DDoS-attack dataset. This is a binary dataset; however, we use a softmax cross entropy loss function. This is a popular loss function heavily used in deep learning for multi-class classification problems but is also completely appropriate in binary classification problems. We also use a learning rate scheduler set to lower the learning rate as the loss lowers. A learning rate is a hyperparameter (tuning nob) that governs the size of the step taken during the training process as the optimization strategy, stochastic gradient descent in this case, updates model parameters in order to lower the loss.

We run all experiments with two different neural network architectures: ResNet34 and MobileNetV2. Both of these models require input images to be of size 400 width and 400 height with 3 color channels. We resize the images in the Marist DDoS-attack dataset using binlinear interpolation and also normalize the pixel values by their color channel using a mean of 0.5 and a standard deviation of 0.5 in order

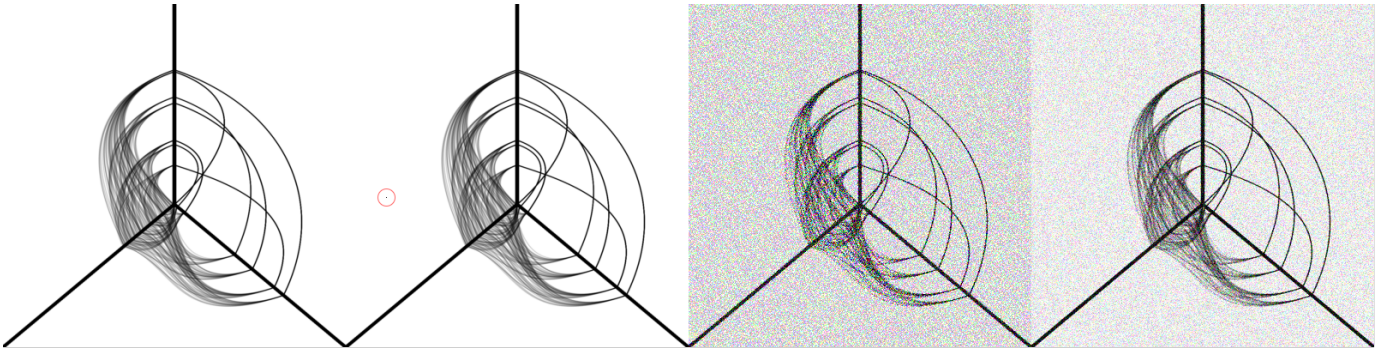


Fig. 2. These are hive plot visualizations from a single timestep sent through different forms of adversarial perturbation. Far left (a) is the original non-perturbed hive plot image, (b) is the single pixel attack, (c) gradient sign attack, and (d) projected gradient descent attack.

to improve numeric stability. No great care was taken in the selection of these values we just wanted to ensure the values were normalized in some standard fashion. No other preprocessing transformations are performed as they would not be appropriate for this data.

In order to get a sense of how the classifier, without an adversary, fits the dataset we first train the classifier for 50 epochs with a batch size of 32 samples per batch. When training a machine learning classifier data is chunked into batches of n examples and sent into the model one batch at a time in an iterative fashion. When you have iterated through every unique batch in your training dataset you are said to have completed an epoch. When this experiment was performed we observed that the classifier could perform well on the dataset and it took about 20 epochs before we observed peak achievable performance. We also ran a few experiments comparing the results with batch sizes of 16, 32, 64, and 128. Changing the batch size can affect model performance as the model updates it's parameters on every batch. If your batch size is large the effect of the errors made on every example in the batch experiences a dilution effect and therefore smaller batch sizes are preferable when solely considering model performance on the training set; however, smaller batch sizes are less able to be parallelized on the GPU and therefore result in longer training times. We found no significant difference in model performance (ability to lower loss) between training regimes with batch sizes of 16, 32, 64.

In order to measure model performance we observed the loss during training as well as precision and recall. Precision $= \frac{TP}{TP+FP}$ gives us a measure of specificity or how well our classifier is at distinguishing between the two classes where TP is the number of true positives and FP is the number of false positives. Recall $= \frac{TP}{TP+FN}$ provides a measure of sensitivity or how well our classifier is at retrieving only the correct class. Note for both precision and recall the highest score is 1.0 indicating the model is excellent at retrieving only the correct class and is very good at distinguishing between the two classes. Precision and recall are useful metrics in evaluating model performance and are not sensative to class imbalance like accuracy or area under receiver operating characteristic curve AUC-ROC. Note that the Marist DDoS-attack dataset is a balanced dataset meaning that there are the same number of examples from both classes.

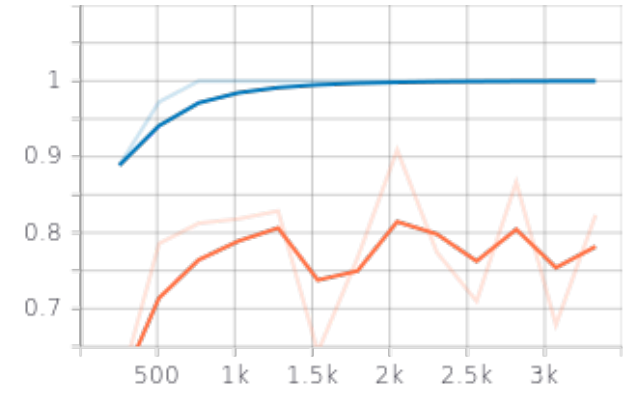
The training regime used for all experiments was to train the model for 20 epochs with a batch size of 64 we ran 5 replicas for each experiment and took the average along with standard deviation for a measure of central tendency. Experiments performed were: train a classifier with no adversarial perturbation, train a classifier with GSA gradient-based adversarial perturbation, train a classifier with PGD gradient-based adversarial perturbation, train a classifier with gradient-free Single Pixel Attack adversarial perturbation, and then train with all 3 adversarial perturbations simultaneously. In order to determine if robustness to several different methods of adversarial perturbations could be achieved simtaneously by a single classifier we allowed all three methods of adversarial attack to be present during the training process in the *All* experiment shown in Table 1. Adversarial perturbations were introduced to the model every 3rd batch during the training process. This seemed appropriate for the model to learn the variety of different adversarial representation of the input.

Shown in Table 1 are the validation results on all experiments achieved using the training regime described above. We observe that model performance is very slightly improved when no adversarial perturbations were part of the training process. This however, is not significantly different enough from models trained with adversarial perturbations and therefore we can say that this method is effective at improving a classifier's robustness to adversarial perturbation. Furthermore adversarial perturbations add a significantly larger variety of configurations of the input that the model must learn to associate with a given class. When comparing the validation performance of models trained to be robust to adversarial perturbation with a standard classifier if there is no significant difference in performance the model can be said to be adversarially robust to a given method of adversarial perturbation. The performance difference between different methods of adversarial perturbation showed no significant

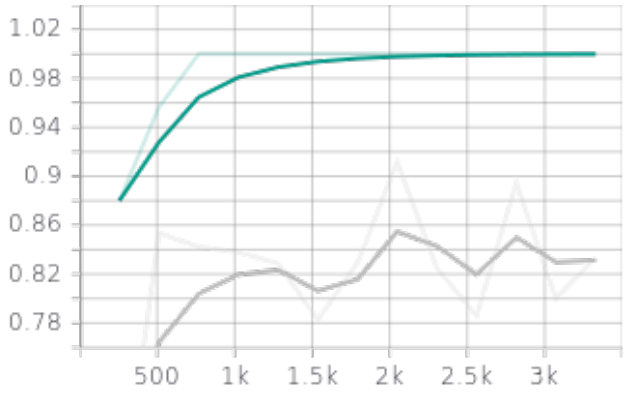
TABLE I

RESULTS FROM ALL EXPERIMENTS PERFORMED IN THIS WORK CONTAINED IN THE TABLE ABOVE. ALL EXPERIMENTS RUN ON THE MARIST DDOS-ATTACK CLASSIFICATION DATASET. THESE ARE VALIDATION RESULTS AFTER 20 EPOCHS OF TRAINING WITH A BATCH SIZE OF 64 USING STANDARD STOCHASTIC GRADIENT DESCENT. REPORTED IS AN AVERAGE OF 5 REPLICAS WITH STANDARD DEVIATION.

Classifier Architecture	Attack Method	Precision	Recall	Loss
Resnet34	No Attack	0.919 +/- 0.0258	0.903 +/- 0.0320	0.255 +/- 0.0457
Resnet34	GSA	0.915 +/- 0.0291	0.900 +/- 0.0359	0.249 +/- 0.0297
Resnet34	PGD	0.911 +/- 0.0111	0.891 +/- 0.0156	0.263 +/- 0.0279
Resnet34	SPA	0.910 +/- 0.0273	0.884 +/- 0.0406	0.288 +/- 0.0419
Resnet34	All	0.928 +/- 0.0292	0.922 +/- 0.0322	0.187 +/- 0.0387
MobileNetV2	No Attack	0.931 +/- 0.0253	0.922 +/- 0.0292	0.249 +/- 0.0285
MobileNetV2	GSA	0.912 +/- 0.0090	0.891 +/- 0.0156	0.284 +/- 0.0229
MobileNetV2	PGD	0.920 +/- 0.0253	0.902 +/- 0.0410	0.239 +/- 0.0258
MobileNetV2	SPA	0.907 +/- 0.0151	0.884 +/- 0.0236	0.279 +/- 0.0194
MobileNetV2	All	0.932 +/- 0.0151	0.918 +/- 0.0198	0.276 +/- 0.0113



(a)



(b)

Fig. 3. ResNet34 trained with no adversarial perturbation validation (a) recall by class (b) precision by class. These plots are from a single replica with no adversarial perturbation. Note that one class is being modeled exceptionally well while the other seemed to be modeled less effectively.

difference suggesting that the classifier can become robust to both gradient-based, gradient-free, and simultaneously both gradient-based and gradient-free adversarial perturbations regardless of the method used.

Fig. 3 shows the validation recall and precision by class for a single replica of training of the ResNet34 architecture trained without adversarial perturbation. The behavior suggests that one class is being modeled very well while the classifier has a more difficult time modeling the other class. Upon further investigation this behavior was found not to be anomalous and regardless of replica or experiment type (any combinations of classifier architecture and attack method) resulted in similar behavior. This prompts improving the model architecture such that it can sufficiently model both classes perhaps even leveraging temporal information provided by the 8 timesteps in the Marist DDoS-attack dataset. Improving the model performance is important as it will likely yield a model that can learn to be even more robust to a variety of different adversarial perturbations.

VI. CONCLUSION

In this work we build off [5] and work to find DDoS-attack classifier that is robust to adversarial perturbations. Existing methods for creating a model robust to adversarial perturbation have been shown effective on computer vision datasets such as MNIST and CIFAR10. These methods typically only consider robustness against a single method of attack. In this work we transfer tasks to DDoS-attack classification using hive plot images. We leverage methods used by [7] in order to create classifiers robust to both gradient-based white-box attacks, gradient-free black-box attacks, and simtaneously both gradient-based and gradient-free attacks. We show that by including adversarial perturbations in the training process based on the method introduced in [7] we are able to achieve robustness on a variety of different attacks. This is important work as machine learning systems are starting to be at the center of many critical applications. Machine learning systems monitoring plots of data for abnormalities are susceptibile to a variety of known attacks in the form of adversarial perturbations. We hope this could be an important starting point towards building such machine learning systems robust to many such attacks and worth further investigation.

The code to reproduce our experiments can be freely accessed under an MIT license on this github repository:

`https://github.com/mguarin0/advrbtddos`

ACKNOWLEDGMENT

We acknowledge the support of Marist College's School of Computer Science & Mathematics, and the Rivas AI lab.

REFERENCES

[1] J. Porter, "Amazon says it mitigated the largest ddos attack ever recorded," *The Verge*, 2020.

[2] N. Vlajic and D. Zhou, "Iot as a land of opportunity for ddos hackers," *IEEE*, vol. 51 (7), no. 26-34, 2018.

[3] W. H. C. of Economic Advisers, "The cost of malicious cyber activity to the u.s. economy," 2018.

[4] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, 2017.

[5] P. Rivas, C. Decusatis, M. Oakley, A. Antaki, N. Blaskey, S. Lafalce, and S. Stone, "Machine Learning for DDoS Attack Classification Using Hive Plots," *2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2019*, pp. 0401–0407, 2019.

[6] J. S. &. C. S. Ian J. Goodfellow, "Explaining and Harnessing Adverserial ML," *International Conference on Learning Representations (ICLR)*, pp. 1–11, 2015.

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–28, 2018.

[8] C. Xiao, J. Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[9] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," *arXiv preprint arXiv:1612.06299*, 2016.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pp. 1–10, 2014.

[11] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[12] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," 2015.

[13] N. Papernot, "On the Integrity of Deep Learning Systems in Adversarial Settings," no. May, pp. 1–53, 2016.

[14] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[16] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018.

[17] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse dnns with improved adversarial robustness," in *Advances in neural information processing systems*, 2018, pp. 242–251.

[18] V. M. Kabilan, B. Morris, H.-P. Nguyen, and A. Nguyen, "Vectordefense: Vectorization as a defense to adversarial examples," in *Soft Computing for Biomedical Applications and Related Topics*. Springer, 2018, pp. 19–35.

[19] B. Lindqvist, S. Sugrim, and R. Izmailov, "AutoGAN: Robust Classifier Against Adversarial Attacks," *arXiv preprint arXiv:1812.03405*, 2018. [Online]. Available: http://arxiv.org/abs/1812.03405

[20] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," in *International Conference on Learning Representations*, 2018.

[21] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations*, 2018.

[22] A. Athalye and N. Carlini, "On the robustness of the cvpr 2018 white-box adversarial example defenses," *arXiv preprint arXiv:1804.03286*, 2018.

[23] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang, "Adversarial domain adaptation with domain mixup," *arXiv preprint arXiv:1912.01805*, 2019.

[24] H. Qiu, C. Xiao, L. Yang, X. Yan, H. Lee, and B. Li, "Semanticadv: Generating adversarial examples via attribute-conditional image editing," *arXiv preprint arXiv:1906.07927*, 2019.

[25] L. R. work) Schott, "Towards the First Adversarially Robust NN Model on Mnist," *Iclr*, vol. 3, pp. 1–16, 2019.

[26] V. Joseph, P. Liengtiraphan, G. Leaden, and C. DeCusatis, "A software-defined network honeypot with geolocation and analytic data collection," in *Proceeding of 12th annual IEEE/ACM information technology professional conference, Trenton, NJ (March 17, 2017)*, 2017.

[27] D. N. Gisolfi, M. Gutierrez, T. V. Rimaldi, C. DeCusatis, and A. G. Labouseur, "A honeynet environment for analyzing malicious actors," 2018.

[28] M. Krzywinski, K. Kasaian, O. Morozova, I. Birol, S. Jones, and M. Marra, "Linear layout for visualization of networks," in *Genome Inform*, 2010.

[29] M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra, "Hive plots—rational approach to visualizing networks," *Briefings in bioinformatics*, vol. 13, no. 5, pp. 627–644, 2012.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[33] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323 (6088), no. 533-536, 1986.

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

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[36] G. W. Ding, L. Wang, and X. Jin, "Advertorch v0. 1: An adversarial robustness toolbox based on pytorch," *arXiv preprint arXiv:1902.07623*, 2019.

[37] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," in *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. [Online]. Available: http://arxiv.org/abs/1707.04131

[38] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[39] V. Fomin, J. Anmol, S. Desroziers, Y. Kumar, J. Kriss, A. Tejani, and E. Rippeth, "High-level library to help with training neural networks in pytorch," https://github.com/pytorch/ignite, 2020.