Enhancing the Robustness of Deep Neural Networks Against Security Threats Using Radial Basis Functions

By

Matthew P. Burruss

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

May 8, 2020

Nashville, Tennessee

Approved:

Abhishek Dubey, Ph.D.

Xenofon Koutsoukos, Ph.D.

DEDICATION

To BPL, gone too soon.

ACKNOWLEDGMENTS

Thank you to my advisor Abhisheky Dubey Ph.D. and colleague Shreyas Ramakrishna for supporting me throughout my undergraduate career at Vanderbilt and guiding me throughout my Master's program. Their knowledge and professionalism throughout my time in their lab has inspired my curiosity in computer science and pushed me to continue to learn more about the field. I am also thankful for the Institute of Software Integrated Systems for allowing me to be a part of the research community over the past 3 years.

I would like to thank my family, especially my parents who financially and lovingly supported me throughout college and during this degree program. They have always been by my side and I am forever aware and grateful of their support. I would also like to thank my grandparents, all of which inspire me to be my best. Finally, I would like to thank all of my friends at Vanderbilt that I have made over the past four years during my undergraduate career and who have been an integral part of my Vanderbilt experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC Activation Clustering

API Application Programming Interface

DNN Convolutional Neural Network

DNN Deep Neural Networks

GTSB German Traffic Sign Benchmark (Data Set)

LEC Learning Enabled Components

OOD Out-of-Distribution

RBF Radial Basis Function

CHAPTER I

Introduction

I.1 A Quick Overview of Security Threats Facing Deep Neural Networks

Today, Cyber-Physical Systems (CPS) have started to rely on learning enabled components (LECs) as part of the control loop in autonomous tasks. Commonly, machine learning approaches like deep neural networks (DNNs) are used as LECs due to their success in a variety of complicated tasks [1]. For example, NVIDIA's DAVE-II convolutional neural network (CNN) architecture has been used as an LEC to provide steering controls for autonomous vehicles [2]. Tesla's autonomous driving has also recently completed 2 billion driving miles [3] using several LECs to perform object detection and image segmentation [4]. Additionally, DNNs have been used in aircraft collision systems to reduce the policy search [5]. However, despite the success of DNNs as LECs, it is still difficult to assure their correctness. For example, in 2016 a fatal incident occurred when Tesla autopilot failed to recognize an incoming trailer and crashed [6]. Additionally concerning are the plethora of algorithms that could target a CPS system by exploiting a corner case [7], poisoning the underlying DNN [8], or producing adversarial instances [9].

Typically, algorithms to produce these security threats differ in their goal and knowledge of the DNN [9]. For example, an attack can be *non-targeted* to induce a general misprediction or *targeted* to induce a misprediction of a particular label. A *white-box* attack has access to the model architecture and may be able to adjust model parameters or modify the training procedure whereas a *black-box* attack has no knowledge or access to the base DNN. Black-box attacks instead rely on the transferability of an attack from one DNN to another or exploiting a known corner case to accomplish their goals [10]. Finally, security threats can be an intentional result of an attack or an unintentional result of the environment e.g. out-of-distribution (OOD) data.

OOD data or anomalies pose a risk to CPS by potentially causing the LEC to behave unexpectedly. This type of vulnerability was partly responsible for the fatal Tesla 2016 crash. In the most extreme case, when fed an unrecognizable image, like random noise, a DNN may respond with a consistent high-confidence prediction [11]. There are, however, more realistic OOD examples. For example, in the self-driving simulator CARLA, it was experimentally shown that a shadow resembling a lane that is projected across a road can induce crash scenarios [7]. This type attack can be referred to as a *black-box physical attack* and is successful mainly due to the fact that DNNs are unable to reliably lower their confidence to the OOD data [11]. However, simple two-layer radial basis function (RBF) networks (see Figure I.1) require a dense amount of highly activated features to make a confident prediction and can therefore reliably reject anomalies [10]. We hypothesize that a deep RBF classifier can use this property to successfully reject real-time anomalies like those shown on the CARLA simulator.

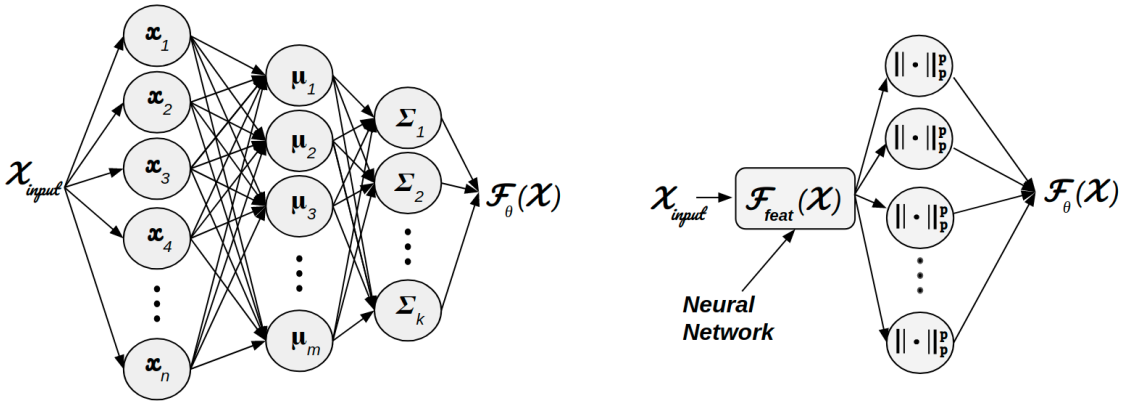


Figure I.1: (Left) A simple two-layer RBF network and (right) a deep RBF network. The simple two-layer network has a single hidden layer where the j^{th} RBF unit operates on the input using $e^{-\beta||X-\mu_j||_p^p}$ and the output is a weighted sum of the RBF units per class $k \in \{1,...,c\}$. The deep RBF network consists of a DNN feature extractor F_{feat} that extracts salient features $F_{feat}(x)$ from the input which then undergo a high dimension ℓ-p norm distance operation that compares $F_{feat}(x)$ to the learned class prototypes stored in the weights of each RBF unit. The prediction is the the prototype nearest to the extracted features.

A newer and even more pervasive threat to DNNs is a data poisoning attack where an attacker modifies the training procedure, alters the model's logic, or manipulates labels in

the training data set to encode a backdoor key that can be exploited in production. For example one data poisoning attack on a facial recognition system allowed an individual to impersonate another by simply wearing sunglasses [12]. Poisoning attacks mainly pose a threat to the training pipeline as data sets are often pulled from an unreliable online source or models are pretrained using unknown weights [13]. The most popular data poisoning attacks focus on injecting poisoned samples into the training data set by encoding a backdoor key into a clean image and manipulating the ground truth label so that when the backdoor key is present in a test image the poison target label will be predicted [12]. Concerningly, a poisoned model may succumb to the attack while maintaining high confidence on clean data. However, we show that deep RBFs can generalize better than regular DNNs due to their strict encoding of the feature space of the data and are therefore harder to attack by a poisoning attack. This discovery combined with the deep RBF's ability to reduce its confidence to anomalous data is the basis for our RBF outlier detection method which can effectively clean a sparsely poisoned data set without relying on a verified, clean data set.

Finally, adversarial attacks add imperceptible non-random perturbations to an input to cause high-confidence mispredictions [14]. Adversarial examples exploit the linear property of DNNs rather than their non-linearity. For example, the Fast Gradient Sign Method produces adversarial images by linearizing the loss function in ℓ_∞ neighbourhood and solving a closed form equation to generate an adversarial image [10]. Since this initial discovery, several other powerful white-box algorithms have been proposed to generate adversarial instances [9, 10, 15–17]. Not only can these attacks generalize as black-box attacks [10], but advanced attacks like the Carlini & Wagner attack can adapt to and evade defense mechanisms [18]. Two-layer RBF networks are known to be resistant to adversarial images in that they respond to adversarial images with a decrease in confidence [10]. However, two-layer RBF networks lack the capacity to generalize to complicated tasks. Only, recently has it been shown that a deep RBF can be successfully trained on MNIST data set [19]; however, to the best of our knowledge it has yet to be shown whether or not a non-trivial

example of a deep RBF is robust to adversarial attacks; therefore, in order to argue that a deep RBF can be used in CPS we extend the evaluation on adversarial attacks on non-trivial examples.

I.2 Major Contributions

We address the security concerns threatening LECs that use DNNs, specifically OOD black-box physical attacks, white-box adversarial attacks, and data poisoning attacks, through the following contributions:

1. A scalable and real-time OOD black-box physical attack detection mechanism implemented on an end-to-end autonomous RC vehicle utilizing a deep RBF to simultaneously provide steering angles and a rejection class.

2. Experimental evidence and theoretical discussion on why a deep RBF is less susceptible to a data poisoning attack than a traditional softmax classifier.

3. A novel RBF outlier detection method which can be used to clean a sparsely poisoned data set without relying on a verified, clean data set.

4. An analysis of the robustness of a production-grade deep RBF classifier on a non-trivial task against a variety of state-of-the-art white-box adversarial attacks to show that deep RBFs can viably be used in CPS systems.

I.3 Outline

Chapter II explains how to design a deep RBF. Section II.2 discusses the state-of-the-art defense mechanisms to defend against the previously introduced security threats. Chapter III explains and evaluates the integrated OOD black-box attack detection mechanism offline and online using the DeepNNCar autonomous RC car testbed. Chapter IV describes two data poisoning attacks to explain why deep RBFs are less susceptible than traditional DNNs and compares our novel RBF outlier detection mechanism to the state-of-the-art

data poisoning cleaning method. Chapter V evaluates a production-grade deep RBF on a variety of white-box adversarial attacks and compare its robustness to a regular DNN. Section VI.1 summarizes the findings and describes potential areas of future work using deep RBFs. Finally, our conclusions are presented in Section VI.2.

CHAPTER II

Background

II.1 Deep RBF Network

In its simplest form, a RBF network is a two-layer neural network with a single fully-connected hidden layer and an output layer that uses RBFs as the activation function. The following section defines deep RBFs based on previous work [19].

II.1.1 *Definition*

A RBF is a real-valued function that measures the distance of an input x to some prototype vector. The similarity measure can be captured in the following definition of a RBF unit using ℓ_p-norm distance where $A \in \mathbb{R}^{n \times l}$, $b \in \mathbb{R}^l$, $x \in \mathbb{R}^n$ and $l \leq n$ [19].

$$\phi(x) = (||A^T x + b||_p)^p \tag{II.1}$$

In the context of deep learning, RBF units can be applied to the high-level features $f(x)$ extracted by the model from the raw input x in order to classify the input into k classes such that $k \in \{1, .., c\}$. Using the Euclidean metric and allowing $A = \mathbb{I}_n$, the deep-RBF unit is defined as follows

$$\phi_k(x) = (||f(x) - W_k||_2)^2 \tag{II.2}$$

where $W_k \in \mathbb{R}^{|f(x)|}$ is a trainable weight vector intuitively representing the learned prototype of class k. A prediction is therefore the prototype nearest to the input. We have found that in practice, applying hyperbolic tangent function to the features $f(x)$ preceding the RBF layer and randomly initializing $W \in [-1, 1]$ achieves sufficient model performance.

II.1.2 *Training a Deep RBF*

A metrics-learning inspired loss function named *SoftML* has been proposed to avoid the vanishing gradient problem of deep RBFs [19].

$$J_{SoftML} = \sum_{i=1}^{N} (\phi_{y_i}(x^{(i)}) + \sum_{j \notin y_i} log(1 + e^{(\lambda - \phi_{y_i}(x^{(i)}))})) \tag{II.3}$$

where y_i is the correct class of input $x^{(i)}$. We select $\lambda = 0.5$ because it has been previously noted that λ has little effect on convergence [19].

II.1.3 *Interpreting the Deep RBF Output*

Advantageously, it was shown that Eq. II.3 can be interpreted as the negative log-likelihood [19]. Therefore, the outputs can be interpreted as non-normalized probabilities following the transformation below.

$$P(y = k|x) = \frac{e^{-\phi_k(x)}(1 + e^{\lambda - \phi_k(x)})}{\prod_j (1 + e^{\lambda - \phi_k(x)})},\ k \in \{1, 2, ..., c\} \tag{II.4}$$

As such, a rejection class $k = 0$ can then be defined to capture the probability that x belongs to no class in $\{1, 2, ..., c\}$.

$$P(y = c + 1|x) = \frac{1}{\prod_j (1 + e^{\lambda - \phi_k(x)})} \tag{II.5}$$

Therefore, a prediction of a deep RBF is defined such that

$$\hat{y}(x) = \underset{k \in \{1, ..., c+1\}}{\operatorname{argmax}} \ P(y = k|x) \tag{II.6}$$

where the rejection class can be optionally included depending on the task at hand. For example, when presented with OOD data we use the rejection class to detect the anomaly; however, in the data poisoning experiments we ignore the rejection class in favor of using

the confidence assigned to the ground truth (potentially altered) label to detect a poisoned instance.

II.2 Related Research

We now consider related work in defending adversarial attacks, OOD data, and data poisoning attacks. Various defenses have been proposed to combat these threats against DNNs, yet to the best of our knowledge, no single defense mechanism has been able to generalize across the different attack types and be incorporated directly into the DNN architecture without significantly affecting the model's accuracy. We now briefly explore the current state-of-the-art methods of defending against these threats.

Defenses to mitigate adversarial attacks generally rely on input transformations, gradient masking, or adversarial training. Input transformations (preprocessing) methods use the intuition that an adversarial attack perturbs a clean image. Typically, these techniques denoise or capture the salient features of the image (e.g. *denoising autoencoder* or *JPEG compression* [20,21]). The biggest criticism of input transformation techniques is that they introduce an accuracy-robustness trade-off on clean images. Gradient masking techniques make the gradient non-differentiable or force it to zero to prevent adverse responses to small changes in the input. For example, *defense distillation* [22] trains a robust model on the predictions of a trained model, rather than the actual labels, using the class probabilities to enhance its generalization. However, the adaptive Carlini & Wagner attack [17], which iteratively uses various parameterizations to generate adversarial inputs, was shown to bypass this defense [23]. Finally, adversarial training methods augment the training data set with adversarial instances, effectively learning gradient masking [10,14]. Although simple, this method is unable to scale to new attacks.

Other adversarial defense mechanisms have focused on detection instead of mitigation. Examples include training an adversarial detector [24], analyzing the activation artifacts for outliers [25], or comparing the prediction of the adversarial image to a fabricated input

that captures the salient features of the clean image (e.g. *feature squeezing*) [26]. Besides feature squeezing, these other detection technique have been shown to also fail the Carlini & Wagner attack when the attack is properly tuned [27]. Deep RBFs are most similar to detection methods in that they can decrease their confidence to adversarial images [10]. Previous work has been done on training a deep RBF on the trivial MNIST data set [19] to shows that the network can reliably reject adversarial images [28]. We extend this work to analyze a deep RBF in a non-trivial task which we find necessary to show that the robustness scales with the model's size and can be used in CPS.

Defense mechanisms for OOD data have largely focused on detecting, rejecting, or mitigating these anomalies. For example *DeepXplore* is a white-box framework that first detects potential corner cases in the model by analyzing neuron coverage of the training distribution and then fabricates patches for the problem [29]. This method makes the strong assumption, however, that high neuron coverage eliminates corner cases. Furthermore, this method is offline. Other approaches involve using a diversified ensemble method in which the output of a diverse set of models are averaged to increase the reliability of the prediction [30]; however ensemble approaches typically require longer training time and more computational resources. Our work is most similar to outlier detection methods which have previously relied on clustering of the training space [31] or higher level *feature space partitioning* to discover outliers or areas of the network that lack sufficient training [32]. Autoencoder anomaly detectors have also been proposed, relying on the latent space representation of the input to perform anomaly detection [33, 34]. A disadvantage of these detection techniques is that they are not integrated into the model's architecture and must work outside the base model. The RBF rejection class however is computed alongside the class probabilities and can thus more readily be used in CPS. Furthermore, the RBF technique scales in dynamic environments because it is able to learn the rejection class without specifying a particular label during training.

Data poisoning defense methods often focus on detecting the poisoned data and either

mitigating the attack or removing the poisoned sample. Previous work focused on outlier detection; however, such methods rely on a clean data set in order to work effectively [35]. Currently to the best of our knowledge, there is only one proposed defense mechanism that is able to clean a poisoned data set without relying on a certified clean data set. This activation clustering (AC) method relies on k-means clustering of the activations of the DNN's penultimate layer following a dimensionality reduction [36]. However, this method has a large number of hyper-parameters (e.g. the number of clusters, dimensions to reduce, and the choice of the dimensionality reduction and clustering technique) and assumes that a significant portion of the data set has been poisoned in order to have discriminative clusters. In the realistic scenario where we have a sparsely poisoned data set ($<$10%) [13] this assumption fails and the AC method has a large false-positive rate which can reduce the model's baseline accuracy and increase the potential number of corner cases. Our intuition is that a deep RBF trained on a sparsely poisoned data set will lower its confidence of the (poisoned) target label due to the presence of the clean images features in the poisoned image allowing the deep RBF to create a discriminative ordering of the clean and poisoned data. In later sections, we validate this hypothesis by showing that a deep RBF is less susceptible to data poisoning attacks and then compare our RBF outlier detection method directly to the AC method.

CHAPTER III

Out-of-Distribution Black-Box Attack Detection

OOD data can be detrimental to CPS systems. For example, these anomalies can cause self-driving cars to misclassify road signs [37] or even crash from the presence of tire-like marks on a road [7]. However, deep RBFs are able to decrease their confidence in response to anomalies due to their strict class prototype representations. We explore the sensitivity of this response and in particular use the rejection class defined in Eq. II.5 to show that NVIDIA's DAVE-II architecture [2] can be re-formulated as a deep RBF to detect an OOD black-box physical attack similar to that described by [7] in which black lines drawn on the input space (images of a road) induce a crash scenario. We then implement this attack on an end-to-end autonomous RC vehicle known as DeepNNCar [38] to analyze the defense mechanism online.

DeepNNCar is a testbed for autonomous algorithms built on the chassis of the Traxxas Slash 2WD 1/10 RC car and computationally powered by a Raspberry Pi 3 [39]. The sensors on the vehicle include a camera to collect RGB images (320x240x3 @ 30 FPS) and a slot-type IR opto-coupler sensor attached near the rear wheel to measure the RPM and compute speed. DeepNNCar can be wirelessly controlled to collect training data or deployed with a learning algorithm like an end-to-end DNN model.

We use a data set collected by DeepNNCar in the following section to perform the offline attack and then deploy a model on DeepNNCar to perform the online attack. In the offline attack, image processing techniques are used to design the attack whereas in the online attack, a physical black lane is placed across the track to emulate tire marks, debris, or a shadow on the road that may induce a misprediction.

III.1 Black-Box Physical Attack Setup

NVIDIA's DAVE-II model uses normalized 66x200x3 images collected by a frontward facing camera to steer a self-driving vehicle. The DeepNNCar data set ($n = 6000$) which was originally designed for a regression task is discretized into 10 categories based on the ground truth steering label to convert the task into a classification task and then randomly split 70/15/15% into training, testing, and validation. Each discretized class represents a range of 6°, allowing DeepNNCar to turn discretely between -30° (sharp left, $y_i = 0$) and 30° (sharp right, $y_i = 9$) from its forward facing direction.

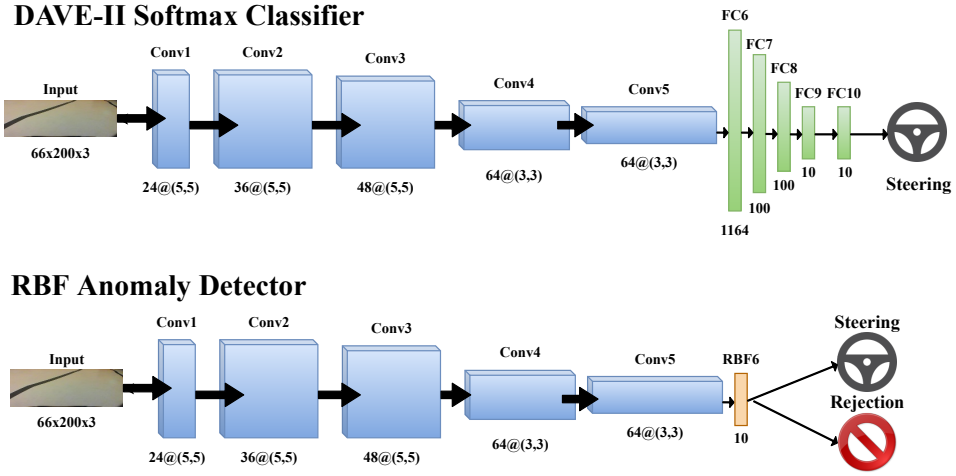
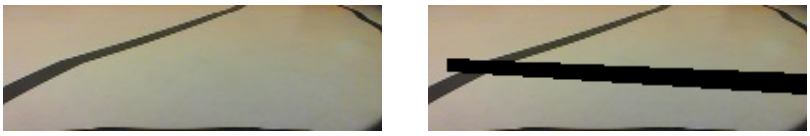


Figure III.1: The modified DAVE-II CNN architectures evaluated in the OOD online and offline attacks.

Figure III.1 summarizes the two architectures evaluated on the physical attack. The regular DAVE-II regression architecture is converted into a classification network (k=10) by adding 10 fully connected neurons to the last layer followed by softmax activation. The deep RBF classifier is designed by adding a hyperbolic tangent activation layer following the convolutional layers of the DAVE-II architecture and replacing the fully connected layers with an RBF layer. The DAVE-II classifier and its RBF counterpart are both trained for 150 epochs using the default parameters of the Adam optimizer [40] and respectively categorical cross-entropy and softML loss.

The black-box physical attack is inspired by [7] and first performed offline using image



(a) Clean Image **(b)** Physical Attack

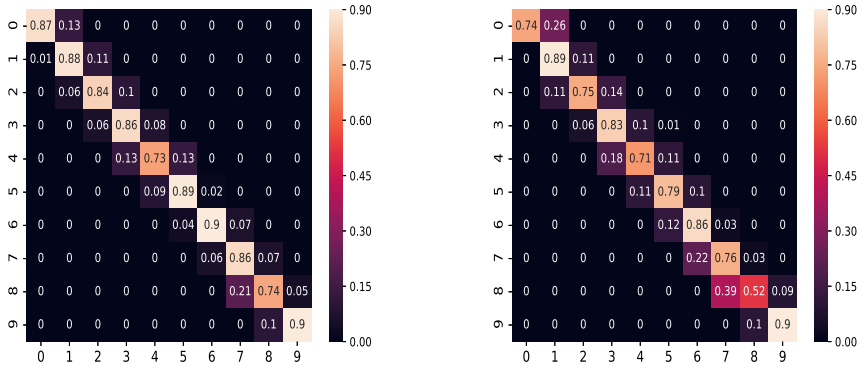
Figure III.2: The regular DAVE-II architecture predicts a steering output in the range 12 to 1 8° or right turn on the clean image (a) and -18 to -24° or left turn on the physical attack image (b). On the other hand, the deep RBF correctly rejects the physical attack image while classifying the clean image between 6 to 12° or slight right turn.

processing to draw a potentially dangerous black "lane-like" marks on 120 clean test images selected randomly but equally from every class $y_i \in \{0,1,2,7,8,9\}$ which represents left and right steering controls. For $y_i \in \{0,1,2\}$ we draw the lane leading diagonally right with some jitter and for $y_i \in \{7,8,9\}$ we draw the lane leading diagonally left with some jitter. Figure III.2 shows an example of the physical attack conducted offline where $y_i = 7$. We denote a successful attack on the regular DAVE-II model and deep RBF model based on the *dangerous criteria* defined below by a boolean statement.

$$dangerous\ criteria = \begin{cases} \hat{y}_i > 1 \pm y_i & f_\theta = REG \\ \hat{y}_i > 1 \pm y_i \wedge \neg reject & f_\theta = RBF \end{cases}$$

where f_θ specifies which trained model is currently being attacked. The dangerous criteria is selected because neither model f_θ violated the bound $\hat{y}_i > 1 \pm y_i$ on the clean test data set *when the rejection class was ignored for the deep RBF* as shown by the confusion matrices in Figure III.3. Furthermore, this criteria allows the deep RBF to reject the point anomalies per Eq. II.5 and II.6. The criteria ensures that the deep RBF's prediction is deemed dangerous if the prediction is off by more than one of the true class and the RBF fails to reject the class. Because of the rejection class, a false-positive can occur for the RBF whenever $\hat{y}_i \leq 1 \pm y_i \wedge + reject$ or when the prediction otherwise would've been considered safe but the RBF model rejects the input.

To provide a baseline comparison, we assess each model on the clean test data ($n = 908$)



(a) Regular DAVE-II Confusion Matrix

(b) RBF DAVE-II Confusion Matrix

Figure III.3: The $i'th$ column in the confusion matrices above represents the ground truth label of the prediction in the $j'th$ row, where true-positives are defined along the diagonal $i = j$. Neither the DAVE-II architecture (a) nor the deep RBF (b) violate the bound $\hat{y}_i > 1 \pm y_i$ on the clean test data set ($n = 908$). This bound is the basis for the dangerous criteria with which we evaluate the OOD data detection mechanism.

using the dangerous criteria. On the clean data, neither model had a dangerous prediction, although the deep RBF had a false-positive rejection rate of 0.25. However, this can likely be reduced by data augmentation, increasing model capacity, or adjusting training hyperparameters to improve the model's accuracy on clean data. Furthermore, one can add an additional threshold γ such that we only accept the rejection class whenever $P(k = c+1|x) > \gamma$; otherwise we only consider the probability scores for classes $\{1,...,c\}$. In the online attack, we use $\gamma = 0.6$ which covers the tail end of the rejection class confidence for both the clean and OOD data (see Figure III.4). We believe this method of using an offline attack to threshold the online attack could be useful in experimentally determining a good threshold to reduce false positives in an online scenario. On the offline attack, we do not use a threshold.

III.2 Offline Out-of-Distribution Detection Results

The regular DAVE-II architecture and the deep RBF are now evaluated offline on the OOD black-box attack. For the regular DAVE-II classifier, the prediction was deemed

dangerous 50% of the time according to the dangerous criteria. Closer analysis of the predictions on the physically manipulated images reveals that the model always predicts a sharp left turn ($\hat{y}_i \in \{0,1\}$) even when the drawn anomaly leads to the right. These results align with previous findings that point anomalies can result in consistent, high confidence misclassifications [11]. On the other hand, the dangerous criteria rate for the deep RBF classifier was 0.0 because the deep RBF was able to successfully reject the anomalous images. Without the rejection criteria, the deep RBF classifier would have predicted an unsafe situation 89% of the time due to the physical attack.

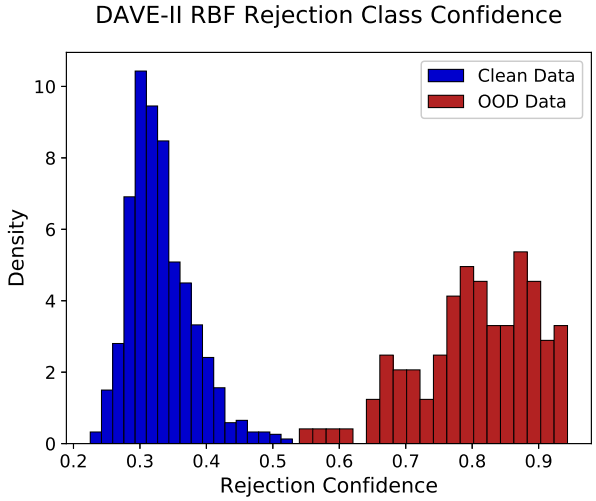


Figure III.4: A significant shift ($p < 0.0001$) was discovered in the distribution of the rejection class for the clean data ($\mu = 0.33, \sigma = 0.049, n = 908$) and the distribution of the offline physical attack data ($\mu = 0.817, \sigma = 0.096, n = 120$) for the deep RBF version of DAVE-II.

In comparison to the clean test data, Figure III.4 shows how the deep RBF exhibited a significant shift (p<0.001) in the confidence of the rejection class for the physical attack data. To further analyze the sensitivity of the deep RBF's rejection confidence, we drew the "fake lane" on the clean image with various opacities controlled by an α value where $\alpha \in [0,1]$. Figure III.5 shows the sensitivity of the rejection class confidence for the six classes that the physical attack targeted. The results show a generally increasing S-shaped curve with inflection points in the range $\alpha \in [0.30.45]$. As the anomaly becomes more present in the clean image the standard deviation of the confidence also decreases. Figure

III.5 also reveals that the deep RBF was more sensitive to anomalies when the ground truth indicated a right turn rather than a left turn. This provides a informal explanation as to why the regular DAVE-II model consistently predicted a sharp left turn when presented with the OOD data. The lack of sensitivity to the left turns is indicative of a poor relative understanding of the feature-space encoding of left turns, likely caused by inaccurate or insufficient data covering those classes. Therefore, it is likely that corner cases could exist that cause the regular model erroneously predict left turns.

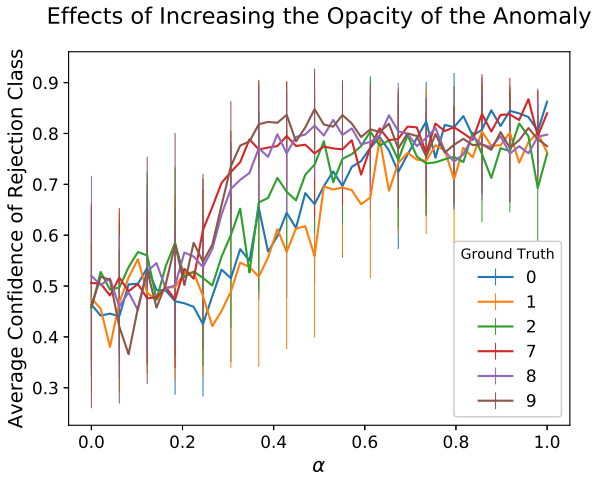
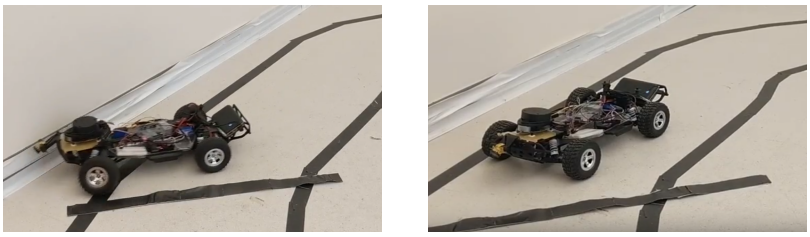


Figure III.5: The deep RBF's rejection class confidence increases as the α-value controlling the opacity of the anomaly increases for the 6 classes targeted by the physical attack while the standard deviation of the confidence decreases.

III.3 Online Out-of-Distribution Detection Results

We now replicate the attack on the DeepNNCar physical test bed and use the rejection class of the deep RBF to detect the anomaly in real-time. The online physical attack is performed by placing an adversarial lane across the track as seen in Figure III.6. We place the adversarial lane at four distinct sections of the track (a left, a straight leading to a left, a right, and a straight leading to a right) at various angles θ off the line that perpendicularly intersects the track such that $\theta \in \{-30, 0, 30\}$. For each trial, we approach the lane at a constant speed and record the number of times a crash occurs for the regular DAVE-II model compared to the deep RBF model. For the deep RBF, we introduce a rejection acceptance

threshold of $\gamma = 0.6$ to reduce false positives, noting that a higher threshold will increase the number of false negative. DeepNNCar is instructed to stop when using the deep RBF when $P(k = c+1|x) > \gamma$. Sample videos of a left and right turn trial run for both models can be found at https://drive.google.com/open?id=10Ek4SH2mBVL-M8pUb7pH-dT_qGDcblDs.



(a) DAVE-II Deployed **(b)** Deep RBF Deployed

Figure III.6: The physical attack caused DeepNNCar to crash when being controlled by the regular DAVE-II model; however the deep RBF was able to catch the physical anomaly and safely stop.

The results of the offline physical attack can be found in Table III.1. Because the DAVE-II model is an end-to-end black box DNN, we consider four scenarios to evaluate the results of the 12 trials. A *crash* is when DeepNNCar completely leaves the lane; a *partial crash* is when DeepNNCar partially leaves the lane but recovers; a *successful navigation* is when DeepNNCar does not leave the lane; and in the case of the deep RBF, a *safe stop* is when the rejection class triggers DeepNNCar to safely stop before the physical attack.

In the online attack, we see the regular DAVE-II model consistently follows the track and crashes several times; however, the deep RBF model is able to often safely reject the physical attack and execute the stop action. In the scenarios that the deep RBF did not reject the physical attack, we see that it never encountered a crash scenario. Furthermore, the inference time of the deep RBF is lower than the regular model. This is because the deep RBF has fewer operations to perform following the convolutional blocks; however it shows that the transformations required to compute the class probabilities and rejection class can be computed simultaneously in real-time.

In summary, the rejection class allowed for real-time attack detection and response that prevented DeepNNCar from crashing. In the online experiments, the response to a rejection

Table III.1: The results of the online black-box physical attack performed when both the RBF and regular DAVE-II model are deployed on DeepNNCar.

	RBF DAVE-II	**REG DAVE-II**
Crash	0	5
Partial Crash	2	3
Successful Navigation	3	4
Safe Stop	NA	7
Inference Time (ms)	44 ± 11	63 ± 17

was to simply stop DeepNNCar. In practice, a CPS system can use the rejection class to alert a human operator to take control or fix the anomaly post-hoc by partially re-training the model offline with the anomaly. Additionally, after detecting the anomaly the end-to-end system possible could hand the controls to a safety guaranteed simplex architecture [38] which could allow for a more complex, yet still autonomous safety response.

CHAPTER IV

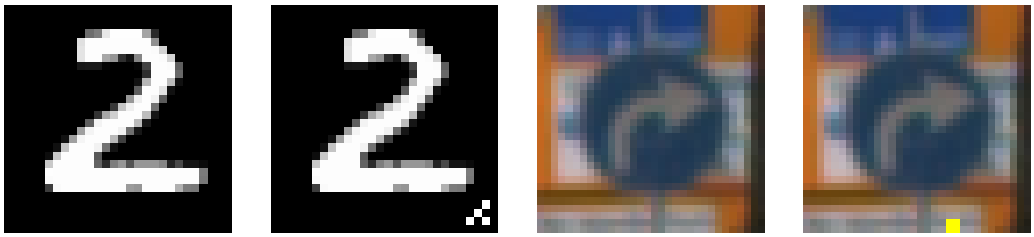
Data Poisoning Detection and Mitigation

We now consider the data poisoning security threat. Data poisoning attacks modify the training procedure to allow the attacker to exploit the adversarial DNN. It is typically assumed the attacker has limited influence on the training procedure and access to a small, fixed portion of the data set. We adapt the popular *injected pattern-key* attack where the labels of the training data set are altered whenever a backdoor key is encoded into the training input, allowing the attacker to exploit the attack by encoding the backdoor key into a test instance [12] [13]. This section first describes experimental evidence that RBFs are less susceptible to data poisoning attacks than regular classifiers and then provides a theoretical discussion as to why this is the case. The theoretical discussion is motivation for our novel RBF outlier detection method to clean a sparsely poisoned training data set without relying on a verified clean data set. The first poisoning attack we evaluate is a toy rotating-label attack performed on the MNIST hand-written digit recognition task (k=10) [28]. The second poisoning attack is a targeted label attack performed on the German Traffic Sign Benchmark (GTSB) data set (k=43) that attempts to cause a DNN to predict a road sign as 80 km/hr whenever a backdoor key similar to a post-it note is encoded in the input [41].

IV.1 Data Poisoning Attack Setup

To poison the MNIST data set, we (1) uniformly at random select n_p instances from X_{train}, (2) add the pattern-key shown in Figure IV.1b and (3) rotate the label such that $y_i = (y_i + 1)\%10$. To poison the GTSB data set we (1) uniformly at random select n_p instances outside of the 80 km/h class, (2) add a yellow, post-it like note at a random location in the image shown in Figure IV.1d, and (3) change the instance label to that of the 80 km/h road sign. A poisoning attack is successful if a model predicts non-poisoned

images as their ground truth and poisoned images as the modified label.



(a) MNIST Clean **(b)** MNIST Poison **(c)** GTSB Clean **(d)** GTSB Poison

Figure IV.1: Examples of poisoned backdoor instances.

For the MNIST data poisoning attack, we train three CNN models f_{m1}, f_{m2}, and f_{m3}. Table IV.1 summarizes the model's architectures. The major differences between the three models is that after the convolutional blocks f_{m1} performs classification using two fully-connected layers followed by softmax activation; f_{m2} uses one fully-connected layer followed by one RBF layer; and f_{m3} uses one RBF layer. This is to explore the effects of the fully-connected layer in a rotating-label attack. For each poisoning attack, the models are trained on the poisoned data set (n=60000) for 10 epochs and evaluated on the test data set which always includes 9000 clean instances and 1000 random backdoor instances.

Table IV.1: A summary of the DNN architectures used in the MNIST and GTSB data poisoning attacks.

	Model	**Architecture**
MNIST Poisoning Attack	f_{m1}	2 conv-pool-ReLU blocks, 2 fully-connected layers, softmax activation
	f_{m2}	2 conv-pool-ReLU blocks, 1 fully-connected layer, 1 RBF layer
	f_{m3}	2 conv-pool-ReLU blocks, 1 fully-connected layer, 1 RBF layer
GTSB Poisoning Attack	f_{g1}	ResNet20: 20 CNN layers with residual connections, 1 fully-connected layer, softmax activation
	f_{g2}	Modified ResNet20: 20 CNN layers with residual connections, 1 RBF layer

For the GTSB data poisoning attack, we train two models. The first model f_{g1} uses the ResNet20 [42] to classify the 43 German road signs. The second model f_{g2} is a deep RBF that uses the same ResNet20 architecture but replaces the final fully-connected layer with an RBF layer preceded by hyperbolic tangent function. Both models are trained on the poisoned data set (n=39209) for 10 epochs and evaluated on 11430 clean test images and 1200 backdoor instances. The RBF networks are trained using the SoftML loss and the regular classifiers are trained using categorical cross-entropy. All models are trained using the Adam optimizer with default parameters.

IV.2 Explaining the Data Poisoning Results

We now evaluate the susceptibility of the deep RBFs on the data poisoning attacks by adjusting the number of poisoned samples n_p in the training data set. We then use the findings summarized in Figure IV.2 to explain why deep RBFs are less susceptible to data poisoning attacks than regular DNN classifiers.

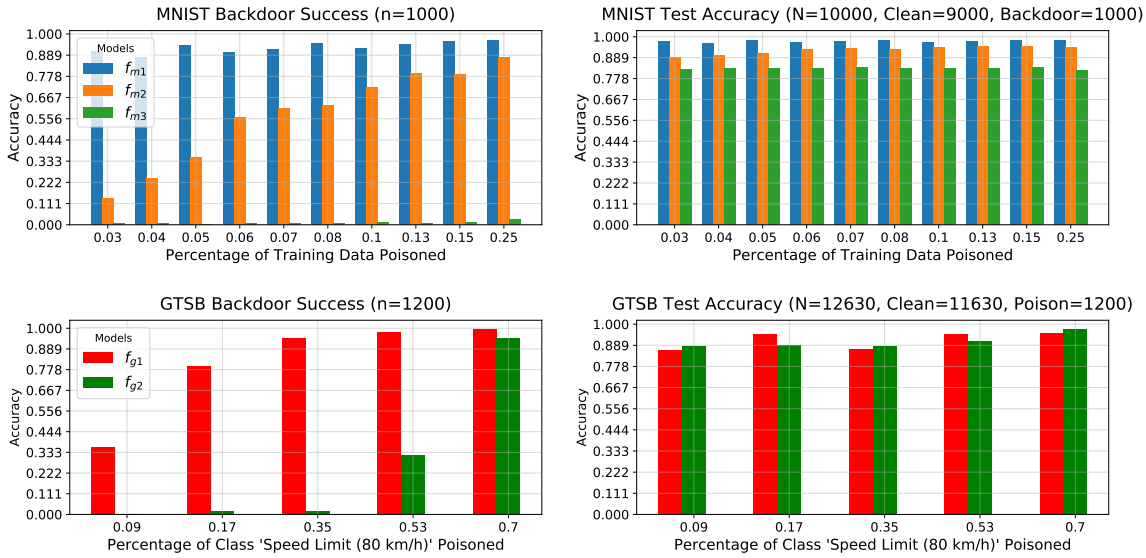


Figure IV.2: The accuracy of f_{m1}, f_{m2}, and f_{m3} on the backdoor MNIST test images (top left) as well as the overall accuracy on the test data (top right). The bottom row graphs show the same data but for f_{g1} and f_{g2} on the GTSB poisoning attack. The RBF models are less susceptible to data poisoning attacks as a result of learning strict representations of the feature-space that aids in generalization.

The MNIST poisoning results indicate that only 3% of the training data set needs to be

poisoned for the attack on f_{m1} to have more than a 90% success rate and 5% to have more than a 95% success rate. This is because the high dimensional dot product performed by the two fully-connected layers can easily overfit the data and encode the logic to rotate the label given a backdoor key. We also see the f_{m2} slowly succumbs to the poisoning attack as n_p increases. Given a large enough n_p, the fully-connected layer of f_{m2} can encode the logic necessary to rotate the label without manipulating the highly non-linear weights controlling the RBF layer which would cause a dramatic increase in the loss. However, f_{m3}, which has no fully-connected layers, never succumbs to the MNIST poisoning attack ($<$1% attack success rate $\forall n_p$). This is because f_{m3} can not encode the poisoning attack logic in its single RBF layer without sacrificing the current representation of each classes prototype and dramatically increasing the loss function. Still, f_{m3} had lower general accuracy on the test data than both f_{m1} and f_{m2}. This is likely due to lack of capacity of the architecture and makes it unclear whether or not this was the reason f_{m3} did not succumb to the MNIST poisoning attack. Also, the backdoor key of the MNIST attack only affects the bottom right corner of the input where there are no features of interest (i.e. the MNIST digits are always centered).

In the GTSB poisoning attack, both f_{g1} and f_{g2} achieve similar overall accuracy on the test data. This eliminates the possible argument that f_{g1} is simply learning the data distribution better than f_{g2} and is therefore more likely to be successfully poisoned. In fact, the poisoning success rate of f_{g1} is greater than 30% after only 5% of the class data has been poisoned despite f_{g2} having better overall accuracy in that trial. We can also see that f_{g2} eventually succumbs to the poisoning attack, but requires that over 30% of the 80 km/hr class data is poisoned to only begin to have a poisoning success rate that nears 30%. It is important, however, to remember that the GTSB attack was performed on a single class rather than globally in the rotating-label attack. Therefore, beyond a certain percentage it is unreasonable to assume that any model given enough capacity would not succumb to the GTSB attack.

In summary the data poisoning results can be explained by comparing the effects of a high dimensional dot product in the case of a regular classifier and the highly non-linear RBF operation that forces the deep RBF to learn strict, feature-based representations of the data. As a result, when n_p is small the regular classifier can still overfit the data and encode the logic necessary to encode the backdoor key without significantly affecting the model's performance on clean data. However, the RBF layer must sacrifice the current learned representation of the targeted class to encode the backdoor logic which increases the error on the clean data for both the target class and the base class. As a result, the deep RBF is less susceptible to data poisoning attacks and requires a larger n_p for the poisoning attack to succeed.

IV.3 RBF Outlier Detection Method

The RBF outlier detection method for sparsely poisoned data sets is based on the previous findings that the RBF layer and the loss function in Eq. II.3 forces the network to learn strict, rigid representations of the feature spaces for each class $k \in \{1, ..., c\}$. When applied directly to the features extracted by the network before entanglement (i.e. before the fully connected layers) the deep RBF will be able to discriminate between clean and poisoned instances by producing an ordering of the training data where $\phi_{y_i}(X^i_{poison}) > \phi_{y_i}(X^i_{clean})$. As a result, we can introduce a threshold β such that we label any training instance X^i with ground truth y_i as poisoned whenever $\phi_{y_i}(X^i_{poison}) > \beta$. The hyper-parameter β is task-specific and depends on the number of data points that are suspected to be poisoned and the distribution of $\phi_{\hat{y}_i}(X^i_{train})$. In practice, we find that for sparsely poisoned data sets, the RBF can robustly separate poisoned data from clean data. We now use the RBF outlier detection method to clean the poisoned MNIST and GTSB data sets using the poisoned models f_{m3} and f_{g2} respectively, remembering that these models have an RBF layer that directly follows the convolutional blocks.

Figure IV.3 shows the receiver operating characteristics (ROC) curve for the RBF out-

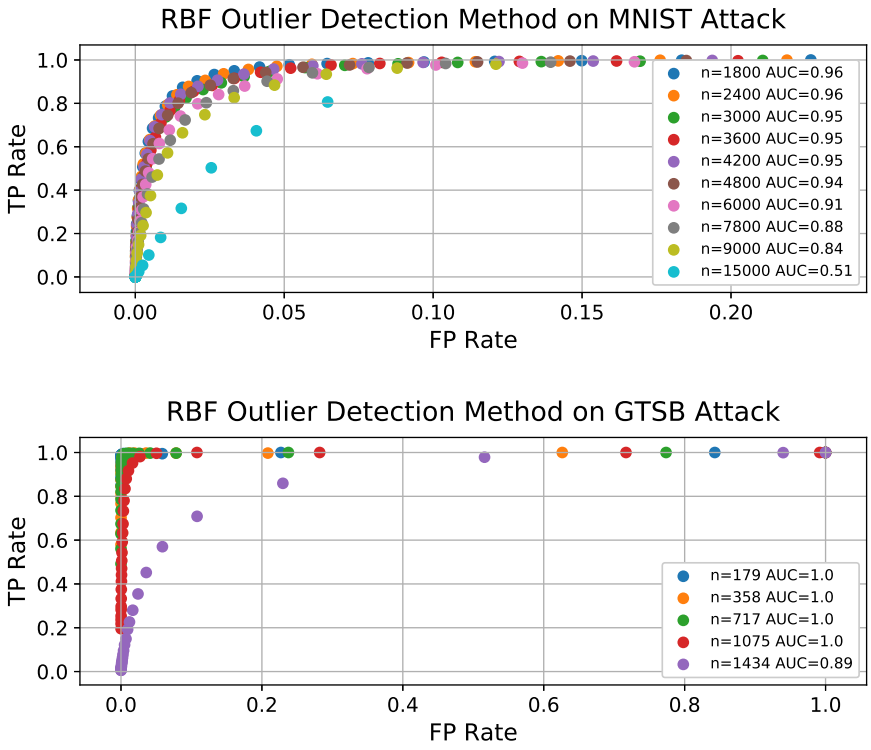


Figure IV.3: The performance of the RBF outlier detection method is effective as long as the number of poisoned samples is insignificant; however, if the attacker has access to a large amount of the training data then the AC method is preferred.

lier detection method applied to the MNIST and GTSB data set. The area under the curve (AUC) score provides an aggregate measure of the effectiveness of a binary classification for various values of β. For the MNIST attack, the AUC is greater than 90% until the data poisoning exceeds 10% of the total training set ($n_p = 10000$). For the GTSB attack, the AUC is 1.0 until 43% of the 80 km/hr speed sign class data set has been poisoned ($n_p = 1434$) at which the AUC only drops to 0.89 despite the poisoning attack success rate of f_{g2} exceeding 90%. This show that the RBF outlier detection method can still succeed despite a high poisoning success rate. Secondly, a comparison of the MNIST and GTSB RBF outlier detection cleaning reveal that as the detector's accuracy increases on the clean data, so does its ability to effectively clean. Therefore, for the best AUC score, it is important to train a model with sufficient capacity and to properly tune the hyperparameters.

IV.4 Comparison of RBF Outlier Detection and AC Method

We now compare our RBF outlier detection method to the activation clustering (AC) method [36] which clusters the penultimate layer's activations to separate poisoned and clean instances. To perform the AC method we use the author's suggestion of K-means (k=2) and PCA to reduce the penultimate layer's activations to 10 dimensions. Figure IV.4 compares the AC method to the RBF outlier detection method using $\beta = 1.72$ to modestly cover the tail end of the distribution of $\phi_{y_i}(X^i_{poison})$. We now compare the two techniques, the AC method and RBF outlier detection method, in cleaning the MNIST and GTSB poisoning tasks in both sparse and non-sparse poisoning conditions.

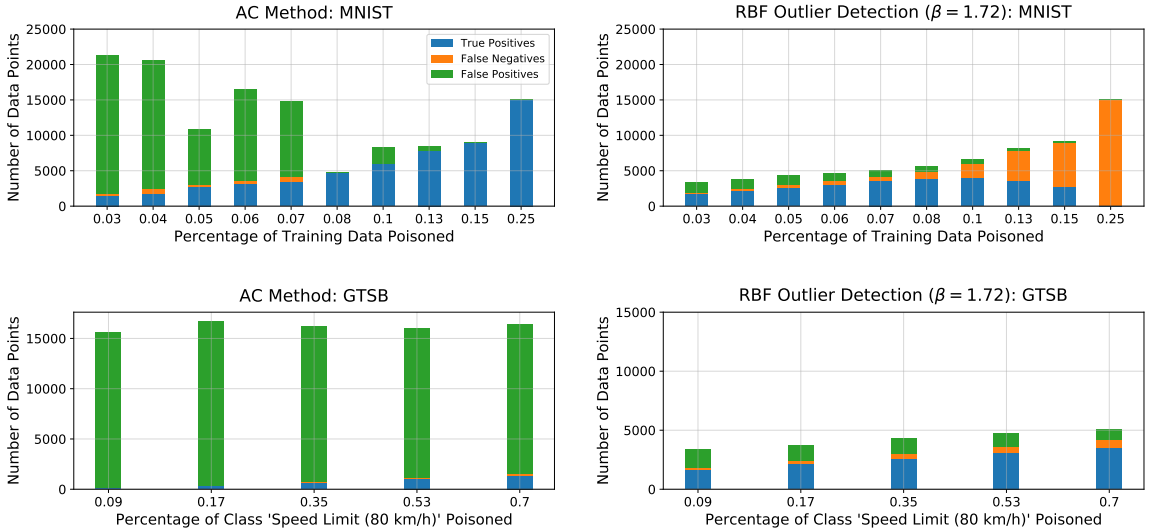


Figure IV.4: The stacked bar charts compare the AC method using K-means ($n_c = 2$) and PCA ($|D| = 10$) with the RBF outlier detection method ($\beta = 1.72$) and show the number of true positives, true negatives, and false positives based on the method labeling each training sample as *poisoned* or *clean*. When there is a significant amount of poisoned data the AC method outperforms our method; however, if the number of poisoning data is sparse, the RBF outlier detection method is preferred due to the high number of false positives that the AC method produces.

Figure IV.4 reveals the results of adjusting n_p and comparing the two cleaning methods for both the MNIST and GTSB poisoning attacks. In the sparse poisoning conditions for the attacks, the RBF outlier detection method was able to achieve on average higher true positive rates and lower false positive rates than the AC method. This is especially true for the GTSB attack in which the AC method was never able to lower its false positive rate,

even when 43% of the data was poisoned ($FP_{rate} = 0.39$). At the lower values of n_p where the poisoning success rate on the regular classifiers still exceeds 90%, the AC method tends to predict fewer true positives and a significant number of false positives exceeding 25000 ($FP_{rate} = 0.39$) and 15000 ($FP_{rate} = 0.34$) for MNIST and GTSB respectively. However, the RBF method begins to fail as the number of poisoned instances increases whereas the performance of the AC method dramatically increases.

In summary, the RBF outlier detection method is more effective in sparse poisoning conditions whereas the AC method is more effective when a significant portion of the training data set has been poisoned. This is expected as the AC method requires a significant number of poisoned instances to consider them as a unique cluster whereas our RBF outlier detection method requires a significant number of clean data to consider the poisoned instances as outliers. Still, the ability to control β in the RBF outlier detection method allows easy tuning of the number of true positives at the sacrifice of false positives whereas it is less obvious how one can better steer the AC method in the sparse poisoning conditions. In practice, selecting an appropriate defense depends on a variety of factors including the number of expected poisoned samples or the effects of accepting false positives. However, with the addition of the RBF outlier detection method, one has the tools necessary to clean sparsely poisoned data sets without relying on a verified clean data set and can also use a deep RBF to reduce the likelihood that a poisoning attack succeeds at all.

CHAPTER V

White-Box Adversarial Attack Robustness

Adversarial attacks manipulate the input space by adding non-random imperceptible noise typically using linear approximations of the gradient of the neural network to induce a high confidence misprediction [10]. In this paper, the adversarial attacks are generated using IBM's Adversarial Robustness Toolbox [43] which provides an API for performing FGSM, I-FGSM, Carlini & Wagner, Deepfool, and PGD [10, 15–18]. We use Carlini & Wagner's guidelines [27] to holistically evaluate the robustness of the deep RBF against these specific white-box adversarial attacks, and (**1**) evaluate the deep RBF on a variety of attacks stronger than FGSM, (**2**) prove its robustness against a white-box, adaptive attack (Carlini & Wagner Attack), (**3**) report the true positive and false positive rates of the atttack success, (**4**) use a data set more complicated than MNIST, and (**5**) release source code at https://github.com/burrussmp/AdversarialDefense. The deep RBF classifier that is evaluated is based on the InceptionV3 architecture [44] which has 42 layers and over 22 million parameters. We evaluate a non-trivial architecture to show that the RBF units still enhance the robustness against white-box adversarial attacks despite using a model with enough capacity for realistic CPS tasks.

V.1 Adversarial Attack Setup

The RBF InceptionV3 classifier is designed by replacing the final fully-connected layer of InceptionV3 with a RBF layer which is preceded by a hyperbolic tangent function. To effectively assess the adversarial attacks, the classification task is simplified to 10 random classes from the ILVRC2012 challenge (1300 images per class) [45]. The data set is split 70/15/15% for the training, validation, and testing data respectively with simple data augmentation that includes zooming, rotation, and random horizontal flips. Each model is trained for 150 epochs using the Adam optimizer with default parameters with a learn-

ing rate scheduler decay after 100 epochs. The regular InceptionV3 classifier and its RBF counterpart achieve an accuracy of 79.95% and 78.11% respectively on the clean test data ($n = 1950$).

Experimentally we find that for FGSM, I-FGSM, and PGD it is useful to adjust the parameters $\varepsilon, \Delta\varepsilon = 0.0001$ which control the added amount of perturbation and the change in perturbation (for the iterative algorithms) respectively to generate more realistic adversarial images. Figure V.1 shows the effects of manipulating ε using the FGSM algorithm to generate adversarial images. The default parameters are used for the Carlini & Wagner and DeepFool attacks. Furthermore, for simplicity we use the non-targeted versions whose goal is to cause a misprediction of no particular label.

V.2 Definition of Robustness

This section briefly defines robustness against an adversarial attack. For example, consider the non-targeted version of the FGSM algorithm below [10].

$$X_i^{adv} = X_i + \varepsilon \cdot sign(\nabla_{X_i} J(X_i, Y_i)) \tag{V.1}$$

In FGSM, the RHS calculates the non-random perturbation to add to the clean image X_i using a linearized approximation of the gradient. To determine the robustness, we would like to know the minimum amount of perturbation $\Delta_{adv}(X_i, F)$ we can add to X_i to cause the model to produce a misprediction. Formally, we can write this as

$$\Delta_{adv}(X_i, F) = \underset{\delta X_i}{\operatorname{argmin}}\{||\delta X_i|| F(X + \delta X_i) \neq F(X_i)\} \tag{V.2}$$

Therefore, the robustness of a model F against adversarial attacks is defined as

$$\rho(F) = \mathbb{E}_D[\Delta_{adv}(X_i, F)] \tag{V.3}$$

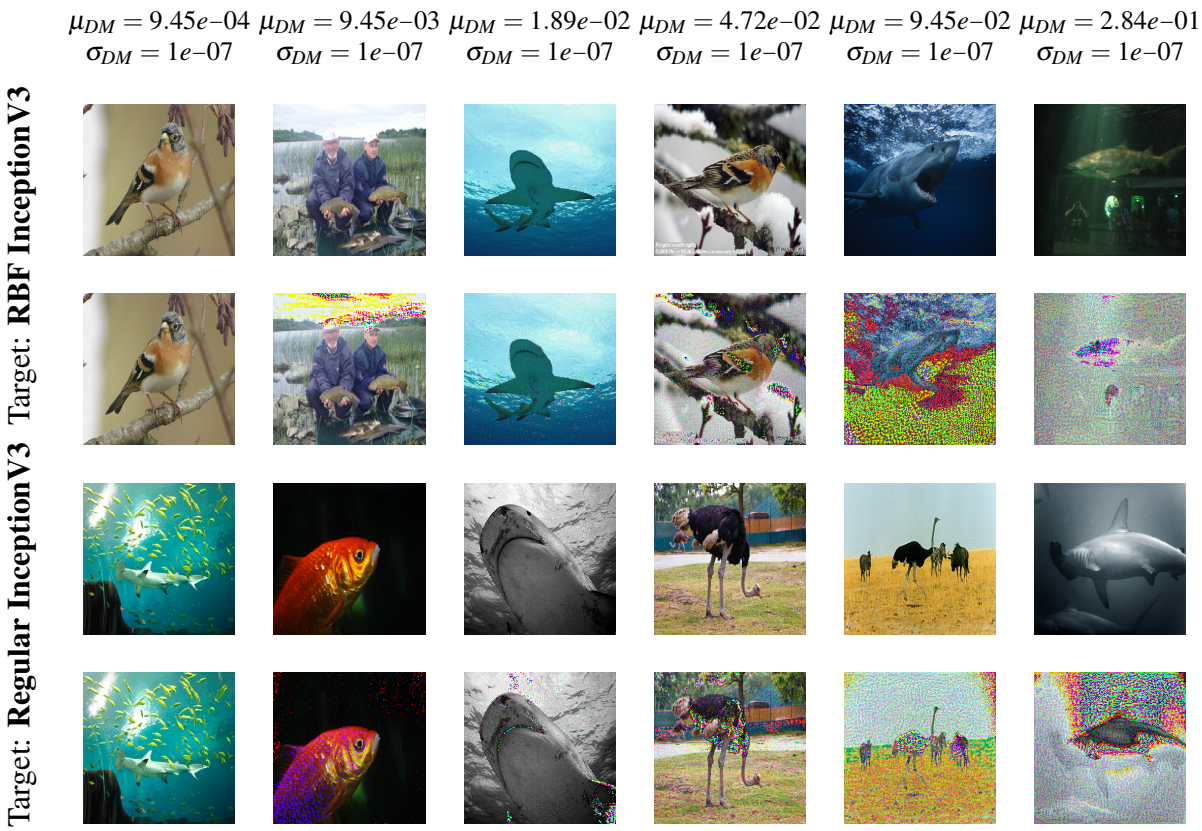


Figure V.1: FGSM was used to target the RBF and regular InceptionV3 models and produce adversarial images. The top row in every pair represents the clean image and the bottom row represents the adversarial image. Each column shows an example image from a batch (n=100) where the mean and standard deviation of the distortion measure *DM* of the batch are recorded at the top of each column.

where D is the distribution of data from which X_i is generated. Using this formal definition of robustness, we can compare different models by analyzing the adversarial attack success rate with respect to the amount of perturbation that is added to X_i.

V.3 Evaluation of Adversarial Attacks

To evaluate each attack, we construct 100 adversarial images X^{adv} from X^{clean} which are drawn uniformly at random from the test data set. To provide a baseline comparison, the model's accuracy on the clean data is first reported followed by the true and false positive attack success rate. A true positive is defined as a misprediction caused by X^{adv}, and a false positive is a misprediction otherwise caused by X^{clean}. For each true positive $X_{TP}^{adv} \in X^{adv}$, the mean, maximum, and minimum of the perturbation added defined by $(||X_{TP}^{clean} - X_{TP}^{adv}||_2)^2$ and the model's average confidence on X_{TP}^{adv} and X_{TP}^{clean} are reported. Finally, the true and false positive transfer attack success rate are reported by assessing the attack success rate of the adversarial images generated from one model on the other model.

To provide a metric on the robustness of the networks we consider the distortion measure DM defined as $\sqrt{\frac{(\sum X_i^{clean} - X_i^{adv})^2}{n}}$ which measures the perturbation evenly distributed among the number of pixels n. We then use FGSM to generate adversarial instances with various levels of distortion and compare the accuracy of the regular InceptionV3 architecture to the deep RBF.

V.4 Results of Adversarial Analysis

Table V.1 compares the robustness of the regular InceptionV3 model and the deep RBF version. On the clean images, both models have similar accuracy. Unlike the original InceptionV3 model, the RBF version is robust to all the attacks, including the adaptive white-box attack (Carlini & Wagner Attack). Furthermore, the deep RBF model generally expresses a decrease in confidence on X_{TP}^{adv} unlike the regular InceptionV3 architecture which typically expresses higher confidence on X_{TP}^{adv} than X_{TP}^{clean}. Secondly, the transfer attack is more successful on the RBF model than on the regular model. However, the

transfer attack success rate on the RBF model is still significantly lower than the direct attack success rate on the regular model. Overall, the significant finding of Table V.1 is that the deep RBF is more difficult to attack than the regular DNN classifier at the given level of perturbation for all of the attacks evaluated.

Table V.1: Each column shows the results of a particular attack where 100 adversarial images were constructed by targeting first the regular model and then the deep RBF. The RBF version of InceptionV3 is more robust than the regular model on all of the attacks tested and also expresses a decrease in confidence to the adversarial images

	FGSM		**I-FGSM**		**Carlini & Wagner**		**DeepFool**		**PGD**	
	Reg	**RBF**	**Reg**	**RBF**	**Reg**	**RBF**	**Reg**	**RBF**	**Reg**	**RBF**
Baseline Accuracy on X^{clean}	77%	76%	77%	76%	83%	79%	77%	76%	83%	80%
True Positive Attack Success Rate	72%	7%	77%	15%	76%	26%	72%	22%	83%	10%
False Positive Attack Success Rate	11%	23%	11%	23%	8%	14%	14%	20%	9%	18%
Average Confidence on X_{TP}^{clean}	0.895	0.590	0.902	0.604	0.893	0.834	0.923	0.763	0.902	0.602
Average Confidence on X_{TP}^{adv}	0.845	0.544	0.998	0.575	0.613	0.440	0.790	0.512	0.998	0.653
Average ℓ_2-norm Perturbation	5.17e-2	5.17e-2	4.42e-2	4.24e-2	4.83e-2	3.62e-2	112	2.27e-2	4.41e-2	4.25e-2
Maximum ℓ_2-norm Perturbation	5.17e-2	5.17e-2	4.82e-2	4.49e-2	1.45	0.176	1402	0.117	4.65e-2	4.37e-2
Minimum ℓ_2-norm Perturbation	5/17e-2	5.17e-2	4.15e-2	4.09e-2	1.65e-05	5.75e-4	8.71e-05	4.62e-4	4.18e-2	4.19e-2
True Positive Transfer Attack Success Rate	7%	20%	7%	17%	3%	6%	2%	34%	3%	12%
False Positive Transfer Attack Success Rate	22%	20%	20%	23%	17%	21%	22%	24%	15%	17%

Figure V.2 allows us to more closely analyze the effects of increasing the distortion applied to an image to look at the robustness of the deep RBF. Random images were selected to create adversarial batches (n=100) with different mean distortion measures using FGSM to target the regular model and the RBF model. We compared the attack success rate for a direct attack and a transfer attack as well as the effect of random uniform noise added with different opacities to a clean image. The results reveals a few key insights. Firstly, using a paired t-test we find that FGSM is significantly more effective at targeting the regular model than the deep RBF model at various distortion measures ($p<0.05$). Secondly, we can find a significant difference in comparing the direct and transfer attacks on both models at various distortion measures ($p<0.05$). Thirdly, the FGSM attack appears to be more effective in targeting the RBF model than injecting random noise; however, there is no significant difference ($p>0.05$). Finally, Figure V.2 shows how the deep RBF model

eventually increases its confidence in rejecting the adversarial images for $\mu_{DM} > 0.0001$.

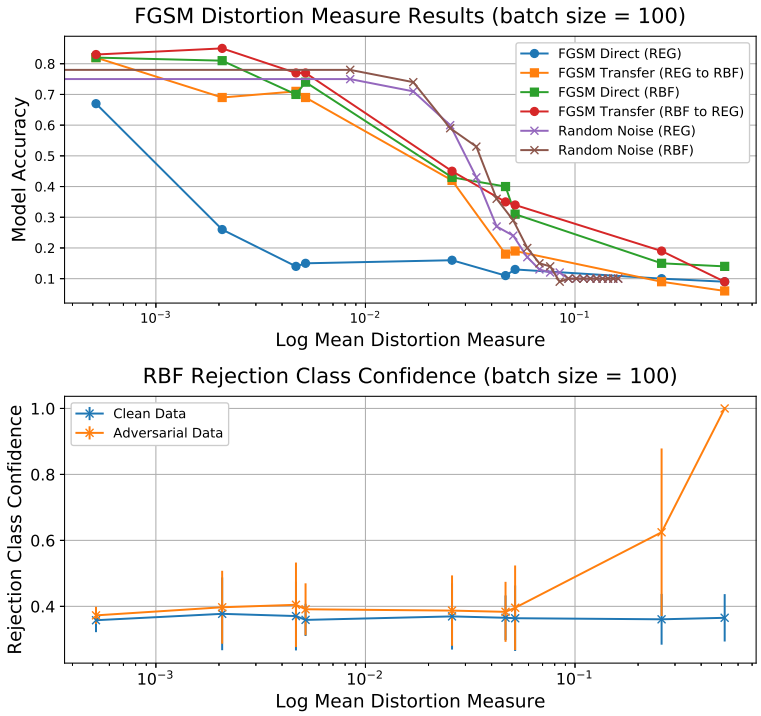


Figure V.2: (Top) The accuracy of the deep RBF and regular InceptionV3 model for both the direct FGSM attack, transfer FGSM attack, and random noise attack as a function of the mean distortion of the adversarial batch (each mark corresponds to a trial with a batch (n=100) selected uniformly at random from the test data set). (Bottom) The rejection class confidence of the RBF model for the clean and adversarial data.

Combining these key insights reveal two important properties of deep RBFs. Firstly, although the RBF layer increases the network's robustness, adversarial algorithms are still more successful than random noise; however, once sufficient distortion has been introduced, the confidence in rejection will increase allowing the adversarial input to be rejected by a deep RBF. Secondly, adversarial algorithms are able to partially succeed by exploiting the linearity of preceding layers which is why we see that an attack on the RBF network is more effective than a random noise attack, as effective as the transfer attack on the regular model, but less effective than the direct attack on the regular model. This is because FGSM is able to partially exploit the linearity of the layers preceding the RBF layer. We would also expect adaptive attacks like Carlini & Wagner to more be more effective at exploiting this vulnerability as was found in Table V.1 in comparing the various attacks on the

deep RBF. Although this is still a preliminary finding, future work can focus on validating this effect by injecting RBF layers into different areas of the network to determine if the robustness of the network increases.

CHAPTER VI

Closing Remarks

VI.1 Discussion and Future Work

Deep RBFs address three major problems with DNNs that restrict their use in CPS systems. Firstly, traditional classifiers tend to respond abnormally to OOD data points whereas the deep RBF networks provide a rejection class to handle such situations. Furthermore, the confidence of this rejection class is sensitive to how obtrusive the anomaly is, allowing a CPS system to respond with various protocols depending on the confidence of the rejection class. These findings are significant in that the OOD detection using RBFs is scalable (i.e. doesn't require specific labeling of anomalies) and real-time (i.e. rejection class predicted alongside class probabilities). We experimentally showed that an RBF-version of NVIDIA's DAVE-II architecture could catch an OOD black-box attack in a self-driving task and prevent a crash in real-time. Future work should address the sensitivity of this rejection class to various types of OOD data (rotation, brightness, occlusion, etc.) and address the problem of intelligently thresholding the rejection class to provide safety guarantees in CPS systems by combining the RBF rejection in a simplex architecture.

Secondly, traditional DNNs are susceptible to data poisoning attacks that allow an attacker to exploit a backdoor key encoded by the network. We provided experimental evidence and a theoretical discussion as to why deep RBFs are less susceptible to data poisoning attacks. Using this understanding of deep RBFs, we devised a method to clean sparsely poisoned data sets using a deep RBF trained on the poisoned data. We found that our RBF outlier detection method outperforms the AC method in the realistic scenario that the data set has been sparsely poisoned, whereas the AC method outperforms our method when a larger portion of the data set has been poisoned. This is because the AC method relies on a large sample of poisoned data to form discriminative clusters to separate poisoning and

clean data whereas the RBF outlier detection method depends on a large sample of clean data to effectively label the poisoned data as outliers. However, the RBF outlier detection method has the additive advantage of being easily tuneable and with fewer hyper parameters. In reality, both tools should be used in the scenario that one has no idea what portion of the data has been compromised. In reality, the cleaning process of a poisoned data set will not use a black-box method. Therefore, future work should focus on using the RBF outlier detection method in a visual analytics tool to help assess potential poisoned samples and evaluate the effects of selecting various hyperparameters.

Finally, we showed that deep RBFs are more robust to white-box adversarial attacks than regular classifiers in that they are more difficult to attack and express a decrease in confidence on adversarial instances when the added perturbation crosses a certain threshold. However, adversarial attacks generated by FGSM were slightly more effective than the random noise injection attack due to the ability of FGSM to capitalize on the linearity of preceding layers. Future work should focus on injecting RBF activations into intermediary layers in order to increase the robustness of the network. However, such work would require accomodating for the unstable gradient caused by the intermediary RBF units. Also, while we have only explored classification tasks, future work can find ways to inject RBF activations in segmentation, detection, and regression networks.

VI.2 Conclusion

Despite the potential of using DNNs in CPS, they are susceptible to many security threats. We analyzed three specific threats including black-box OOD attacks, data poisoning attacks, and white-box adversarial attacks. Until now defense mechanisms have been disjoint from the underlying architecture and have not generalized across the attack types. First, we showed that an integrated out-of-distribution detection mechanism can be implemented on an end-to-end autonomous RC vehicle using the rejection class of the deep RBF to achieve real-time and scalable anomaly detection and classification. Furthermore,

we explained how deep RBFs are fundamentally less susceptible to data poisoning attacks because of their ability to learn strict representations of the feature space of the data that helps the model generalize and not overfit the training data. We then used this finding to present a novel RBF outlier detection method for cleaning a sparsely poisoned data set without relying on a verified, clean data set. Finally, we showed that a production-grade deep RBF classifier can parallel the performance of a traditional DNN yet be more robust against a variety of white-box adversarial attacks. Overall, we showed that the enhanced robustness of deep RBFs against these security threats could help foster more widespread use of DNNs in safety-critical CPS.

BIBLIOGRAPHY

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] "Just how far ahead is tesla in self-driving." https://www.forbes.com/sites/greatspeculations/2019/11/08/just-how-far-ahead-is-tesla-in-self-driving/#3b71f97c1b24.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[5] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–10, IEEE, 2016.

[6] F. Lambert, F. Lambert, and Fred, "Understanding the fatal tesla accident on autopilot and the nhtsa probe," Jul 2016.

[7] A. Boloor, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Simple physical adversarial examples against end-to-end autonomous driving models," *arXiv preprint arXiv:1903.05157*, 2019.

[8] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *CoRR*, vol. abs/1804.00792, 2018.

[9] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, *et al.*, "Adversarial attacks and defences competition," in *The NIPS'17 Competition: Building Intelligent Systems*, pp. 195–231, Springer, 2018.

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

[11] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

[12] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[13] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[14] 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.

[15] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *CoRR*, vol. abs/1511.04599, 2015.

[16] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2016.

[17] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[19] P. H. Zadeh, R. Hosseini, and S. Sra, "Deep-rbf networks revisited: Robust classification with rejection," *arXiv preprint arXiv:1812.03190*, 2018.

[20] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[21] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, "Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression," *arXiv preprint arXiv:1705.02900*, 2017.

[22] 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 Symposium on Security and Privacy (SP)*, pp. 582–597, IEEE, 2016.

[23] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," *arXiv preprint arXiv:1607.04311*, 2016.

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

[25] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[26] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[27] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," *CoRR*, vol. abs/1705.07263, 2017.

[28] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[29] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, ACM, 2017.

[30] A. J. Sharkey and N. E. Sharkey, "Combining diverse neural nets," *The Knowledge Engineering Review*, vol. 12, no. 3, pp. 231–247, 1997.

[31] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, pp. 1–13, 2017.

[32] X. Gu and A. Easwaran, "Towards safe machine learning for cps: infer uncertainty from training data," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 249–258, 2019.

[33] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pp. 4–11, 2014.

[34] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," 2018.

[35] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Advances in neural information processing systems*, pp. 3517–3529, 2017.

[36] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[37] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *CoRR*, vol. abs/1707.08945, 2017.

[38] S. Ramakrishna, A. Dubey, M. P. Burruss, C. Hartsell, N. Mahadevan, S. Nannapaneni, A. Laszka, and G. Karsai, "Augmenting learning components for safety in resource constrained autonomous robots," in *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, pp. 108–117, IEEE, 2019.

[39] M. Burruss, S. Ramakrishna, G. Karsai, and A. Dubey, "Deepnncar: A testbed for deploying and testing middleware frameworks for autonomous robots," pp. 87–88, 05 2019.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *IEEE International Joint Conference on Neural Networks*, pp. 1453–1460, 2011.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[43] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.1.0," *CoRR*, vol. 1807.01069, 2018.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.