Deep-RBF Networks for Anomaly Detection in Automotive Cyber-Physical Systems

Matthew Burruss*§, Shreyas Ramakrishna†, and Abhishek Dubey†
* Microsoft Corporation, † Vanderbilt University

***Abstract*—Deep Neural Networks (DNNs) are popularly used for implementing autonomy related tasks in automotive Cyber-Physical Systems (CPSs). However, these networks have been shown to make erroneous predictions to anomalous inputs, which manifests either due to Out-of-Distribution (OOD) data or adversarial attacks. To detect these anomalies, a separate DNN called assurance monitor is often trained and used in parallel to the controller DNN, increasing the resource burden and latency. We hypothesize that a single network that can perform controller predictions and anomaly detection is necessary to reduce the resource requirements. Deep-Radial Basis Function (RBF) networks provide a rejection class alongside the class predictions, which can be utilized for detecting anomalies at runtime. However, the use of RBF activation functions limits the applicability of these networks to only classification tasks. In this paper, we show how the deep-RBF network can be used for detecting anomalies in CPS regression tasks such as continuous steering predictions. Further, we design deep-RBF networks using popular DNNs such as NVIDIA DAVE-II, and ResNet20, and then use the resulting rejection class for detecting adversarial attacks such as a physical attack and data poison attack. Finally, we evaluate these attacks and the trained deep-RBF networks using a hardware CPS testbed called DeepNNCar and a real-world German Traffic Sign Benchmark (GTSB) dataset. Our results show that the deep-RBF networks can robustly detect these attacks in a short time without additional resource requirements.**

***Index Terms*—Cyber-Physical Systems, Deep Neural Networks, Radial Basis Functions, Adversarial Attacks**

I. Introduction

Emerging Trend: Deep Neural Networks (DNNs) are popularly used for implementing autonomy related tasks in automotive Cyber-Physical Systems (CPS). One way to use these networks in autonomous driving is in an end-to-end (e2e) fashion, where the network takes in sensory inputs (e.g. camera images) to predict control actions (e.g. steer) (see Fig. 1). A well-known example of an e2e network is NVIDIA's DAVE-II Convolutional Neural Network (CNN) [1] that was used to steer a car autonomously around the city of New Jersey. Despite being widely used, these networks are shown to be susceptible to anomalies that manifest as Out-Of-Distribution (OOD) data or adversarial attacks. To detect these anomalies, a separate DNN called Assurance Monitor (AM) is often trained and used in parallel to the controller DNN (see Fig. 1). These monitors identify if an operational test input to the DNN belongs to the training distribution or not.

State-of-the-art and Challenges: Recently, generative models such as Generative Adversarial Networks (GANs), and variants of the Autoencoder family (e.g. Variational Autoencoder (VAE)) have been popularly used as monitors to detect DNN based anomalies. Although these monitors have shown robust performance for anomaly detection in CPSs [2], [3], they introduce additional resource and time overhead as found out in our previous work [4]. We hypothesize that a single DNN to perform both controller predictions and anomaly detection is required for CPSs which usually have limited resources and short inference times. In this direction, Amini, Alexander, et al. [5] has used a single VAE to perform continuous steering predictions and anomaly detection. However, generative models are dependent on several hyperparameters (e.g. latent space size), and training them is challenging (e.g. mode collapse problem of GANs).

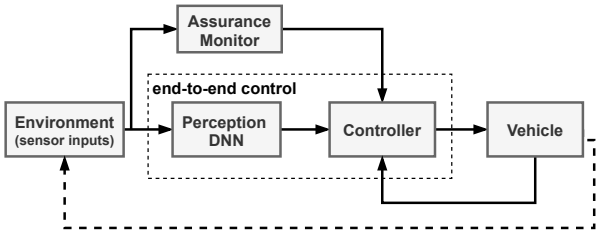


Fig. 1: end-to-end automotive CPS architecture with an assurance monitor. The monitor receives the same sensory input as the controller DNN, and its detection results are often used in control decision making.

Deep-Radial Basis Function (RBF) networks [6] provide a rejection class alongside the class predictions, which can be utilized for anomaly detection. These networks are conventional DNNs with an output RBF layer and do not have additional hyperparameters to tune. Recently, the rejection capability has been used to detect adversaries in toy classification datasets such as MNIST [7], and CIFAR10 [8]. However, to the best of our knowledge, it has yet to be shown whether these networks are capable of detecting anomalies in the CPS domain. Additionally, these networks are mostly designed for classification tasks, limiting their utility in regression tasks.

Our Contributions: In this work we design a single deep-RBF network for predicting control actions (e.g. steering) and detecting anomalies (especially adversarial attacks) in CPS regression tasks. We hypothesize that the non-linearity introduced by the RBF layer decreases the network's susceptibility to anomalies (especially gradient attacks) while increasing its confidence in recognizing in-distribution data and consequently rejecting OOD data. However, as these networks are limited to classification tasks, we discuss the steps required for transforming a CPS regression task (continuous steering) to a

§Work performed during Master Thesis at Vanderbilt University.

classification task (discreet steering classes). We then integrate RBFs to the output layer of well-known DNNs such as NVIDIA's DAVE-II and ResNet20 and train the resulting deep-RBF network. We then use the resulting rejection class of the trained network to detect several adversarial attacks including physical attacks and data poison attacks. We craft a physical attack on a hardware CPS testbed called DeepNNCar [9] which performs e2e steering within a track. Further, we craft a data poison attack on the real-world traffic sign dataset called German Traffic Sign Benchmark (GTSB). Our results show that the deep-RBF network can effectively detect physical attacks with an F1-score of 93.53% within a short time of 44 milliseconds. Further, the network shows higher robustness to poisoned data and requires a significantly larger (35%) part of the GTSB dataset that needs poisoning for misclassifications.

Outline: In Section II we provide a brief overview of the existing methods in detecting DNN related anomalies. In Section III we introduce the deep-RBF networks and the adversarial examples of interest to this work. In Section IV we describe a workflow for anomaly detection using deep-RBF networks. In Section V we evaluate the trained network for detecting adversarial attacks on two CPS applications. Finally, in Section VI we present our conclusions and the possible future extensions.

II. Related Work

DNNs have recently been used as Assurance Monitors for detecting CPS related anomalies that manifest either due to OOD data or adversarial attacks. Generative models such as Generative Adversarial Networks (GANs), and variants of Autoencoders (e.g. VAE) have been recently used to detect DNN related anomalies. In our previous work [3], [10] we used the disentangled latent space generated by a β-VAE to detect automotive CPS related OOD conditions such as high brightness, and high precipitation. For the same domain, the authors in [2] have used the reconstruction capability of a VAE to detect OOD data (high precipitation), and physical attacks. These authors have also used a VAE based regression model for detecting an FGSM attack [11]. In another work [12], a Gaussian Mixture VAE has been used to defend against several black-box and white-box attacks. Despite robust detection capabilities, these models require an independent DNN to be trained in parallel to the controller DNN, which adds resource burden and latency as observed in our previous work [4].

To address this, the authors in [5] have used a single VAE network for both controller predictions and anomaly detection. Though this is significant for reducing the resource burden, the encoder-decoder architecture of the VAE is still resource expensive. Additionally, the detection efficiency of the VAE depends on hyperparameters (e.g. size of the latent space) which has no standard value. Deep-RBF networks are another variant of AMs that have shown robustness in detecting anomalies, especially adversarial attacks [6]. Zadeh. et al. [7] has shown a single deep-RBF network is capable of predicting both class predictions and anomalies without the need for tuning complex hyperparameters. These networks have a DNN structure with RBF attached to the output layer. The classification capability of the RBF along with a threshold is used as a rejection class to detect adversarial attacks on datasets such as MNIST [7], and CIFAR10 [8]. However, RBFs limit the applicability of these networks only to classification tasks on trivial datasets such as MNIST. In this work, we transform a CPS regression task to classification and evaluate the deep-RBF network's anomaly detection capability for runtime CPS applications.

Although the above discussed DNN variants have shown robustness against several anomalies, their defense capability is not explored for data poisoning attacks. Existing methods often focus on detecting the poisoned data or removing them. Game theory models [13], or ensemble of classifiers [14] have been popular approaches for detecting the poisoned data. While these approaches have shown robustness against data poisoning, the game theory approach is complicated and difficult to integrate with the existing system. Additionally, the ensemble of classifiers is resource inefficient as a large number of classifiers are often required. Further, data sanitization [15] has been used to remove the poisoned data from the training set. This approach has worked well but it relies on the availability of a certified clean training set which may not always be available. To the best of our knowledge, the activation clustering (AC) [16] is the only method that can clean a poisoned dataset without relying on a certified clean training set. However, this method has many hyperparameters (e.g. the number of clusters, dimensions to reduce, etc.) and relies on the assumption that a significant portion of the dataset has been poisoned to have discriminative clusters. Such assumptions fail in realistic sparsely poisoned datasets ($<$10%) [17]. We hypothesize that a deep-RBF network trained on a sparsely poisoned dataset can be used for the discriminative ordering of clean and poisoned data without the need for a certified clean training set.

III. Background

In this section we briefly introduce the deep-RBF networks and the adversarial anomalies that are used in this work.

A. Deep-RBF network

Deep-RBF network is a conventional DNN with an output layer of RBF activation functions. 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 $l \leq n$ [7].

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

In the context of a DNN, 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:

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

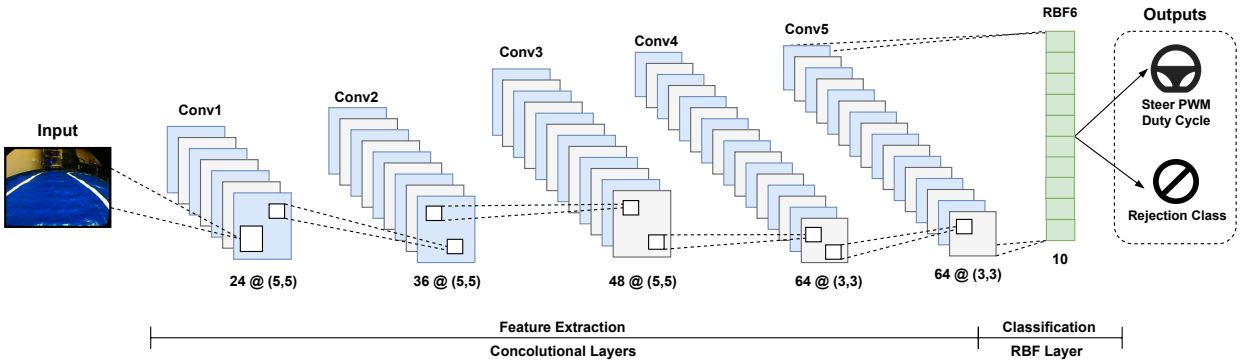


Fig. 2: Deep-RBF network for e2e control of the DeepNNCar example. The image features are extracted through the convolutional layers which are sent to an RBF layer to perform classification. The outputs are the class predictions (discreet steering values), and a rejection class to detect anomalies.

where $W_k \in \mathbb{R}^{|f(x)|}$ is a trainable weight vector intuitively representing the learned prototype of class k. The prediction category that results in the smallest distance is selected as the correct class during the evaluation phase. In practice we have found applying hyperbolic tangent function to the features $f(x)$ preceding the RBF layer and randomly initializing $W \in [-1, 1]$ achieves sufficient model performance.

*1) **Training Loss Function**:* The deep-RBF network can be trained using a metrics-learning inspired loss function named *SoftML*, which is shown in Eq. (3). The function was proposed in [7] and is shown to avoid the vanishing gradient problem.

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

where y_i is the correct class of input $x^{(i)}$, and $\lambda > 0$. The first term in the cost function aims at decreasing the distance between the prediction and the correct class, while the second term aims at increasing the distance of the negative class. Further, as discussed in [7], the value of λ has little effect on convergence and can be arbitrarily chosen.

*2) **Interpreting deep-RBF network output**:* From a probabilistic point of view, the Eq. (3) can be interpreted as the negative log-likelihood as discussed in [7]. Therefore, the class prediction output of the deep-RBF networks can be interpreted as non-normalized probabilities following the transformation.

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

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

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

Therefore, the output of a deep-RBF can be defined as:

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

B. Problems

The Assurance Monitor must be able to detect adversarial attacks, especially physical attacks in the environment and data poisoning attacks. In general these attacks can be summarized as deliberate perturbations added to the input of a DNN which results in an erroneous output prediction. Algorithms to craft adversarial examples differ in their goal and knowledge of the DNN under consideration [18]. A *white-box* attack has access to the DNN architecture and may be able to adjust model parameters whereas a *black-box* attack has no knowledge of the DNN, but they exploit any known corner cases [6].

Physical attacks are adversaries that are physically realizable in the real world and they include perturbing physical objects (e.g. traffic signs) that are captured as images that are fed to DNNs. In this work, we adapt the physical adversary introduced in [19], where physical black lines are added at specific positions and angles to confuse an e2e model to predict erroneous steering angle (see Fig. 4).

Data poisoning attacks modify the training procedure to allow the attacker to exploit the poisoned model. In this setting, an attacker can modify the training procedure, alters the network's logic, or manipulate labels in the training set to encode a backdoor key. In this work we adopt the *injected pattern-key* attack where the labels of the training 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 [20] (see Fig. 7b).

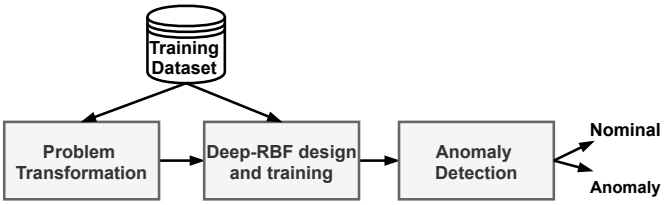


Fig. 3: Workflow for detecting anomalies using deep-RBF networks. (1) problem transformation to transform a regression task to classification, (2) deep-RBF design and training, and (3) anomaly detection.

IV. Anomaly Detection Workflow

In this section we discuss the workflow (see Fig. 3) for anomaly detection using deep-RBF networks. The steps involved are: (1) problem transformation to transform a regression task to classification (this step can be skipped for classification tasks), (2) deep-RBF design and training, and (3) anomaly detection.

A. Problem Transformation

Certain CPS tasks (e.g. control predictions) are regression-based, and using the deep-RBF network for anomaly detection requires transforming the regression task to classification. For explanation, we consider an example of a perception DNN that predicts continuous steering predictions. The DNN observes a sequence of images $\mathbb{X}_k = x_k \cdots x_{k-t}$ from the environment and predicts a continuous steering angle s in range $[+\theta, -\theta]$. Here, $+\theta$ corresponds to a full right turn, and $-\theta$ corresponds to a complete left turn. This continuous steering angle needs to be discretized into n different classes $k = \{0, 1, ..., n\}$. That is, each class corresponds to a small steering angle range (s_k) is calculated as:

$$s_k = \frac{|+\theta| + |-\theta|}{n} \tag{7}$$

Where class k = 1 corresponds to a full right turn, and class k = n corresponds to a full left turn. The intermediate classes result in a certain right or left turn within $[+\theta, -\theta]$. The number of classes (n) for the discretization is problem-specific, but it can impact the sensitivity of the anomaly detection and the control predictions, so should be carefully chosen. A large number of classes will result in highly sensitive detection resulting in high false negatives but will provide fine-grained control over control predictions. A small number of classes will make the detection insensitive resulting in high false positives, but will not provide fine-grained control over the control predictions. So, the number of classes must be appropriately chosen to balance the robustness of control predictions and anomaly detection.

B. Deep-RBF design and training

As discussed earlier, a deep-RBF network is a conventional DNN with an RBF layer attached to the output. The deep-RBF network for the DeepNNCar regression task (continuous steering prediction) is shown in Fig. 2. In this network, the image features are extracted through the convolutional layers which are sent to an RBF layer to perform classification. The number of RBF units in the output layer corresponds to the number of classes (n) that was derived in the previous section (For a classification example, the number of RBF units directly correspond to the number of classes in the task). The output of the deep-RBF network has two components: (1) class predictions - for our running example is the discreet steering angle (or a steering class), and (2) rejection class probability - can be utilized for anomaly detection.

The designed deep-RBF network can then be trained using the *SoftML* loss function discussed in Eq. (3). The training does not involve additional hyperparameters other than the standard ones such as the number of epochs, batch size, learning rate, and the optimizer type. However, the only problem in training the network is that the RBF units introduce high non-linearity making it difficult to train the network, so we apply the RBF layer directly after the convolutional layers rather than after a series of fully-connected layers.

Algorithm 1 Anomaly detection using deep-RBF network

Input: Image x_t at time t.
Output: Binary output $Anom_t$.
Require: Trained deep-RBF network A_{RBF}, Rejection threshold γ.

```
P(ŷ_t = k|x_t), P(ŷ_t = c + 1|x_t) = A_RBF(x_t)
if P(k = c + 1|x) ≥ γ then
    Anom_t = 1
else
    Anom_t = 0
end if
return Anom_t
```

C. Anomaly Detection

During evaluation the test inputs are passed through the trained deep-RBF network, and the output class predictions (discreet steering) can be used to control the CPS, and the rejection class probability (Eq. (5)) can be used along with a pre-selected threshold γ to perform anomaly detection as shown in Algorithm 1. That is, if $P(\hat{y_t} = c + 1|x_t) \geq \gamma$, the input can be isolated to be an anomaly.

At runtime, the output of the anomaly detector can be used in contingency planning and high-level decision-making tasks to improve the safety of CPS.

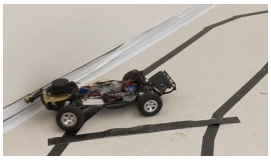
V. Evaluation

We evaluate [1] the performance of the deep-RBF networks for detecting (a) black box physical attack on a hardware testbed called DeepNNCar which performs steering predictions (regression), and (b) data poisoning attack on the real world GTSB classification dataset.

A. Detecting Black Box Physical Attack

*1) **Experimental Setup**:* In our first example, we implement the black-box physical attack introduced in [21] on a hardware platform called DeepNNCar [9]. This testbed is built on the chassis of a Traxxas Slash 2WD 1/10 RC car and computationally powered by a Raspberry Pi 3. The sensors on the car include a forward-looking camera that is configured to collect RGB images of size (320x240) @ 30 FPS, and a slot-type IR opto-coupler attached near the rear wheel to measure the RPM and compute speed. The primary controller is NVIDIA's DAVE-II CNN which uses forward-looking camera images to steer the car autonomously. The car is first manually driven to collect training data which includes 6000 images, steering, and speed PWM values. This set is randomly split into training, testing, and validation in a ratio of 70/15/15%. We then follow the steps in Section IV-A, to transform this regression task into classification by discretizing the continuous steering labels into 10 categories. Each discretized class represents a range of 6°, allowing the car to turn discretely between -30° (sharp left, $y_i = 0$) and 30° (sharp right, $y_i = 9$).

[1] Jupyter notebooks to replicate the experiments can be found at https://github.com/Shreyasramakrishna90/RBF-Adversarial-Detection.git



(a) DAVE-II DNN

(b) RBF DAVE-II DNN

Fig. 4: The physical attack caused DeepNNCar to crash when being controlled by the DAVE-II network; however its RBF extension was able to detect the anomaly and safely stop the car. Videos of these runs can be found on our github (https://github.com/Shreyasramakrishna90/RBF-Adversarial-Detection.git).

The physical attack is performed by placing a black lane across the track as shown in Fig. 4. The lane is placed 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.

*2) **Competing Baselines**:* Two baseline networks are compared for the proposed detection method. First is the NVIDIA's DAVE-II regression network which is converted into a classification network (k=10) by adding 10 fully connected neurons to the last layer followed by softmax activation. The other is the RBF DAVE-II network which 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.

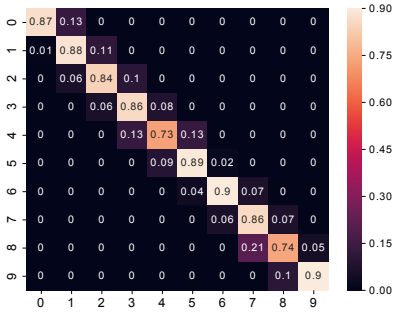
The networks are trained on 4200 training images for 150 epochs using adam optimizer, with categorical cross-entropy loss for the DAVE-II network and SoftML loss for RBF DAVE-II network. For the RBF DAVE-II network, we introduce a rejection threshold of $\gamma = 0.6$, which is empirically selected to reduce false positives.

*3) **Results**:* **(a) Offline Experiments:** We performed an offline evaluation on 1028 images that were collected from two trial runs around the track. This set had 908 clean images (images with no physical attack line), and 120 images with physical attack lines (OOD data). We denote a successful attack on the DAVE-II network and RBF DAVE-II network based on the *failure criteria* defined below.

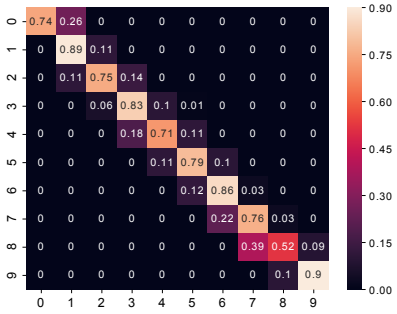
$$criteria = \begin{cases} |\hat{y_i} - y_i| > 1 & f_\theta = DAVE - II \\ |\hat{y_i} - y_i| > 1 \wedge \neg reject & f_\theta = RBFDAVE - II \end{cases}$$

where f_θ specifies which trained model is currently being attacked. The failure criteria is selected because neither model violated the bound $|\hat{y_i} - y_i| > 1$ on the clean dataset *when the rejection class was ignored for the RBF DAVE-II network* as shown by the confusion matrices in Fig. 5. Furthermore, this criteria allows us to reject the point anomalies per Eq. (5) and Eq. (6). The criteria ensures that the RBF DAVE-II network prediction is deemed erroneous if the prediction is off by more than one true class and the RBF network fails to reject the class.

On the 908 clean images, neither network had an erroneous prediction, although the RBF DAVE-II network had a false-positive rejection rate of 0.25. A false-positive can occur for the RBF DAVE-II whenever $|\hat{y_i} - y_i| > 1 \wedge +reject$ or when the prediction would have been considered safe but the



(a) DAVE-II Confusion Matrix



(b) RBF DAVE-II Confusion Matrix

Fig. 5: 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 network (a) nor the RBF DAVE-II network (b) violate the bound $|\hat{y_i} - y_i| > 1$ on the clean dataset ($n = 908$). This bound is the basis for the failure criteria with which we evaluate the anomaly detection.

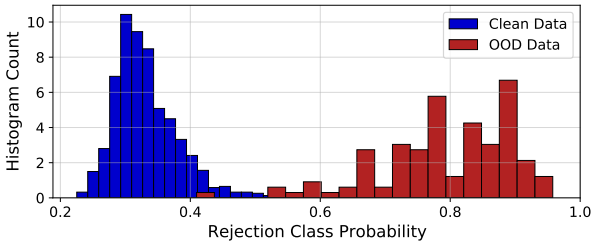


Fig. 6: A significant shift in the distribution of the rejection class probabilities was discovered for the clean data and OOD data. The threshold for the rejection class was chosen to be $\gamma = 0.6$.

network rejects the input. The false positives can likely be reduced by data augmentation, increasing model capacity, or adjusting training hyperparameters to improve the network's accuracy on clean data. Furthermore, one can add 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 $\{1, ..., c\}$.

Further for the entire 1028 images, the RBF DAVE-II network exhibited a significant shift in the confidence of the rejection class for images with a physical attack as shown in Fig. 6. From this, we select the rejection class threshold $\gamma = 0.6$ which covers the tail end of the rejection class confidence for both the clean and OOD data (see Figure 6). We did not use this threshold during the offline evaluation but we use it during runtime experiments with the physical testbed.

Finally, we evaluated the DAVE-II network and the RBF

Actions	DAVE-II	RBF DAVE-II
Out-of-track	2	8
Successful Navigation	3	4
Safe Stop	7	NA

TABLE I: Performance of the DAVE-II and RBF DAVE-II in preventing the DeepNNCar from going out-of-lane because of the physical attack.

Network	Precision (%)	Recall (%)	F1-score (%)
RBF DAVE-II	96.4	90.83	93.53
VAE based Reconstruction	88.5	90	89.24

TABLE II: Performance comparison of the RBF DAVE-II network and the VAE based reconstruction network for physical attack images.

DAVE-II network along with the rejection threshold (γ) for detecting anomalies in the 1028 images. The regular DAVE-II network predicted 50% of the images into a wrong class based on the failure criteria. Closer analysis of the predictions reveals that the network 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 [22]. On the other hand, the RBF DAVE-II misclassified 0% of the images because the RBF DAVE-II was able to successfully reject the anomalous images by putting them into the rejection class.

(b) Online Experiments: We implemented the RBF DAVE-II network along the rejection threshold $\gamma = 0.6$ for runtime anomaly detection on the physical testbed. If the rejection class probability $P(k = c + 1|x) > \gamma$, the car was instructed to stop using a reverse speed PWM. We also compared the performance of the two baseline networks in preventing the car from moving out-of-lane (see Table I). Each network is used to run the car for 12 trial runs. For each trial, we approach the attack lanes at a constant speed (0.5 m/s) and record the number of times the two networks lead the car out-of-track.

To evaluate the performance we have classified the actions of the car into three classes. An *Out-of-track* is when the car moves out of the track. A *successful navigation* is when the car does not move out of the track but completes navigating the track. Finally, in the case of the RBF DAVE-II network, a *safe stop* is when the rejection class triggers the car to safely stop. In summary, the RBF DAVE-II is able to often safely reject the physical attack and execute the stop action as compared to DAVE-II.

(c) Comparison with other approaches: We have compared the RBF DAVE-II with a reconstruction-based VAE. The VAE network has 5 convolutional layers $24/36/48/48/64$ with 5 x 5 filters and 3 x 3 filters followed by one fully connected layer with 1164 neurons. A symmetric deconvolutional decoder structure is used as a decoder. This network was trained for 150 epochs using the same training images that were used to train the RBF-DAVE II network. We used the previously collected 1028 offline images for these evaluations. Table II shows the precision, recall, and the F1-score of these networks in detecting the physical attack. As seen, the RBF DAVE-II performs slightly better in detecting the attacks and it also has smaller false positives in detecting the clean images.

(a) Clean Image (b) Poisoned Image

Fig. 7: Examples of poisoned backdoor instances for the GTSB dataset.

B. Detecting Data Poison Attacks

*1) **Experimental Setup**:* The poisoning attack is performed on the GTSB dataset [23] that has over 50,000 images of traffic signs from 43 traffic sign classes. We adapt the popular *injected pattern-key* attack where the attack is a targeted label attack that attempts to cause a DNN to predict any road sign as an 80 km/hr signboard (see Fig. 7b), whenever a backdoor key similar to a post-it note is encoded in the image. To poison the dataset we (1) randomly 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 (see Fig. 7b), 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.

*2) **Competing Baselines**:* Two baseline networks are compared for the proposed detection method. First is the ResNet20 [24] network which has 20 convolutional layers and 1 layer of softmax activation functions. The other is the RBF ResNet20 network, which replaces the final fully-connected layer with an RBF layer preceded by hyperbolic tangent function. We split the GTSB dataset into a training set of 39209 images and an evaluation set with 11430 clean images and 1200 backdoor instances. The training set is poisoned while the evaluation set is kept clean. The networks are trained on the training set for 150 epochs using the adam optimizer, with categorical cross-entropy loss for the ResNet20 network, and SoftML loss for the RBF ResNet20 network.

*3) **Results**:* **(a) Robustness evaluation:** We evaluate the robustness of the baseline networks by adjusting the number of poisoned samples (n_p) in the training data. We incrementally increase the poisoned images in the training data and record the poison attack success rates for both networks. As seen in Fig. 8, the poisoning success rate of ResNet20 is greater than 30% after only 5% of the class data has been poisoned, and the success rate increases to 90% after 35% of the data is poisoned. In comparison, RBF ResNet20 requires a larger amount of the data to be poisoned for the attack to be successful. The success rate is negligible after 35% of the class data is poisoned, and it increases to 30% only after 53% of the class data is poisoned. However, both the networks succumb to the attack after 70% of the data is poisoned. Further, in Fig. 8-b, we find that both ResNet20 and RBF ResNet20 achieve similar overall accuracy on the test data. This eliminates the possible argument that ResNet20 is simply learning the data distribution better than RBF ResNet20 and is, therefore, more likely to be successfully poisoned.

(b) Sanitizing poisoned dataset: Further, we evaluate

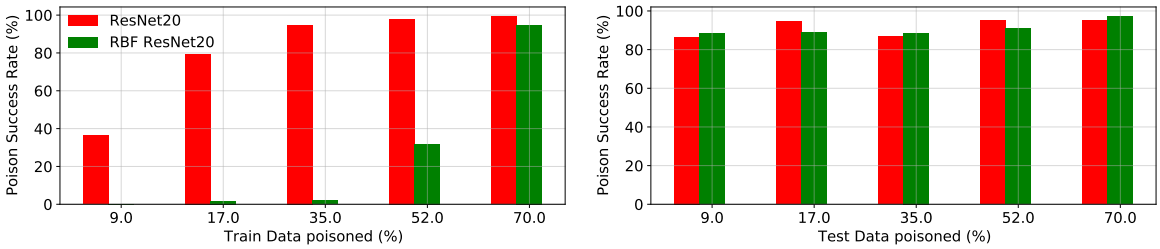


Fig. 8: The poison attack success rate for ResNet20 and RBF ResNet20 networks. (left) train data poisoned using 1200 backdoor key instances, and (right) test data poisoned using the same 1200 instances. The RBF ResNet20 network only starts to fail only when $> 35\%$ of the training data gets poisoned.

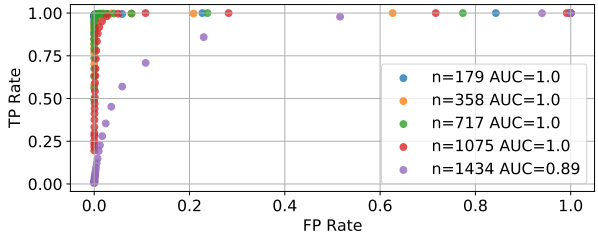


Fig. 9: The ROC curve representing the RBF ResNet20 network's data cleaning capability. The AUC is indicative of the effectiveness of the binary classifier. The AUC increases as the network's accuracy on the clean data increases and as the data set is more sparsely poisoned.

RBF ResNet20 network's capability to separate the clean and poisoned data in the poisoned dataset. We believe the network 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 $\phi_{y_i}(X^i_{poison}) > \beta$.

Fig. 9 shows the receiver operating characteristics (ROC) curve to represent the RBF ResNet20 network's data cleaning capability. The area under the curve (AUC) provides an aggregate measure of the effectiveness of a binary classification (clean data vs. poison data) for various values of β. The AUC is 1.0 until 43% of the dataset ($n_p = 1075$) has been poisoned, and it slightly drops to 0.89 when 58% of the dataset ($n_p = 1434$) has been poisoned. These results show the RBF ResNet20 network can still succeed in rejecting poisoned data despite the dataset being highly poisoned.

(c) Comparison with other approaches: We compared the RBF ResNet20 network to the activation clustering (AC) method [16] 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. For the RBF ResNet20 network, a rejection threshold of $\beta = 1.72$ was used to modestly cover the tail end of the distribution of $\phi_{y_i}(X^i_{poison})$.

Fig. 10 shows the results of adjusting n_p and comparing the two cleaning methods for the poisoned GTSB dataset. In the sparsely poisoned conditions, the RBF ResNet20 network was able to achieve on average higher true positive rates and lower false positive rates than the AC method. Even at 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 15000. However, the RBF ResNet20 network (see Fig. 10 (right)) has a higher true positive and lower false positives (not exceeding 5000) for different values of n_p. The results show that the RBF ResNet20 network is highly robust to sparsely poisoned data and begins to slowly fail as the value of n_p increases, whereas the AC method dramatically fails.

C. Resource Evaluation

The resource evaluations were performed on a desktop with AMD Ryzen Threadripper 16-Core Processor, 4 NVIDIA Titan Xp GPU's, and 128 GiB memory. We compare two approaches: (a) RBF DAVE-II network which performs both discreet steering predictions and anomaly detection, and (b) the NVIDIA DAVE-II regression network for continuous steering predictions and a reconstruction based VAE for anomaly detection. The structure of the VAE is discussed in Section V-A.

*1) **Execution Time**:* The DAVE-II network took an average of 65.4 milliseconds for steering angle predictions and the reconstruction-based VAE network took an average of 53 milliseconds for anomaly detection. In comparison, the RBF DAVE-II network only took an average of 44 milliseconds for both discreet steering predictions and anomaly detection. In summary, the RBF DAVE-II network has a 62.8% reduction in the execution time compared to the VAE. This reduction is because of the reduced operations the RBF DAVE-II network has to perform following the convolutional layers.

*2) **Memory Usage**:* The DAVE-II network utilized an average memory of 2.0 GB and the reconstruction-based VAE network utilized an average memory of 3.6 GB. In comparison, the RBF DAVE-II network only utilized an average memory of 1.1 GB. The RBF DAVE-II network utilizes less memory because of the fewer operations following the convolutional layers, compared to the VAE network that has a bulky encoder-decoder architecture, that requires higher computations.

VI. Conclusion and Future Work

This paper evaluates the efficiency of deep-RBF networks for detecting DNN related anomalies in CPS applications. We propose the use of a single deep-RBF network to perform both controller predictions and anomaly detection in CPS regression tasks. However, the use of RBF functions limits the network's applicability only to classification tasks. So, we show the steps on converting a CPS regression task (continuous steering predictions) to a classification task (discreet steering predictions)

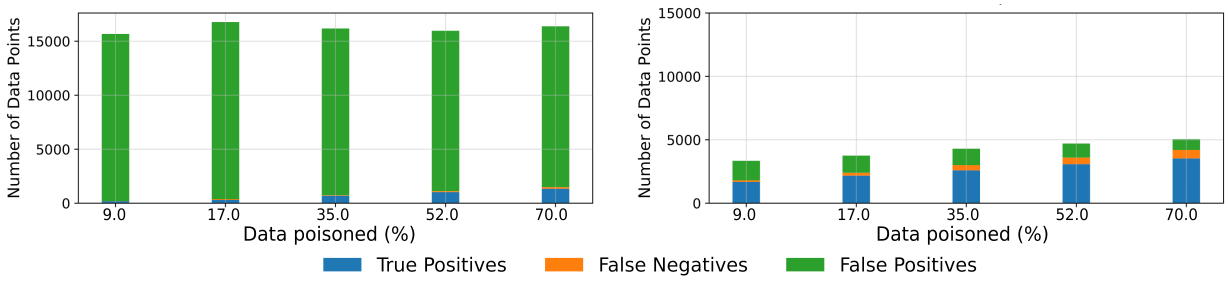


Fig. 10: (left) AC method using K-means ($n_c = 2$) and PCA ($|D| = 10$), (right) RBF ResNet20 network's rejection capability with $\beta = 1.72$. The plots show the capability of the two methods in correctly classifying the training samples as *poisoned* or *clean*. For this evaluation different percentages of the training dataset is poisoned.

and then train a deep-RBF network for class prediction and anomaly detection. To support our hypothesis, we evaluated the deep-RBF networks for two different attacks on CPS applications. First, for physical attacks on the DeepNNCar platform, the trained deep-RBF network could detect the attack with a precision of 96.4% and could prevent the car from moving out of track for 10 out of 12 trials. Further, our evaluations also show that the deep-RBF network is robust to data poison attacks over the GTSB dataset, and the network's rejection class could be used for cleaning the poisoned dataset.

Future extensions of the deep-RBF networks include: (1) extending the rejection class capability to different types of OOD data (e.g. brightness, occlusion, etc.), (2) the use of rejection capability for high level controller selection as in our previous work [9], and (3) the use of rejection class information along with the inductive conformal prediction framework [25] to better address the dynamic nature of CPSs.

Acknowledgement: This work was supported by the DARPA Assured Autonomy project and Air Force Research Laboratory.

REFERENCES

[1] 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.

[2] F. Cai and X. Koutsoukos, "Real-time out-of-distribution detection in learning-enabled cyber-physical systems," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2020, pp. 174–183.

[3] V. K. Sundar, S. Ramakrishna, Z. Rahiminasab, A. Easwaran, and A. Dubey, "Out-of-distribution detection in multi-label datasets using latent space of β-vae," *arXiv preprint arXiv:2003.08740*, 2020.

[4] C. Hartsell, S. Ramakrishna, A. Dubey, D. Stojcsics, N. Mahadevan, and G. Karsai, "Resonate: A runtime risk assessment framework for autonomous systems," *arXiv preprint arXiv:2102.09419*, 2021.

[5] A. Amini, W. Schwarting, G. Rosman, B. Araki, S. Karaman, and D. Rus, "Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 568–575.

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

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

[8] F. Crecchi, M. Melis, A. Sotgiu, D. Bacciu, and B. Biggio, "Fader: Fast adversarial example rejection," *arXiv preprint arXiv:2010.09119*, 2020.

[9] 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)*. IEEE, 2019, pp. 108–117.

[10] S. Ramakrishna, Z. Rahiminasab, A. Easwaran, and A. Dubey, "Efficient multi-class out-of-distribution reasoning for perception based networks: Work-in-progress," in *2020 International Conference on Embedded Software (EMSOFT)*. IEEE, 2020, pp. 40–42.

[11] F. Cai, J. Li, and X. Koutsoukos, "Detecting adversarial examples in learning-enabled cyber-physical systems using variational autoencoder for regression," in *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 208–214.

[12] P. Ghosh, A. Losalka, and M. J. Black, "Resisting adversarial attacks using gaussian mixture variational autoencoders," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 541–548.

[13] M. Brückner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 547–555.

[14] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *International workshop on multiple classifier systems*. Springer, 2011, pp. 350–359.

[15] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," *arXiv preprint arXiv:1706.03691*, 2017.

[16] 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.

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

[18] 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*. Springer, 2018, pp. 195–231.

[19] 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.

[20] 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.

[21] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking vision-based perception in end-to-end autonomous driving models," *arXiv preprint arXiv:1910.01907*, 2019.

[22] 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*, 2015, pp. 427–436.

[23] 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*, 2011, pp. 1453–1460.

[24] 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*, 2016, pp. 770–778.

[25] G. Shafer and V. Vovk, "A tutorial on conformal prediction," *Journal of Machine Learning Research*, vol. 9, no. Mar, pp. 371–421, 2008.