
DODEM: DOUBLE DEFENSE MECHANISM AGAINST ADVERSARIAL ATTACKS TOWARDS SECURE INDUSTRIAL INTERNET OF THINGS ANALYTICS

A PREPRINT

Onat Gungor

Department of Electrical and Computer Engineering
University of California, San Diego
ogungor@ucsd.edu

Tajana Rosing

Department of Electrical and Computer Engineering
University of California, San Diego
tajana@ucsd.edu

Baris Aksanli

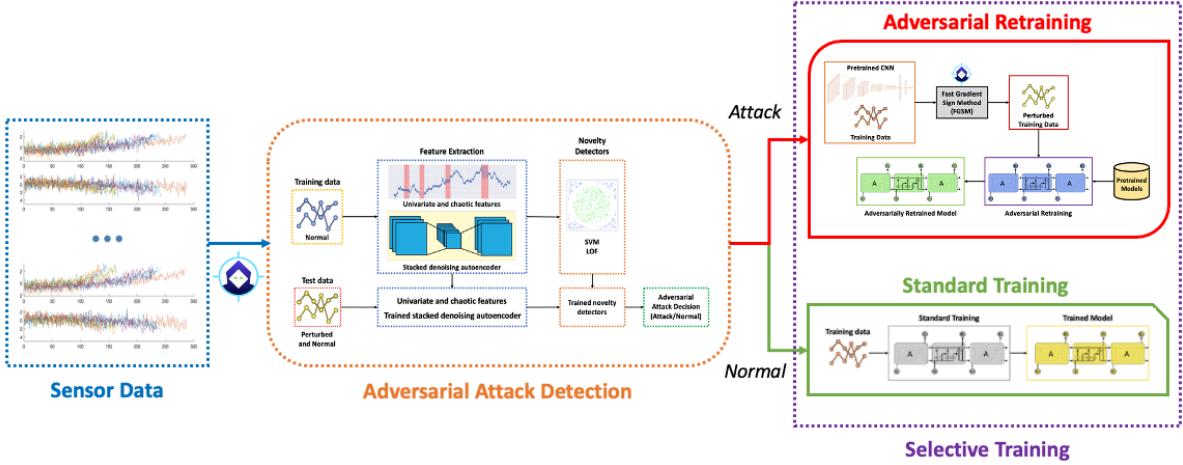
Department of Electrical and Computer Engineering
San Diego State University
baksanli@sdsu.edu

January 25, 2023

ABSTRACT

Industrial Internet of Things (I-IoT) is a collaboration of devices, sensors, and networking equipment to monitor and collect data from industrial operations. Machine learning (ML) methods use this data to make high-level decisions with minimal human intervention. Data-driven predictive maintenance (PDM) is a crucial ML-based I-IoT application to find an optimal maintenance schedule for industrial assets. The performance of these ML methods can seriously be threatened by adversarial attacks where an adversary crafts perturbed data and sends it to the ML model to deteriorate its prediction performance. The models should be able to stay robust against these attacks where robustness is measured by how much perturbation in input data affects model performance. Hence, there is a need for effective defense mechanisms that can protect these models against adversarial attacks. In this work, we propose a double defense mechanism to detect and mitigate adversarial attacks in I-IoT environments. We first detect if there is an adversarial attack on a given sample using novelty detection algorithms. Then, based on the outcome of our algorithm, marking an instance as attack or normal, we select adversarial retraining or standard training to provide a secondary defense layer. If there is an attack, adversarial retraining provides a more robust model, while we apply standard training for regular samples. Since we may not know if an attack will take place, our adaptive mechanism allows us to consider irregular changes in data. The results show that our double defense strategy is highly efficient where we can improve model robustness by up to 64.6% and 52% compared to standard and adversarial retraining, respectively.

Keywords Industrial IoT · Data-driven Predictive Maintenance · Secure machine learning · Defense Strategy

Figure 1: Our Proposed Framework (*DODEM*)

1 Introduction

Cyber-physical systems (CPSs) have become extremely intelligent and autonomous due to significant technological advancements in the Internet of Things (IoT) and machine learning (ML) [28]. Learning algorithms utilize collected sensor data to provide insight into high-level business decisions. Industrial Internet of Things (I-IoT) is a typical CPS enabling the operation, interconnection, and intelligence of industrial systems [57]. I-IoT's small-scale devices, with their limited computation and communication capabilities, make them vulnerable to potential cyber attacks [18]. An adversary can exploit these vulnerabilities to sabotage communication, prevent asset availability, and corrupt monitoring data which may have serious financial consequences [51]. For instance, the average estimated losses were \$10.7 million per breach of data among manufacturing organizations in Asia Pacific in 2019 [32]. Among different malicious attacks, adversarial attacks against deep learning (DL) draw considerable attention as DL methods are becoming more widely adopted. In an adversarial attack, an adversary can access the DL model to create slight but carefully-crafted perturbed examples to deteriorate the model performance. These attacks are especially a grave threat to data-driven predictive maintenance which aims to find an optimal maintenance schedule based on time-to-failure prediction of an industrial asset. Data-driven remaining useful life (RUL) estimation is an essential task in predictive maintenance, and it became popular owing to abundance of I-IoT system monitoring data [19]. It uses ML methods to map sensor input to the corresponding RUL values. Since ML is in the center of data-driven RUL prediction, adversarial attacks may have serious consequences such as wrong maintenance decisions leading to undetected failures in a system [33]. Hence, there is a need for detecting and mitigating such attacks to maintain the integrity and correct functioning of I-IoT systems [1].

Adversarial attack detection aims to distinguish adversarial samples from legitimate ones [31]. Detecting adversarial inputs can provide an additional protection when other defense lines have already been breached [47]. Supervised or semi-supervised ML methods can be adapted for adversarial attack detection [58]. Semi-supervised training presents a more realistic learning process since training data may not contain any adversarial samples. For semi-supervised attack detection, training starts with clean data feature extraction. Those features are then provided to ML model for its training. In testing, adversarial examples are injected into clean test data as a first step. After test data features are extracted, trained ML model makes a binary decision representing if there is an attack or not. This task is defined as *novelty detection* which classifies test data that differs in some way from the data that are available during training [40]. Novelty detection is especially suitable for adversarial attack detection because usually the amount of adversarial examples is extremely small compared to legitimate ones. There are various novelty detection algorithms proposed in the literature such as Bayesian approaches, extreme value statistics, support vector methods, and neural networks [40]. Adversarial examples are not easy to detect [5] and the extracted feature quality can affect the prediction performance significantly. Hence, there is a need for additional defense layers that can minimize the impact of wrong adversarial attack detection decisions.

Adversarial (re)training is one of the most effective approaches to defend DL models against adversarial attacks [3]. It augments the original training data with adversarial examples at each iteration, leading to more robust models when facing adversarial attacks [30]. The quality of adversarial examples included in adversarial (re)training plays a crucial role in model robustness. To craft those instances, we can adopt numerous real-world attack patterns [55]: direct attack, replica attack, and transfer attack. While the first two have some information about the target DNN (e.g., submitting

inputs to the target and receiving results, using an exact replica of the target model, etc.), transfer attack does not assume anything about the model, and thus, it represents a more feasible approach in many real-life attack scenarios. In a transfer attack, an attacker can use a good-enough approximation (i.e., substitute model) of the target model to craft adversarial inputs. Even though a substitute model is different from the target model, *transferability property* makes transfer attacks extremely dangerous. This property is satisfied when an attack developed for a substitute model is also effective against the target model [9]. It has been shown that two models with different architectures are likely to be susceptible to similar adversarial examples if they have been trained with the same data [55]. However, this assumption may not hold if an adversary does not have access to the same data for substitute and target models. Hence, adversarial (re)training should be modified by reconsidering this assumption. Furthermore, another problem with adversarial (re)training is its generalization ability. Although adversarial (re)training increases the robustness on adversarial examples, it might negatively affect the clean data accuracy, i.e., models fortified with adversarial (re)training performs worse when there is no attack[41]. This necessitates an alternative solution when there is no attack.

In this work, we propose a DOuble DEfense Mechanism framework (DODEM) that combines adversarial attack detection with another defense layer, selective training, as illustrated in Figure 1. Given sensor data, our adversarial attack detection module first determines if a sample is adversarial or not. Based on this input, the second defense module, selective training, chooses adversarial retraining or standard training. If there is an attack, our adversarial retraining approach is employed as the attack mitigation strategy to strengthen model robustness on adversarial data. If the attack detection algorithm does not detect any attack, we choose standard training strategy due to its high accuracy on clean data. For adversarial attack detection, we extract features based on multiple techniques, including statistical features, chaos-theoretic measures, and stacked denoising auto encoder. We then train two novelty detection algorithms: one-class support vector machine (OCSVM) [8] and local outlier factor (LOF) [4]. The results show that LOF predictions are significantly better than OCSVM where LOF has 89% average F_2 score while OCSVM can reach 63%. For selective training, we have two options: adversarial retraining or standard training. For adversarial retraining, we first observe that a substitute model trained on a different dataset can fool the target model more compared to training with the same dataset. Based on this observation, we modify traditional adversarial (re)training process. We first train our substitute and target models on different datasets. We then transfer the trained substitute model to create perturbed training examples and include them in adversarial (re)training process of the target model. For standard training, we use the original (clean) training dataset. Overall, our proposed solution presents a two-layer defense mechanism against adversarial attacks. Thus, it becomes more difficult for an adversarial attack to deteriorate the ML prediction performance. We test our defense strategy on the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) data set [48] which is a commonly-used benchmark dataset for RUL prediction. We compare the robustness of our strategy with standard training, and our adversarial retraining approach under white-box adversarial attacks. The results show that our method consistently outperforms standard training and adversarial retraining by adaptively selecting the correct defense strategy. Our proposed defense is by up to 64.6% and 52% more robust compared to standard and adversarial retraining respectively under different adversarial attacks. To the best of our knowledge, our work is the first that proposes a selective defense mechanism towards more robust data-driven predictive maintenance. We can summarize our contributions as follows:

- accurate adversarial attack detection approach built on state-of-the-art feature extraction and novelty detection methods,
- novel and realistic transfer attack-based adversarial retraining strategy which uses models trained on different datasets to create perturbed training instances,
- selective training approach as an additional defense layer that can decide adversarial retraining or standard training based on adversarial attack detection algorithm output,
- effective defense mechanism that can minimize the impact of adversarial attacks under white-box attacks compared to standard training and adversarial retraining.

2 Related Work

I-IoT brings full automation, higher reliability, and fine-grained control to the manufacturing world, utilizing computer networks to collect data from connected machines, and convert this data into actionable information [57]. However, these systems are susceptible to cyber attacks due to inadequate standardization, and the lack of required skills to implement them [27]. To cope with this challenge, cyber security measures should be taken such as cyber-security awareness training, keeping software programs up-to-date, installing a firewall, and using strong passwords [22]. There are several different cyber attacks that can target I-IoT systems, e.g., denial of service, eavesdropping, man-in-the-middle, side channel, zero day, and adversarial attacks against machine learning (ML) [51]. Among these, adversarial attacks against ML became extremely important since ML methods have great success in I-IoT analytics, and have

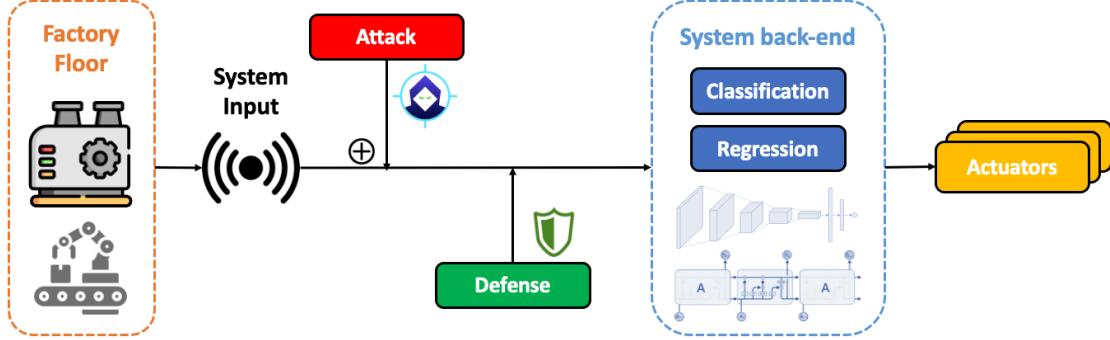


Figure 2: I-IoT Adversarial Attack Workflow

become more widespread [17]. We present a workflow of how I-IoT adversarial attacks take place in Figure 2 [28]. The workflow consists of 4 main components: (i) **System input** is the data collected from sensors in a factory environment. Different sensors can collect input data such as temperature, vibration, pressure, and proximity. (ii) **Attack** begins with the attacker exploring any vulnerability to compromise one common system input data as victim. The attacker then induces a model to craft perturbations. After tampering with the data inputted into the system back-end, the attacker can add malicious perturbations into the input data and complete the attack. (iii) **Defense** refers to protection and mitigation strategies against attacks. There are a variety of defense methods in the literature, e.g., data distortion, regularization, adversarial data detection, and adversarial training. Defense plays a key role in minimizing the impact of adversarial attacks. (iv) **System back-end** applies different ML methods to process the collected input sensor data to generate the desired information to control the actuators. Its role is crucial in extracting meaningful information from the collected data.

DL became extremely popular for data-driven predictive maintenance (PDM) applications due to its superior prediction performance [20]. However, an adversary can create carefully perturbed examples to impact DL prediction performance. To effectively generate those instances, attacker can use 3 different methods [55]: (i) *white box* methods exploit complete knowledge of a DL model, i.e., model parameters and architecture, (ii) *limited black box* methods refine adversarial input based on an output generated from the model, and (iii) *score-based black box* methods refine adversarial input based on the raw predictions (class probabilities) returned from the model. In this work, we use 3 different white box attack methods: fast gradient sign method (FGSM) [14], basic iterative method (BIM) [26], and momentum iterative method (MIM) [10]:

- **Fast Gradient Sign Method (FGSM):** FGSM first calculates the gradient of the cost function with respect to the input of the neural network. Adversarial examples are then created based on the gradient direction: $\hat{x} = x + \epsilon * sign(\nabla_x \mathcal{L}(\theta, x, y))$ where \hat{x} represents the crafted adversarial examples and ϵ denotes the amount of the perturbation.
- **Basic Iterative Method (BIM):** BIM applies FGSM multiple times with really small step size [26]. BIM perturbs the original data in the direction of the gradient multiplied by the step size α : $\hat{x} = x + \alpha * sign(\nabla_x J(\theta, x, y))$ where α is obtained by dividing the amount of perturbation (ϵ) by the number of iterations (I): $\alpha = \epsilon / I$. Then, BIM clips the obtained time series elements to ensure that they are in the ϵ -neighborhood of the original time series.
- **Momentum Iterative Method (MIM):** MIM solves underfitting and overfitting problems in FGSM and BIM by integrating momentum into the BIM [10]. At each iteration i , the variable g_i gathers the gradients with a decay factor μ : $g_{i+1} = \mu * g_i + \frac{\nabla_x J(\theta, \hat{x}_i, y)}{\|\nabla_x J(\theta, \hat{x}_i, y)\|_1}$ where the gradient is normalized by its L_1 distance. The perturbed data for the next iteration is created in the direction of the sign of g_{i+1} with a step size α : $\hat{x}_{i+1} = \hat{x}_i + \alpha * sign(g_{i+1})$

Other than attack generation methods, an adversary also needs to determine how to conduct an adversarial attack for a real-world system based on access level to the target model. We can categorize real-world attack patterns under 3 classes [55]: (i) *direct attacks* allow an adversary to submit inputs to the actual target and receive corresponding results, (ii) *replica attacks* use an exact replica of the target model to refine the adversarial input, (iii) *transfer attacks* select a substitute model which is a good-enough approximation of the target and use this model to craft adversarial examples. Among these three, *transfer attack* is the most realistic attack strategy where an adversary may not have access to the target model. *Transferability property* makes these attacks dangerous where adversarial examples created on a substitute

model can be effective on the target model even though these models are completely different. We take advantage of transfer attacks to propose a modified adversarial retraining approach.

In order to detect and defend cyber-physical systems against adversarial attacks, there are distinct defense approaches proposed in the literature, combined under 3 groups [55, 28]: (i) input defense, (ii) adversarial attack detection, and (iii) model defense. *Input defense* pre-processes input data to remove any adversarial component before the system back-end processes it. Data compression [45], data coding [43], data decomposition [21], and adding noise [60] are some example input defense strategies. *Adversarial attack detection* aims to distinguish attacked data from normal ones before model training and inference. Although it has been shown that these examples are not easy to detect [5], there are two different methods for adversarial attack detection: (i) *data feature analysis* utilizes data statistics analysis methods such as generalized likelihood ratio test [24], maximum mean discrepancy [15], and temporal consistency [59], (ii) *ML-based detection* first extracts features and then train a decision model to determine if the input is an adversarial or not. Different ML methods have been used so far, e.g., one-class classifiers [47], extreme learning machine [58], and reinforcement learning [63]. We can categorize these methods under two main groups [58]: supervised and semi-supervised learning. While supervised training includes both normal and perturbed data, semi-supervised ML model is trained only using legitimate input. It might be hard or infeasible to collect adversarial data for the training process, that is why semi-supervised training provides a more realistic learning. In our work, we use semi-supervised ML-based attack detection due to its great success in detecting attacks [47]. Santana et al. [47] use one class support vector machine and local outlier factor to detect adversarial attacks targeting a photovoltaic power plant, yet they did not propose any defense strategy.

Model defense is the last category of defenses which strengthen the model itself against adversarial attacks, i.e., increasing the robustness. This is the most heavily studied defense approach where there are numerous approaches such as gradient masking [36], defensive distillation [37], generative adversarial network [2], ensemble learning [18], certified defenses [42], and adversarial training [3]. *Adversarial training* proves to be the most effective against adversarial attacks [35]. The idea in adversarial training is to augment training data with adversarial examples to increase the model robustness. There are different adversarial training methods proposed: adversarial regularization [61], curriculum-based adversarial training [62], ensemble adversarial training [16], and adaptive adversarial training [6]. Adversarial training is formulated as a min-max problem where solving the inner maximization problem is a challenge. To create perturbed examples and incorporate them into the adversarial training, different attack generation methods can be used, e.g., fast gradient sign method [25], projected gradient descent [30]. Hence, the quality of the included examples plays a key role in model robustness. In this work, we modify adversarial training process by considering a more realistic transfer attack scenario where the attacker does not access to the training data both for substitute and target models. We train our substitute model on a different dataset and transfer the model to be used at a different dataset. Here, we create perturbed training examples and include them in adversarial retraining process of the target model to increase its robustness. Overall, this work has 4 important contributions that are missing in the state-of-the-art: (i) accurate attack detection approach, (ii) novel and realistic adversarial retraining strategy, (iii) selective training mechanism as an additional defense layer that can select adversarial retraining or standard training based on adversarial attack detection output, and (iv) effective defense mechanism to minimize the impact of adversarial attacks.

3 DOuble DEFense Mechanism Framework (DODEM)

Figure 1 demonstrates our double defense mechanism framework (*DODEM*). Given multi-variate time-series sensor data, we first divide it into training and test sets. In our first defense mechanism, adversarial attack detection, we provide the training data without any attack, yet include perturbed examples in the test data. For adversarial attack detection, we use two different novelty detection algorithms: one-class support vector machine (OCSVM) and local outlier factor (LOF). In training these algorithms, we provide our training data and extract features using 3 different techniques: statistical features, chaos-theoretic measures, and stacked denoising autoencoder. These features are then provided to the selected methods and we obtain the trained classifiers. For the test data, we use the same feature extraction methods and give those to the trained classifiers. As an output, we obtain the labels representing if a specific test instance is an adversarial or normal example. Based on this class output, we continue with the selective training module, our second defense mechanism, which decides if we should choose adversarial retraining or standard training. The selective training module is motivated by the fact that adversarial retraining increases the model robustness on adversarial examples, yet decreases accuracy on clean data. If an adversarial attack is detected, then adversarial retraining process starts. If there is no attack detected, then we select standard training where we simply train target models given training data. By making this selection, our goal is to minimize the impact of adversarial attacks and protect clean data accuracy at the same time.

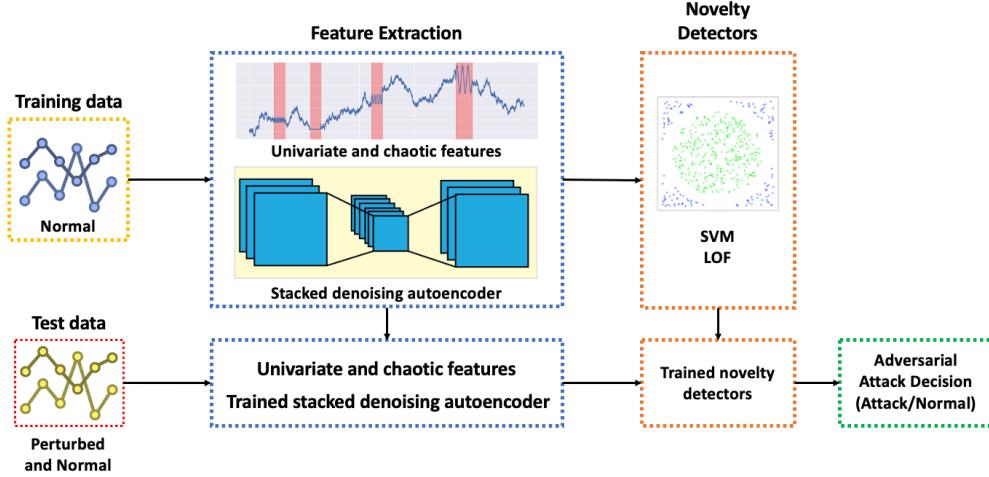


Figure 3: Adversarial Attack Detection

3.1 Adversarial Attack Detection

Figure 3 illustrates our adversarial attack detection module. After input sensor data is divided into training and test, we introduce adversarial examples solely for the test data. Here, we assume that the adversary picks random test instances to perform an adversarial attack. After salient features are extracted for training data, one-class classifiers are trained. Test data features are then provided to obtain if a random test instance belongs to adversarial or normal data. Different than the state-of-the-art, we use an extensive and diverse feature extraction methods and train two different novelty detection algorithms, increasing the generalizability of our study. The adversarial attack detection module has 3 main components which are feature generation, model training, and testing:

1) Feature generation: Feature generation (extraction) is an important process in developing predictive analytical solutions. There are different feature generation methods [58, 1, 47]: denoising autoencoder, uni-variate/multi-variate features, chaos-theoretic measures, and physical features. Our goal is to generate salient features that can better discriminate attacks from normal data. We perform our feature calculations using a sliding window to capture the temporal effects appropriately. We have n sensor measurements, $\zeta^1, \zeta^2, \dots, \zeta^n$ and a sliding window size of w . We use three categories of features, namely statistical uni-variate, chaos-theoretic measures, and denoising autoencoder.

a) Uni-variate features: For each individual measurement ζ^i , we can formulate windowed segment of measurements at time t as $\psi_t^i = \zeta_{t-w}^i, \zeta_{t-w+1}^i, \dots, \zeta_t^i$. We calculate 8 features for ψ_t^i : minimum, maximum, mean, median, standard deviation, range, mean to maximum ratio, and minimum to maximum ratio.

b) Chaos-theoretic measures: Chaos theory focuses on underlying patterns and deterministic laws of dynamical systems. There are two main reasons why we utilize chaos-theoretic measures [1]: (i) although the perturbations are small, they contribute to the complexity of the signal, and (ii) since the perturbations are seemingly chaotic, they may be observable using measures of chaos. For the first reason, we adopt the Sample Entropy [44] which quantifies the complexity of time-series data. For the second reason, we select 3 different features: (i) Detrended Fluctuation Analysis (DFA) [39] is a measure of the statistical self-affinity of a non-stationary signal, (ii) Hurst exponent [23] is a measure of the long-term memory of a time series, and (iii) Lyapunov exponent [46] measures chaos and unpredictability. Overall, we have 4 chaos-theoretic measures.

c) Denoising autoencoder: In deep representation learning, one can stack up multiple layers of shallow learning blocks, train it, and use the last layers to learn useful representations of input data. Autoencoder (AE) is a type of neural network that is commonly used for representation learning. It consists of encoder and decoder sub-models where the encoder compresses the input and the decoder attempts to recreate the input from the compressed version provided by the encoder. To prevent AE to learn trivial hidden representations, denoising AE (DAE) add noise or mask some of the input values. Stacked denoising autoencoder (SDAE) [53] stack up individual DAEs to form a deep network where the outputs of the hidden neurons at the lower layer of DAE is the input to the upper layer of DAE. We use the hidden representations at the last layer of the SDAE to obtain 25 features.

2) Model training: The goal of model training is to map extracted features to binary decisions (attack or normal). We frame this problem as *novelty detection* since the quantity of adversarial data is significantly smaller than normal data.



Figure 4: Random Adversarial Attack Simulation

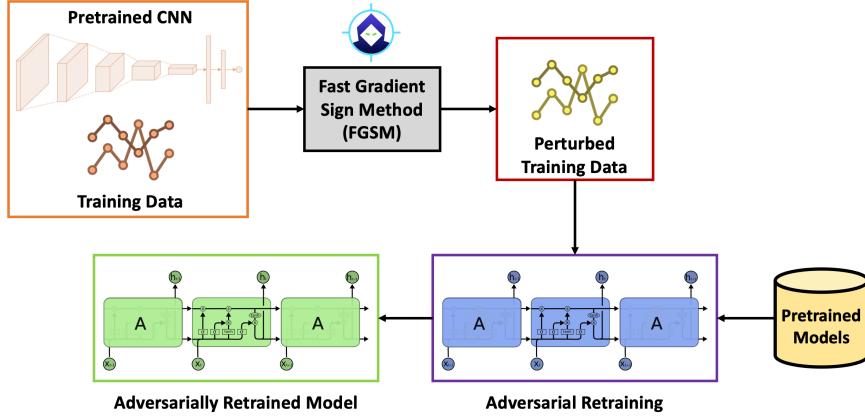


Figure 5: Adversarial Retraining

The problem of novelty detection can be seen as one-class classification where one class has to be distinguished from all other possibilities [34]. In our work, we utilize two different one-class classification algorithms, namely one-class support vector machine and local outlier factor:

a) One-class Support Vector Machine (OCSVM): OCSVM is a variation of traditional SVM where instead of using a hyperplane to separate two classes of instances, OCSVM finds the smallest hypersphere to encompass all of the instances [49]. In testing time, OCSVM can identify whether an instance is within this hypersphere. Based on this, it classifies the instance as adversarial or normal.

b) Local Outlier Factor (LOF): LOF measures the local deviation of a given data point with respect to its neighbors [4]. Based on this measurement, LOF identifies data points with a lower density as novelties. In order to decide how many neighbors to consider, k-Nearest Neighbors algorithm is used.

3) Model testing: To evaluate the performance of our attack detection module, we simulate an attacker that creates adversarial attack samples. In order to have a realistic attack scenario, we assume that the I-IoT system does not have access to these samples, hence they are not included in model training but added to the test data set. When simulating an adversarial attack scenario, we adopt a random attack fashion as illustrated in Figure 4 where red squares illustrate adversarial instances while green ones are regular instances. We control the amount of adversarial samples with a parameter, i.e., adversarial sample ratio. For instance, if this ratio is 5%, then we insert 5% adversarial samples into the original test data set. We create those examples using FGSM due to its efficiency. After we construct the test data which consists of both normal and adversarial instances, we continue with the feature generation for which we calculate uni-variate features, chaos-theoretic measures, and use the last layer of the trained SDAE. Then, we provide these features to the trained classifiers to obtain the corresponding labels (attack/normal). We ultimately calculate the required metrics to measure the performance of our attack detection module, e.g., recall, F_2 score, ROC AUC score.

3.2 Selective Training

The goal of our selective training module is to select adversarial retraining or normal training based on the output from adversarial attack detection module. If an instance is classified as adversarial, then we use adversarial retraining, otherwise we are safe with standard training. We propose this selection approach because of two characteristics of adversarial retraining: (i) poor prediction performance on clean data and (ii) high robustness on adversarial data. We can eliminate the weakness of adversarial retraining by selecting standard training when there is no attack.

1) Adversarial Retraining: Figure 5 depicts our adversarial retraining approach. We modify the traditional adversarial training in terms of three aspects to make it more realistic: (i) we use transfer attack strategy to produce perturbed

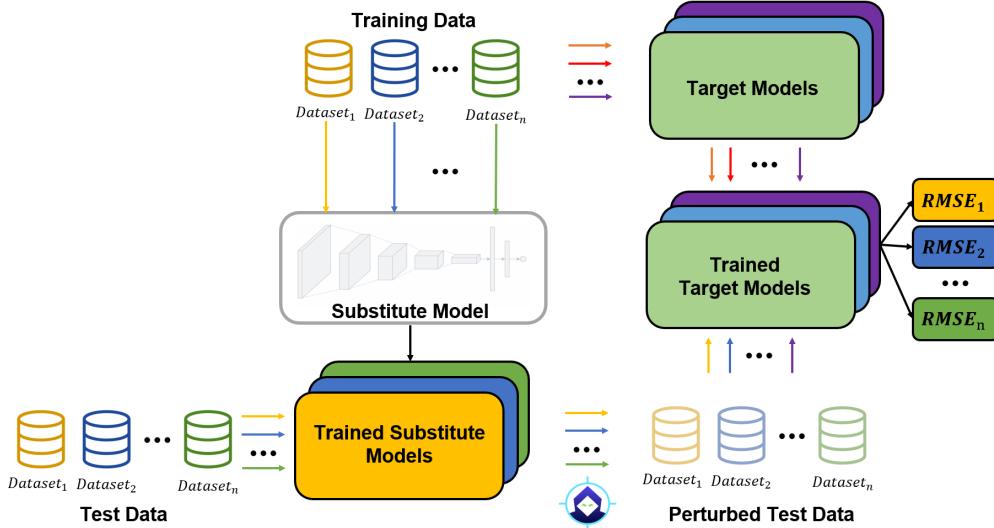


Figure 6: Transferability Framework

training examples to be included in adversarial retraining, (ii) in our transfer attack, we select different DL model structures as substitute and target models, (iii) we prohibit an adversary to use the same data for substitute and target model training. To modify the standard adversarial (re)training process, we first need to decide which trained model(s) to be included in adversarial retraining process. To reach that goal, we perform transferability analysis and substitute model selection:

Transferability Analysis: For transfer attacks, it has been shown that two models with different architectures are likely to be susceptible to similar adversarial examples if they have been trained with the same data [55]. However, we might have a scenario where an adversary does not access to the same training data for substitute and target model training. In that case, we would like to test the transferability of perturbed examples. We present our transferability framework in Figure 6. Given training data, we first train our substitute model and target models. Note that used training data is different for substitute model and target models training. Then, we create perturbed test examples using the trained substitute model via an adversarial attack since adversary does not have access to the target models. We then transfer those examples to the target models to check how much each substitute model can fool target models. We measure this based on the average prediction error value, i.e., root mean square error (RMSE), over all target models. The higher RMSE value represents that a substitute model is able to fool the target models more.

Substitute Model Selection: Based on the results of the transferability analysis, we transfer the trained substitute models in an iterative manner where we start with the model that can fool the target models the most, continue with the second, and so on. To do that, we first sort the transferability analysis output error values in a descending order and start with the substitute model which gave the highest RMSE. We then craft perturbed training instances using the selected substitute model to be included in adversarial retraining. We terminate adding models under two conditions: (i) until we can no longer improve the adversarial retraining model robustness, and (ii) when we end up with the same dataset which is not allowed.

Selected DL Models: We select 1-D CNN as our substitute model and 7 different target models from different architectures (recurrent and hybrid), increasing the generalizability of our study:

- **1-D Convolutional Neural Network (CNN):** 1D convolutional layer slides kernels across a sequence, producing a 1D feature map per kernel and each kernel learns to detect a single very short sequential pattern [13]. We adopt the 1-D CNN network proposed by Li et al. [29] which contains five consecutive CNN layers, Flatten (Dropout) layer, one fully-connected layer (with 100 nodes).
- **Recurrent Neural Network (RNN):** RNN is a time-aware feedforward neural network [13]. Our network contains 3 RNN layers having 64, 32, and 16 units connected to 2 fully connected feed forward neural networks (each with 8 units).
- **Long Short-Term Memory (LSTM):** LSTM has special memory cells to store information for longer. Updates in this cell can happen by the activation of three distinctive gates: 1) forget gate (the memory cell is cleared completely), 2) input gate (memory cell stores the received input), and 3) output gate (next neurons

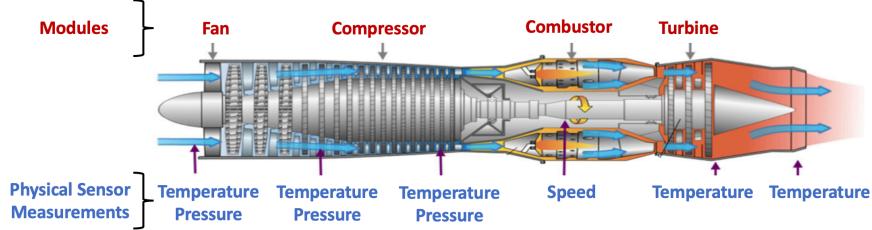


Figure 7: Engine Diagram Simulated in C-MAPSS [48]

Table 1: C-MAPSS Data Set

Data Set	FD001	FD002	FD003	FD004
Train trajectories	100	260	100	249
Test trajectories	100	259	100	248
Max/Min cycles for train	362/128	378/128	525/145	543/128
Max/Min cycles for test	303/31	367/21	475/38	486/19
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

obtain the stored knowledge from the memory cell) [12]. We adapt a similar network structure where RNN layers are replaced with LSTM layers.

- **Bi-directional LSTM (BLSTM):** BLSTM also considers future data by adding a backward direction to LSTM networks [54]. The overall network structure is similar to LSTM model where LSTM layers are replaced with BLSTM layers.
- **Gated Recurrent Unit (GRU):** GRU is a simplified version of LSTM [7]. Specifically, forget and input gates are controlled by a single gate controller and there is no output gate, instead a new gate controller decides which part of the information to be transferred. We use the same network structure as LSTM except we change the LSTM layers to GRU.
- **Bi-directional GRU (BGRU):** Similar to BLSTM, BGRU takes future data into consideration [50]. We construct this model by simply replacing BLSTM layers with BGRU layers.
- **CNN-GRU (CGRU):** We connect our CNN and GRU networks in parallel. On one path, we have our 1-D CNN architecture, on another path we have our GRU model. These are then connected to fully connected NN with 100 units.
- **GRU-LSTM (GLSTM):** We combine our GRU and LSTM models in parallel. GRU and LSTM paths are concatenated and connected to fully connected NN with 100 units.

For the sake of simplicity, we show substitute and target model training in Figure 6. In Figure 5, pretrained CNN corresponds to the substitute model that was able to fool the target models the most. In Section 4.4, we provide an experimental analysis how we select this model. Selected pre-trained CNN and training data, adversary first creates perturbed training instances using FGSM. These created examples are then used in retraining of the target models. As an output, we obtained the adversarially retrained target models.

2) Standard Training: We select standard training if no attack is detected for a specific instance. This is a pretty standard training process where training data is used to train different target models.

4 Experimental Analysis

4.1 Dataset Description

NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [48] is a benchmark dataset for remaining useful life (RUL) prediction. This dataset consists of multiple aircraft engines simulated under various operating and fault conditions. Figure 7 depicts the simplified version of simulated engine diagram and its four major components: fan, turbine, compressor, and combustor. The data is collected using different sensors (e.g. temperature, pressure) placed on these components. NASA C-MAPSS includes 4 different datasets in increasing complexity: FD001~FD004. Table 1 presents the dataset and their corresponding features such as number of train and test trajectories, maximum/minimum

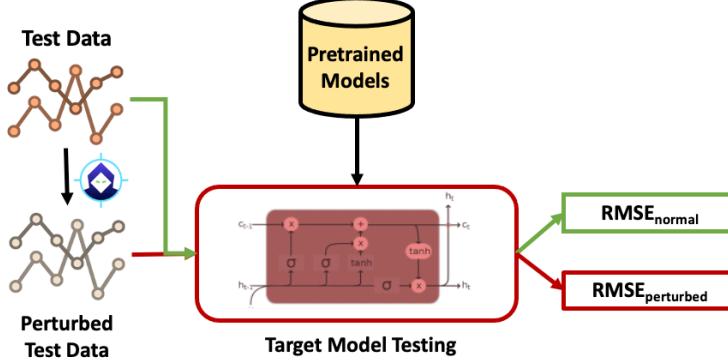


Figure 8: Testing Framework

number of cycles, etc. We observe that FD001 is the simplest data set and FD004 is the most complicated one with the highest number of operating and fault conditions. For each dataset, we have separate training and test data. The training data contains the entire lifetime of an engine and test data is terminated at some point before engine failure. The goal is to predict RUL for the test data. C-MAPSS feature columns include the engine ID, cycle index, three operational settings, and 21 sensor measurements.

4.2 Experimental Setup

We run all experiments on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor. For adversarial attack detection, and adversarial retraining, we use FGSM since it can create adversarial examples efficiently, i.e., fast and less detectable adversarial example generation [56]. In order to test *DODEM* against adversarial attacks, we use FGSM, basic iterative method (BIM), and momentum iterative method (MIM) [11, 10]: amount of perturbation(ϵ)=0.1, step size(α)=0.001, number of iterations(I)=100, decay factor(μ)=1.

Feature Generation: For statistical uni-variate features, we use the NumPy library. For the chaos-theoretic measures, we use the nolds module in Python. Lastly, we use a 2-layer SDAE network, and the network structures for the first-layer DAE and second-layer DAE are 20-50-20 and 50-25-50, respectively, where 50 hidden neuron outputs of the first-layer DAE are used as the inputs to the second-layer DAE. The outputs of the 25 hidden neurons of the second-layer DAE are taken as the features, which give us 25 learned features.

Adversarial Attack Detection: We use the scikit-learn library [38] to implement one-class classifiers. For OCSVM, we select the *rbf* kernel and try different ν values from the set {0.001, 0.005, 0.01, 0.05, 0.1, 0.2}. For the LOF, we set the number of neighbors to 20 while selecting contamination value from the same set we used for OCSVM ν values. To measure the prediction performance of these algorithms, we use the F_β score metric where we set the β to 2 to weigh recall higher than precision:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \quad (1)$$

F_2 score is a more suitable metric for attack detection because it is more important to classify adversarial samples correctly as much as possible, i.e., true positives are more important. The higher F_2 corresponds to a higher attack detection performance. We try different adversarial test ratios ({0.01, 0.05, 0.1, 0.2}) in our experiments. This ratio represents the amount of adversarial data within the entire test dataset.

DL Models: The following hyper-parameters are selected for DL training: Adam optimizer with learning rate 0.001, elu activation function, batch size of 128, and a max number of epochs of 150 where callback is activated (patience is set to 10 for validation data), and sliding time window size of 80. For adversarial retraining, we retrain the target models for 30 more epochs.

Robustness Metric: RUL prediction is a typical regression problem where input data is mapped to a continuous output value. To measure the robustness of our proposed approach against adversarial attacks, we first measure the root mean square error (RMSE) of the RUL prediction process as follows:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N \epsilon_i^2} \quad (2)$$

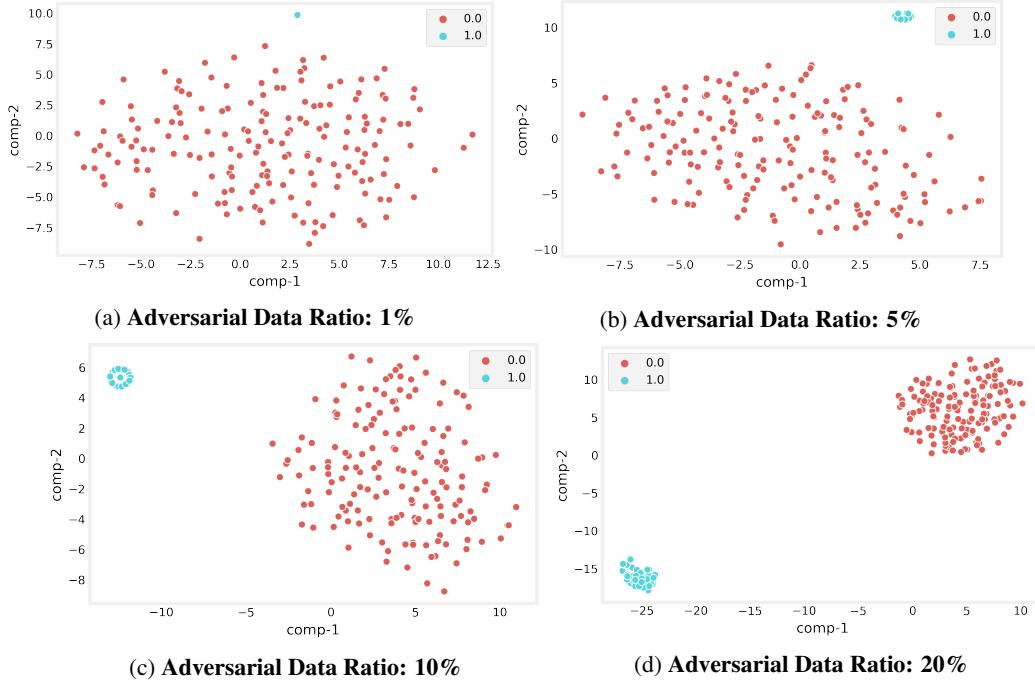


Figure 9: FD002 t-SNE Projection

where \mathcal{N} is the number of samples, ϵ is the difference between the estimated RUL (RUL_{est}) and the true RUL (RUL_{true}). Smaller RMSE means that we have a more robust model since the model is affected less by adversarial attacks.

Testing Framework: We measure the robustness of our proposed method with three white-box adversarial attack scenarios, namely FGSM, BIM, and MIM. These are 3 most common attacks for time series data [1]. Figure 8 illustrates our white-box testing framework. Given clean test data, the adversary first crafts adversarial test samples using pre-trained target models. Clean and perturbed test data are given to the target models and we measure normal and perturbed RMSE, respectively over all target models.

4.3 Adversarial Attack Detection

To understand how selected features are helpful in distinguishing adversarial samples from legitimate ones, we project the selected 37-D feature space to a 2-D space using t-distributed stochastic neighbor embedding (t-SNE) [52]. Figure 9 shows the t-SNE projection for FD002 (Note that we only present FD002 for clarity in this figure. We obtain similar figures for the other datasets). Each sub-figure corresponds to a different adversarial data ratio, varying from 1% to 20%. While red dots indicate normal data, light blue are the adversarial samples. We can see that the normal and attack samples are locally separable in the feature space. This indicates that our attack detection methods should perform well in separating normal data and crafted adversarial data.

For the selected detection methods, we report the best F_2 scores among all possible ν and contamination values for OCSVM and LOF, respectively. We similarly vary the adversarial data ratio from 1% to 20%. Figure 10 shows the attack detection results where each sub-figure corresponds to a different dataset. In each sub-figure, x-axis denotes the adversarial data ratio and y-axis presents F_2 scores. While blue color represents OCSVM, green corresponds to LOF. We observe that LOF significantly outperforms OCSVM at all datasets. While LOF has 89% average F_2 score over all datasets and adversarial data ratios, OCSVM reaches up to 63%. As adversarial data ratio gets bigger, the predictor performance becomes better. For instance at FD003, LOF has an F_2 score of 78% and 98% for 1% and 20% adversarial data ratios, respectively. The classifiers have difficulty in recognizing adversarial samples when they are extremely scarce. As the test data has more adversarial samples, distinguishing these from the normal ones becomes easier. We also observe that as the complexity of the data increases (related to number of operating and fault conditions), the attack detection performance gets worse. LOF can reach 100% F_2 score at FD001 (simplest dataset), but only 80% F_2 score at FD004 (most complex). Based on these observations, we will only report LOF attack detection results in Section 4.5 as LOF has much better adversarial attack detection performance.

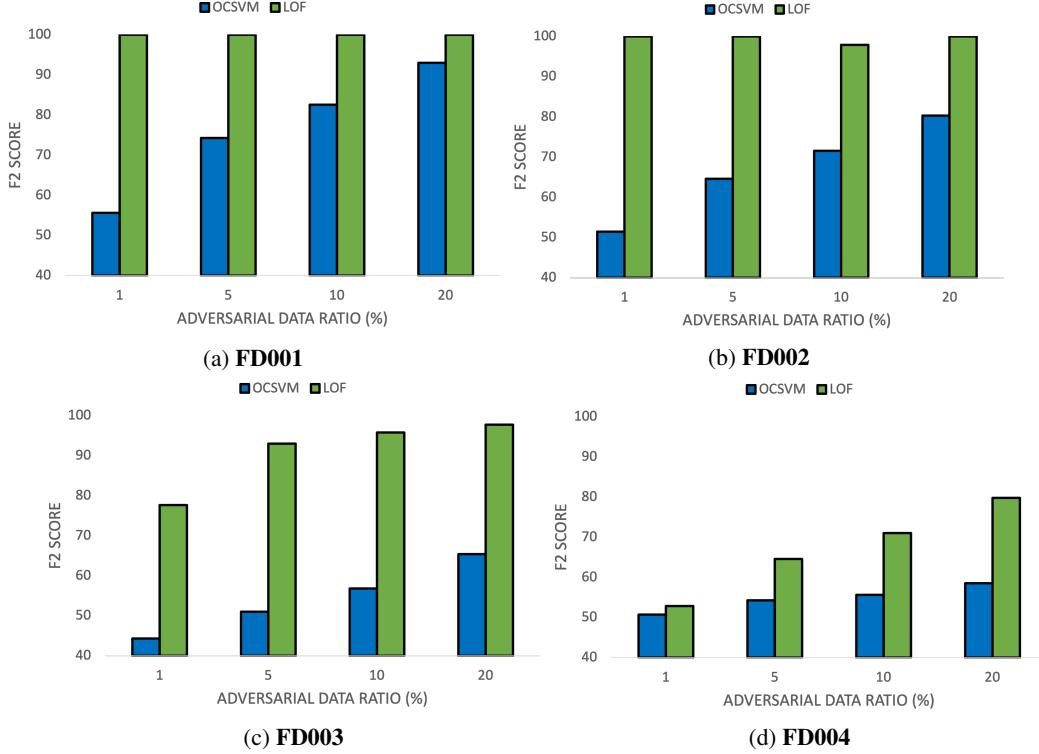


Figure 10: Adversarial Attack Detection Results

4.4 Adversarial Retraining

Transferability Experiments: To determine which trained model(s) to include in the adversarial retraining process, we first conduct transferability experiments. The goal is to test the hypothesis that two models with different architectures are likely to be susceptible to similar adversarial examples if they have been trained with the same data [55]. In these experiments, we train a CNN model on four different datasets and transfer the trained model to other datasets to create perturbed test examples. We then measure the target model prediction performance under adversarial data. In total, we have 16 different scenarios, e.g., train CNN at FD001, transfer this model to FD002, FD003, and FD004. Figure 11 summarizes the transferability results. In this figure, x-axis denotes the test data while y-axis reports the average RMSE value over all target models. Different colors correspond to CNN models trained on different datasets, for example blue denotes CNN model trained at FD001. For each dataset, we would expect that the RMSE values to be highest with the CNN model trained on the same data set. However, the RMSE values are worse with models that are trained on different datasets. For example, for FD002, FD001-trained CNN can fool the target models more, followed by FD003. FD001 trained CNN has an RMSE of 79.9 while this value decreases to 46.7 when same dataset is used. We can have a similar observation with other datasets as well. For FD004, FD001 and FD003 lead to worse prediction performance than FD004-trained CNN. Our transferability experiments conclude that models trained on different datasets can negatively impact the target models more.

Proposed Adversarial Retraining Results: Motivated by our transferability results, we modify the traditional adversarial retraining by adding models trained on different datasets. We perform this addition iteratively where we start adding the trained model that can fool the target model the most, then the second, and so on. For example, for FD002, we add the FD001-trained CNN first (since it leads to the worst prediction performance) and then FD003-trained CNN. We stop adding models based on two conditions: (i) until we can no longer improve model robustness and (ii) when we end up with the same dataset, e.g., adversary cannot use FD002 trained model for FD002 adversarial retraining. After we determine the models to be added, we transfer these to the target dataset to craft adversarial training instances. When analyzing the validation data, we observed that the amount of training data to be used in adversarial retraining has an impact on model robustness. As a result, we choose the number of crafted adversarial samples to be 1% of the training data size. Then, we provide these adversarial samples to the target models for retraining. We then measure RMSE under adversarial test examples.

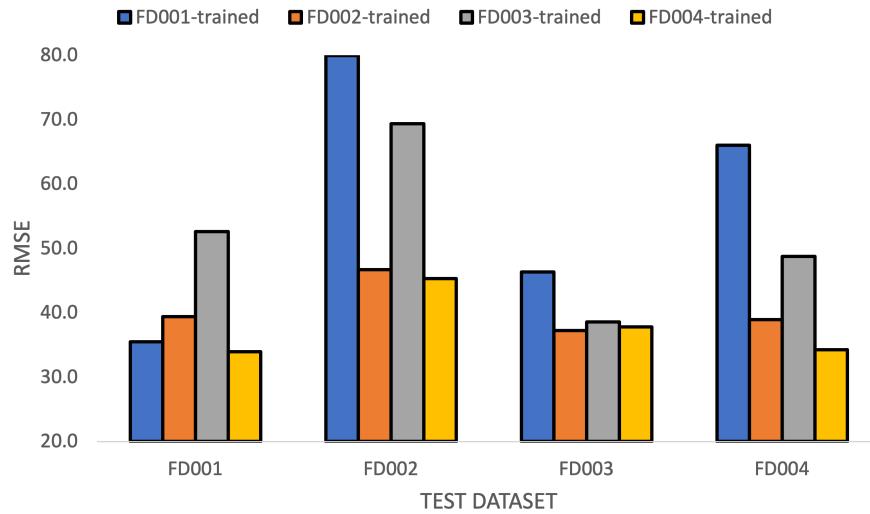


Figure 11: Transferability Experiments

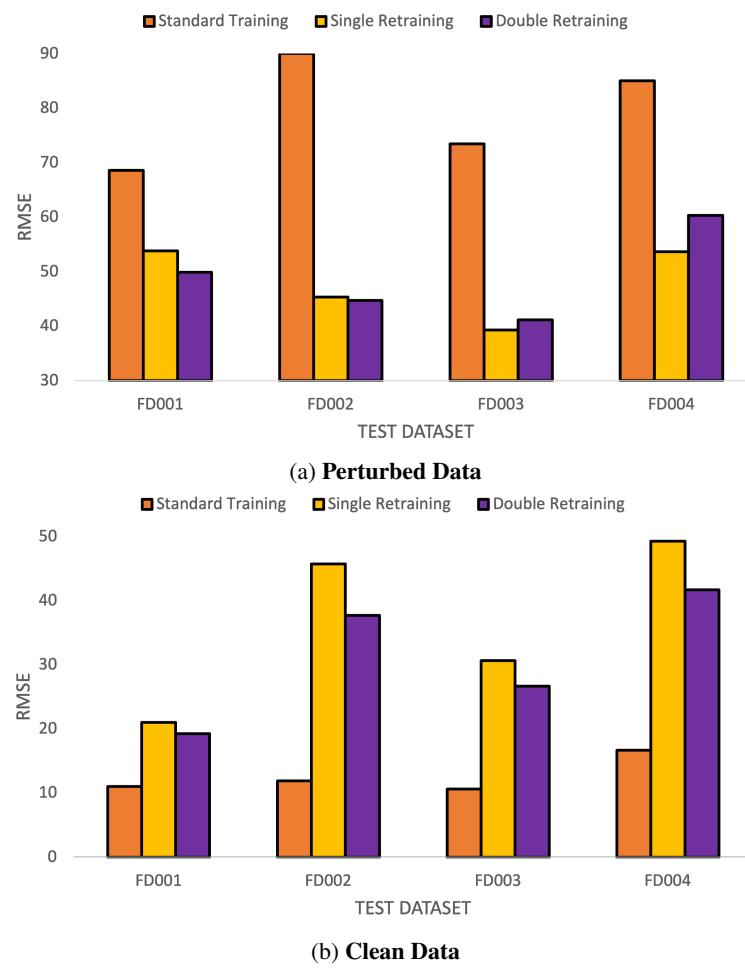


Figure 12: Adversarial Retraining Results

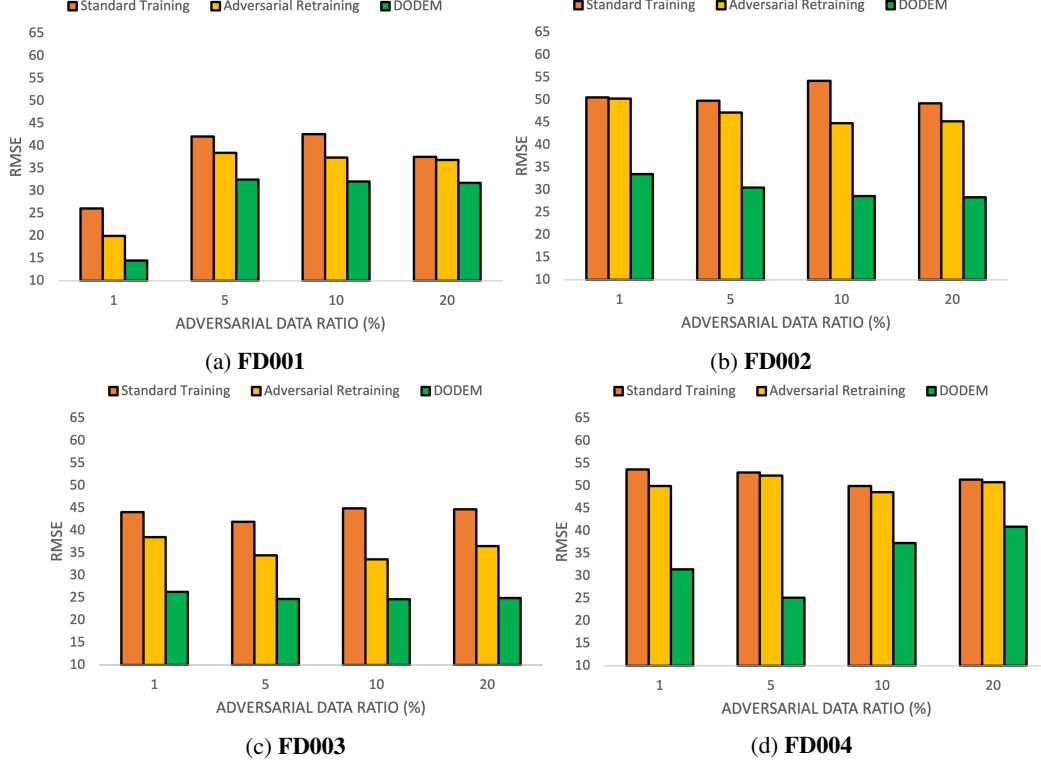
Figure 13: *DODEM* Robustness (FGSM)

Figure 12 presents the adversarial retraining results on perturbed (Figure 12a) and clean (Figure 12b) data compared to standard training. For both sub-figures, we have test dataset on the x-axis and average RMSE (over all target models) on the y-axis. Orange color represents standard training, yellow color denotes adversarial retraining with one model (single retraining), and purple corresponds to adversarial retraining with two models (double retraining). Here, adversarial retraining (single/double) corresponds to our proposed retraining approach. To measure the perturbed and clean data RMSE values, we consider two extreme cases. For perturbed data, we use a test dataset with only adversarial samples, for clean data our test data does not include any adversarial example. Figure 12a shows that our adversarial retraining approach improves the robustness significantly when there is adversarial attack. We improve model robustness by up to 67% (40% on average). We also observe that introducing an additional trained model into adversarial retraining, i.e., switching from single to double retraining, does not bring any significant advantage. Although double retraining brings 7% and 1% more robustness for FD001 and FD002 respectively, it leads to 5% and 11% less robustness for FD003 and FD004. When RMSE values are averaged over all datasets, single retraining in fact outperforms double retraining by 2%.

Figure 12b shows that adversarial retraining performance on clean data is significantly worse than standard training. Although double retraining has smaller error compared to single retraining, they both under-perform compared to standard training when using clean data. This creates an important issue for adversarial training because in a realistic scenario, we may not know which samples are targeted for an adversarial attack or if there is an attack or not at all. Our framework, *DODEM* solves this problem by performing selective training which combines adversarial training with standard training based on adversarial attack detection output. *DODEM* leads to lower error on both clean and perturbed data, providing a more robust solution.

4.5 DODEM Robustness

We compare *DODEM* with two different training settings: standard training and our adversarial retraining with a single model. We evaluate *DODEM* with different adversarial test data ratios (ranging from 1% to 20%) and attack generation methods (FGSM, BIM, and MIM) to represent a variety of attack scenarios, e.g., a stealth attack with a small number of adversarial samples vs. a heavy attack that has a large number of adversarial samples. We report the average RMSE for each attack method calculated over perturbed test data which includes both normal and adversarial samples. Figure 13, Figure 14, and Figure 15 present *DODEM* RMSE values under FGSM, BIM, and MIM attacks

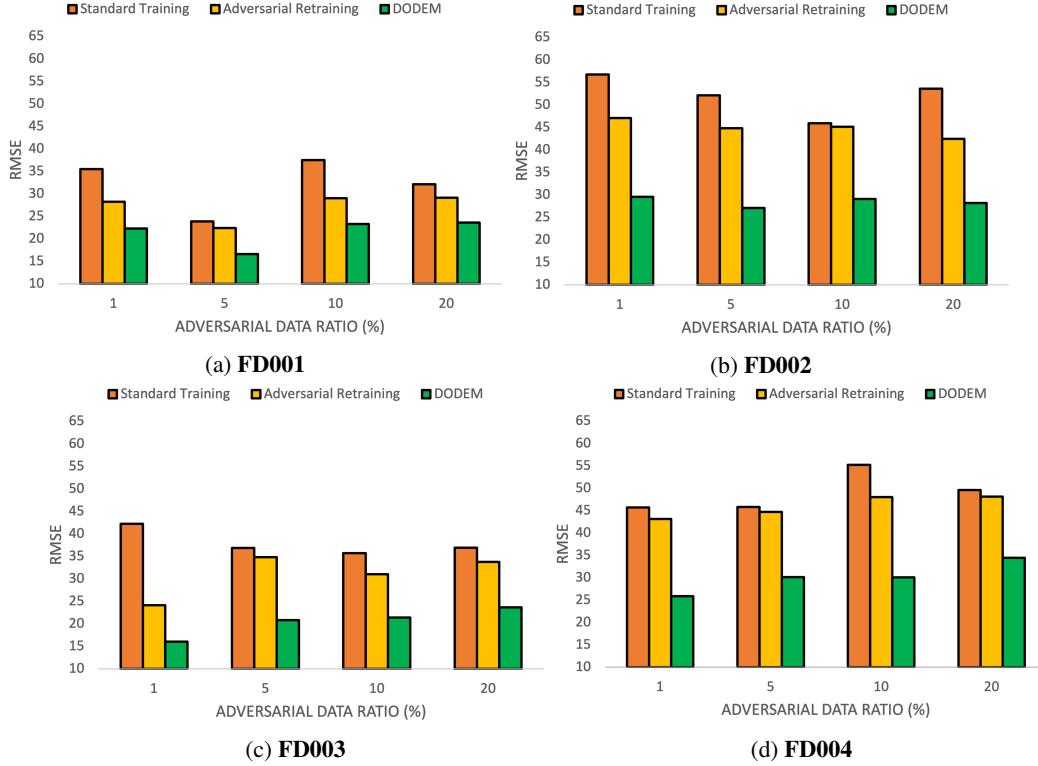
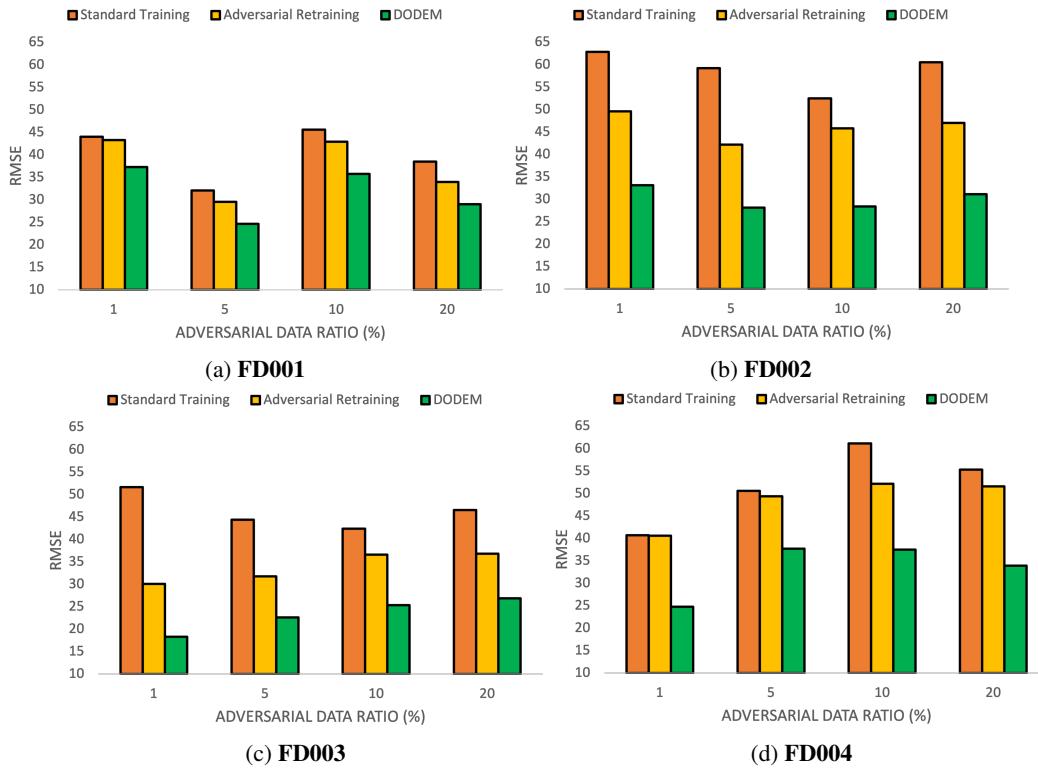
Figure 14: *DODEM* Robustness (BIM)Figure 15: *DODEM* Robustness (MIM)

Table 2: *DODEM* Robustness Improvement (%) over Standard Training and Adversarial Retraining (FGSM)

Dataset/Training Setting	Average		Maximum	
	Standard	Adversarial	Standard	Adversarial
FD001	26.8	17.7	44.4	27.3
FD002	40.5	35.5	47.3	37.3
FD003	42.7	29.5	45.1	31.7
FD004	34.9	32.9	52.6	52.0

Table 3: *DODEM* Robustness Improvement (%) over Standard Training and Adversarial Retraining (BIM)

Dataset/Training Setting	Average		Maximum	
	Standard	Adversarial	Standard	Adversarial
FD001	33.0	21.4	37.9	25.8
FD002	45.0	36.5	48.1	39.6
FD003	45.4	33.7	62.0	40.3
FD004	38.5	34.7	45.6	40.0

respectively with standard training and our adversarial retraining setting. Lower RMSE means that a model is more robust against adversarial attacks. Each figure consists of the dataset results. At each sub-figure, x axis denotes the adversarial data ratio in percentage while y-axis is the RMSE. *DODEM* is represented by green color while standard and adversarial training is depicted with orange and yellow colors, respectively. *DODEM* has the smallest RMSE value across all attacks, datasets, and adversarial data ratios consistently. This clearly shows that *DODEM* is the most robust solution against adversarial attacks. We also see that as the dataset complexity increases (from FD001 to FD004), *DODEM* clearly brings more robustness where there is a larger gap among *DODEM* and other methods. This means that our method can perform even better on more complex datasets. We report *DODEM*'s average and maximum robustness improvement over standard training and adversarial retraining in Table 2, Table 3, and 4. *DODEM* improves the robustness by up to 64.6% and 52% for standard and adversarial training respectively. On average, *DODEM* brings up to 49% and 36.5% more robustness against adversarial attacks, proving to be an effective defense mechanism against adversarial attacks.

5 Conclusion

Industrial Internet of Things (I-IoT), as a typical cyber physical system (CPS), enables fully automated production systems by continuously monitoring devices and analyzing collected data. After an adversary access to the I-IoT system by exploiting its vulnerabilities, they can manipulate legitimate inputs, corrupting ML predictions and causing disruptions in the production systems. Hence, there is a need for sophisticated defense mechanisms that can protect I-IoT systems against adversarial attacks. In this work, we propose double defense mechanism against adversarial attacks. Our first defense method, adversarial attack detection, aims to distinguish adversarial attacks from legitimate ones. To reach that goal, we first extract salient features by using multiple distinct feature extraction methods which are then provided to one-class classifiers. We selected one-class support vector machine (OCSVM) and local outlier factor (LOF). We observed that LOF has a better prediction performance with 89% average F_2 score. selective training, our second defense mechanism, decide if an adversarial retraining or standard training should be used based on LOF predictions. If there is an attack, adversarial retraining is used. We modify the standard adversarial retraining by adding adversarial examples via DL methods trained at different datasets, i.e., transfer attack. We also selected different substitute and target models to make the retraining more realistic. Our results show that our adversarial retraining can improve the model robustness by up to 67% (40% on average). Most importantly, our double defense mechanism can improve the robustness by up to 64.6% and 52% compared to standard and adversarial training respectively under different adversarial attacks.

Table 4: *DODEM* Robustness Improvement (%) over Standard Training and Adversarial Retraining (MIM)

Dataset/Training Setting	Average		Maximum	
	Standard	Adversarial	Standard	Adversarial
FD001	21.2	15.4	24.6	16.7
FD002	48.6	34.6	52.5	38.0
FD003	49.0	31.4	64.6	39.3
FD004	35.5	31.3	39.1	39.0

References

- [1] Mubarak G Abdu-Aguye et al. Detecting adversarial attacks in time-series data. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3092–3096. IEEE, 2020.
- [2] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.
- [3] Tao Bai et al. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.
- [4] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.
- [5] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.
- [6] Minhao Cheng, Qi Lei, Pin-Yu Chen, Inderjit Dhillon, and Cho-Jui Hsieh. Cat: Customized adversarial training for improved robustness. *arXiv preprint arXiv:2002.06789*, 2020.
- [7] Kyunghyun Cho et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint*, 2014.
- [8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [9] Ambra Demontis et al. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium (USENIX security 19)*, pages 321–338, 2019.
- [10] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.
- [11] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Adversarial attacks on deep neural networks for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.
- [12] André Gensler, Janosch Henze, Bernhard Sick, and Nils Raabe. Deep learning for solar power forecasting—an approach using autoencoder and lstm neural networks. In *2016 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 002858–002865. IEEE, 2016.
- [13] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.
- [14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [15] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.
- [16] Onat Gungor, Tajana Rosing, and Baris Aksanli. Dense-defense: Diversity promoting ensemble adversarial training towards effective defense. In *2022 IEEE SENSORS*, pages 1–4. IEEE, 2022.
- [17] Onat Gungor, Tajana Rosing, and Baris Aksanli. Res-hd: Resilient intelligent fault diagnosis against adversarial attacks using hyper-dimensional computing. *arXiv preprint arXiv:2203.08148*, 2022.
- [18] Onat Gungor, Tajana Rosing, and Baris Aksanli. Stewart: Stacking ensemble for white-box adversarial attacks towards more resilient data-driven predictive maintenance. *Computers in Industry*, 140:103660, 2022.
- [19] Onat Gungor, Tajana S Rosing, and Baris Aksanli. Dowell: diversity-induced optimally weighted ensemble learner for predictive maintenance of industrial internet of things devices. *IEEE Internet of Things Journal*, 9(4):3125–3134, 2021.
- [20] Onat Gungor, Tajana S Rosing, and Baris Aksanli. Opelrul: Optimally weighted ensemble learner for remaining useful life prediction. In *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–8. IEEE, 2021.
- [21] Jinping Hao, Robert J Piechocki, Dritan Kaleshi, Woon Hau Chin, and Zhong Fan. Sparse malicious false data injection attacks and defense mechanisms in smart grids. *IEEE Transactions on Industrial Informatics*, 11(5):1–12, 2015.
- [22] Wu He, Ivan Ash, Mohd Anwar, Ling Li, Xiaohong Yuan, Li Xu, and Xin Tian. Improving employees' intellectual capacity for cybersecurity through evidence-based malware training. *Journal of intellectual capital*, 2019.

- [23] Harold Edwin Hurst. A suggested statistical model of some time series which occur in nature. *Nature*, 180(4584):494–494, 1957.
- [24] Oliver Kosut et al. Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures. In *2010 first IEEE international conference on smart grid communications*, pages 220–225. IEEE, 2010.
- [25] Alexey Kurakin et al. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.
- [26] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.
- [27] Marianna Lezzi et al. Cybersecurity for industry 4.0 in the current literature: A reference framework. *Computers in Industry*, 103:97–110, 2018.
- [28] Jiao Li, Yang Liu, Tao Chen, Zhen Xiao, Zhenjiang Li, and Jianping Wang. Adversarial attacks and defenses on cyber-physical systems: A survey. *IEEE Internet of Things Journal*, 7(6):5103–5115, 2020.
- [29] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, 2018.
- [30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [31] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.
- [32] Microsoft. Understanding the cybersecurity threat landscape in asia pacific. <https://news.microsoft.com/apac/features/cybersecurity-in-asia/>, 2019.
- [33] Gautam Raj Mode and Khaza Anuarul Hoque. Crafting adversarial examples for deep learning based prognostics. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 467–472. IEEE, 2020.
- [34] Mary M Moya et al. One-class classifier networks for target recognition applications. *NASA STI/Recon Technical Report N*, 93:24043, 1993.
- [35] Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Jun Zhu, and Hang Su. Boosting adversarial training with hypersphere embedding. *Advances in Neural Information Processing Systems*, 33:7779–7792, 2020.
- [36] Nicolas Papernot et al. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.
- [37] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.
- [38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [39] C-K Peng, Sergey V Buldyrev, Shlomo Havlin, Michael Simons, H Eugene Stanley, and Ary L Goldberger. Mosaic organization of dna nucleotides. *Physical review e*, 49(2):1685, 1994.
- [40] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal processing*, 99:215–249, 2014.
- [41] Aditi Raghunathan et al. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.
- [42] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.
- [43] Krishan Rajaratnam et al. Speech coding and audio preprocessing for mitigating and detecting audio adversarial examples on automatic speech recognition, 2018.
- [44] Joshua S Richman and J Randall Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6):H2039–H2049, 2000.
- [45] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. Defense methods against adversarial examples for recurrent neural networks. *arXiv preprint arXiv:1901.09963*, 2019.
- [46] Michael T Rosenstein, James J Collins, and Carlo J De Luca. A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1-2):117–134, 1993.
- [47] Everton Jose Santana et al. Detecting and mitigating adversarial examples in regression tasks: A photovoltaic power generation forecasting case study. *Information*, 12(10):394, 2021.

- [48] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE, 2008.
- [49] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999.
- [50] Daoming She and Minping Jia. A bigru method for remaining useful life prediction of machinery. *Measurement*, 167:108277, 2021.
- [51] Nilufer Tuptuk and Stephen Hailes. Security of smart manufacturing systems. *Journal of manufacturing systems*, 47:93–106, 2018.
- [52] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [53] Pascal Vincent et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.
- [54] Juijian Wang et al. Remaining useful life estimation in prognostics using deep bidirectional lstm neural network. In *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, pages 1037–1042. IEEE, 2018.
- [55] Katy Warr. *Strengthening deep neural networks: Making AI less susceptible to adversarial trickery*. O’Reilly Media, 2019.
- [56] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.
- [57] Hansong Xu, Wei Yu, David Griffith, and Nada Golmie. A survey on industrial internet of things: A cyber-physical systems perspective. *Ieee access*, 6:78238–78259, 2018.
- [58] Weizhong Yan et al. Attack detection for securing cyber physical systems. *IEEE Internet of Things Journal*, 6(5):8471–8481, 2019.
- [59] Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song. Towards mitigating audio adversarial perturbations (2018). In URL <https://openreview.net/forum>, 2018.
- [60] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. {CommanderSong}: A systematic approach for practical adversarial voice recognition. In *27th USENIX security symposium (USENIX security 18)*, pages 49–64, 2018.
- [61] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.
- [62] Jingfeng Zhang et al. Attacks which do not kill training make adversarial learning stronger. In *ICML*, pages 11278–11287. PMLR, 2020.
- [63] Yuanqiang Zhou et al. A secure control learning framework for cyber-physical systems under sensor attacks. In *2019 ACC*, pages 4280–4285. IEEE, 2019.