IEEE Open Journal of the
Industrial Electronics Society

Received 27 February 2024; revised 13 April 2024; accepted 30 April 2024. Date of publication 15 May 2024;
date of current version 31 May 2024. The review of this article was arranged by Associate Editor Hairong Dong.

Digital Object Identifier 10.1109/OJIES.2024.3401396

Adversarial Attacks and Defenses in Fault Detection and Diagnosis: A Comprehensive Benchmark on the Tennessee Eastman Process

VITALIY POZDNYAKOV [1,2], ALEKSANDR KOVALENKO [1,2], ILYA MAKAROV [1,2] (Member, IEEE), MIKHAIL DROBYSHEVSKIY [2,3,4], AND KIRILL LUKYANOV [2,3,4]

[1]AIRI, 121170 Moscow, Russia
[2]ISP RAS Research Center for Trusted Artificial Intelligence, 109004 Moscow, Russia
[3]Ivannikov Institute for System Programming of the Russian Academy of Sciences, 109004 Moscow, Russia
[4]Moscow Institute of Physics and Technology (National Research University), 141700 Moscow, Russia

CORRESPONDING AUTHOR: VITALIY POZDNYAKOV (e-mail: pozdnyakov@airi.net)

This work was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated November 2, 2021 No. 70-2021-00142.

ABSTRACT Integrating machine learning into Automated Control Systems (ACS) enhances decision-making in industrial process management. One of the limitations to the widespread adoption of these technologies in industry is the vulnerability of neural networks to adversarial attacks. This study explores the threats in deploying deep learning models for Fault Detection and Diagnosis (FDD) in ACS using the Tennessee Eastman Process dataset. By evaluating three neural networks with different architectures, we subject them to six types of adversarial attacks and explore five different defense methods. Our results highlight the strong vulnerability of models to adversarial samples and the varying effectiveness of defense strategies. We also propose a new defense strategy based on combining adversarial training and data quantization. This research contributes several insights into securing machine learning within ACS, ensuring robust FDD in industrial processes.

INDEX TERMS Adversarial attacks, automated control systems (ACS), defense methods, fault detection and diagnosis (FDD), Tennessee Eastman Process.

I. INTRODUCTION

Automated Control Systems (ACS) operate with a variety of digital and analog signals received from sensors and control mechanisms. An example is a chemical plant where a set of sensors reflect the condition of an industrial process. A common task is Fault Detection and Diagnosis (FDD), where one needs to predict and/or classify a failure based on sensors data. Such methods play a pivotal role in monitoring diverse industrial processes, ranging from chemical processes to electromechanical drive systems. The classification proposed by [1] categorizes these methods into three groups: Those based on: 1) expert knowledge; 2) mathematical models; and 3) data-driven approaches. The latter one includes various approaches in machine learning, including neural networks [2], [3], [4]. Machine learning algorithms show themselves better than traditional methods based on rules and become more widespread in the area [5]. Recent studies demonstrate the success for FDD of various neural network architectures: Multilayer Perceptrons, Recurrent Neural Networks, and Convolutional Neural Networks [6], [7].

However, another challenge appears: Modern neural networks are vulnerable to adversarial attacks [8]. The idea is that the attacker slightly changes the input data (unnoticed) such that the FDD model prediction changes to incorrect. Such attacks are modeled in the literature [9], [10], but it remains unclear if there exist defense strategies good against a wide range of attacks. In order to address this challenge we benchmark attacks and defenses on the Tennessee Eastman Process dataset [11] where the task is the FDD in a chemical process. We consider three various deep learning models—Multilayer

Perceptron (MLP), model based on Gated Recurrent Units (GRU), and Temporal Convolutional Network (TCN). We subject these models to six different types of adversarial attacks and explore five defense methods. We analyze the success of these protective measures[1]. Then, a novel protection strategy is proposed, which employs several various defense methods.

Our contributions are as follows.

1) We benchmark popular attack and defense methods on the TEP dataset which shows that existing universal defense methods greatly reduce models quality on original data.
2) To address this issue, we suggest a new defense approach based on adversarial training and data quantization and demonstrate its average effectiveness against various attacks.
3) We discuss benchmark results and conclude that autoencoders have a potential to be a universal defense methods but they need more research.

The rest of this article is organized as follows. Section II gives a review on attacks and defense approaches in the area. Section III describes methods of attacks and defenses used in our work. In Section IV, we describe experiments and discuss them in Section V. Finally, Section VI concludes this article.

II. REVIEW

The operating principles of modern machine learning methods contain vulnerabilities that can be used to carry out various attacks on models. Different methods and areas of application require different adaptations and modifications of an attack developed in one domain area when used in another. Major research into machine learning attack vectors began around 2014 [12]. Over the past five years, this area has advanced very far and many different attack options have been developed.

A. CLASSIFICATION OF ATTACKS ON MACHINE LEARNING MODELS

Attacks on machine learning models are typically categorized into several types, which vary based on the capabilities of the attacker in relation to the target and its characteristics throughout the model's lifecycle. These attack types include evasion attacks [13], [14], [15], poison attacks [16], [17], [18], and exploratory attacks [19], [20], [21]. In addition, some of these attack categories have further subgroups. Furthermore, attacks are segmented into three groups based on the level of information available about the model's architecture and access to its internal parameters and/or requests: White-box, black-box, and gray-box attacks.

In evasion attacks (often called adversarial examples) [13], [14], [15], an attacker interacts with a trained machine learning model and manipulates its behavior by perturbing input samples during testing. The term "evasion" implies that the attacker not only aims to cause the model to behave incorrectly but also seeks to evade detection by both human and automated defense mechanisms.

Poison attacks [16], [17], [18] are a complex term in the literature. Typically, it refers to injecting poisoned samples into the training dataset with the aim of distorting the training process (so-called data poison attacks). Exploratory attacks involve sending queries to the model to understand its principles of operation. Such attacks can pursue various goals: Stealing the model [22], [23], conducting membership inference attacks [19], [20], [21], and others. In this article, the main focus was on evasion attacks, as they pose the greatest threat due to not requiring an insider attacker and directly impacting the model's predictions.

In white-box attacks [24], [25], the attacker possesses complete information, enabling them to execute any operations on their instance of the deployed model (e.g., obtaining gradients, accessing output data from any layer) to construct perturbed samples. In the presence of defense mechanisms, these mechanisms are also susceptible to attack by the adversary.

Black-box attacks [26], [27], [28] assume that the attacker can only make a limited number of requests (L, where $1 \leq L < \infty$) to the deployed model. The models provide the attacker with predictions such as label class or probability, semantic map segmentation, etc.

Gray box (or semiwhite box) [29], [30] attacks represent an intermediate state between white-box and black-box scenarios. They involve the imposition of certain restrictions that provide some information about the learning model/process, albeit incomplete.

B. MOST COMMON METHODS OF ATTACK AND DEFENSE

In the literature, there are numerous articles that explore issues akin to those addressed in this study. However, these methods are either examined on different datasets or pertain to disparate domain areas, necessitating adaptation for our purposes. Notably, many initial attacks were devised for image analysis models, and not all methods have been fully tailored for the domain area under our investigation. Therefore, in this review, we also consider these aspects. Articles such as [31], [32], [33], [34], and [35] delve into the impact of attacks on various image datasets like CIFAR-10, CIFAR-100, ResNet-20, and MNIST. These articles also discuss various protective measures. Key attack methods include L-BFGS, FGSM, PGD, C&W, and DeepFool. While identifying the most prevalent defense methods can be challenging, several key approaches emerge, notably Defense-GAN and Adversarial Training. Moreover, articles such as [9], [10], [36], [37], [38], [39], and [40] discussed attacks and defense methods on datasets like TEP and/or similar domain areas such as CARLA, Electra, SWaT, BATADAL, and WADI. Many attack and defense methods in these articles share operational principles with those used in computer vision.

In white-box adversarial attacks, access to gradients is a primary tool. Attackers exploit gradients by calculating the gradient of the loss function concerning the input. Then, they

[1]The source code to reproduce our results is available at https://github.com/AIRI-Institute/fdd-defense.

perturb the input in the direction of the gradient to maximize the loss. The mathematical details of the attack algorithms that will be used in this article are outlined in the next section (Section III).

The Fast Gradient Sign Method (FGSM), proposed by [41], generates adversarial examples with a single gradient step. It updates the input based on the direction of increasing loss, using a small multiple of the sign of the gradient. While FGSM is fast, its success rate for adversarial examples is low.

To improve the success rate of FGSM, [42] introduced the Projected Gradient Descent (PGD) method. Unlike FGSM, PGD takes multiple smaller steps in the gradient direction and clips the result by a specified value. Although PGD is more effective than FGSM in finding adversarial examples closer to the model's decision boundary, it is computationally more expensive due to requiring multiple iterations.

For untargeted attacks, [43] proposed the DeepFool method optimized for the L_2 distance metric. It assumes the linearity of the decision boundary in neural networks and finds the minimum adversarial perturbation needed to fool the classifier. DeepFool iteratively identifies the direction that maximally changes the current prediction of the neural network and takes a small step in that direction until finding a true adversarial example.

A more sophisticated white-box attack, the C&W attack by [44], was applicable under various distance metrics: L_0, L_2, and L_∞. This attack optimizes a loss function considering the distance between the original input and the adversarial example, along with the classifier's prediction confidence. The optimization includes a constraint on the perturbation size, making the resulting adversarial example more realistic and challenging to detect.

Adversarial training [31], [45], [46] is a widely used defense technique aimed at making neural networks more resilient to adversarial attacks. Instead of relying solely on traditional training data, adversarial training incorporates examples with adversarial biases into the training process. Adversarial training has been shown to be effective in improving the robustness of neural networks to various types of adversarial attacks, including both white-box and black-box attacks. Despite some problems, adversarial training remains one of the most effective methods for protecting against adversarial attacks and is widely used in practice to improve the security and reliability of neural networks.

C. INTERACTIONS AMONG DEFENSE METHODS

While the literature offers many different methods for defending machine learning models, there are few studies that explore building models combining multiple defense methods. In the paper [47], the authors examine the possibility of combining the most popular defense methods against evasion and poisoning attacks. The research concludes that many methods, at the level of algorithmic ideas, are incompatible, demonstrating this through practical examples. Therefore, constructing models that combine defense methods is a complex and underexplored task.

D. SUMMARY OF THE REVIEW

Since the main objective of the article is to create a benchmark, we pay particular attention to the most common methods described in the literature. This research focuses on analyzing vulnerabilities and implementing protection strategies within ACS. The following section elaborates on the mathematical aspects of the methods employed in this research.

III. METHODS

A. FAULT DIAGNOSIS METHODS

Fault detection and diagnosis (FDD) methods, are widely used in monitoring industrial processes, such as chemical processes [48] and electromechanical drive systems [49]. The authors of [1] divided FDD methods into three groups: 1) data-driven; 2) model-based; and 3) knowledge-based approaches. In our work, we investigate the properties of data-driven methods.

Data-driven FDD problem is formulated as follows. Let there be a sequence of observations $X_1, \ldots, X_n$, where $X_t \in R^d$ are the values of sensors at time t. Thus, $X_1, \ldots, X_n$ form a multivariate time series. Also, let there be a sequence of labels $y_1, \ldots, y_n$ where $y_t \in \{0, 1\}^m$ defines the type of fault at time t. If $\arg\max(y_t) = 0$, the process is in the normal state, otherwise $\arg\max(y_t)$ determines the fault number. Then for a sliding window of width k, we need to find such a function $f : R^{d\times k} \to [0, 1]^m$ that

$$f = \arg\min_f \frac{1}{n-k} \sum_{t=k}^{n} l(y_t, f(X_{t-k+1}, \ldots, X_t))$$

where l is some loss function, most commonly cross-entropy, also known as Log Loss. The function f can be found using machine learning methods.

In recent years, many deep learning methods based on different neural network architectures were proposed to solve FDD problem. The simplest one is MLP that was applied to FDD in [6], [50], [51], and [52]. Multivariate time series is converted to a vector of concatenated observation, and then processed by MLP to predict the process state. TCN is another popular architecture for FDD [53], [54], [55]. TCN is a modification of a 1-D convolutional network with causal and dilated convolutions [56] that helps to process sequential data with long-term dependencies. In addition, GRU is a type of recurrent neural networks that shows SOTA results of FDD on many datasets including Tennessee Eastman Process [7], [57].

B. ADVERSARIAL ATTACKS

During the attack, an adversarial sample X'_t is created such that: $f(X'_t) \neq f(X_t)$, where $X'_t = X_t + \mathcal{N}$ and $\mathcal{N} \in R^{d\times k}$ is a perturbation matrix. Strength of an attacks is defined by the maximal shift ϵ as follows: $\|X_t - X'_t\|_\infty \leq \epsilon$.

When choosing types of attacks, we proceeded from the assumption that the attacker has access to either only input

and output data or all information about the data and model architecture. Two black-box (Random noise, FGSM distillation) and four white-box attacks (FGSM, PGD, DeepFool, Carlini, and Wagner) were implemented.

1) RANDOM NOISE

Random noise is the simplest black-box attack based on adding random values to the input data

$$x' = x + \epsilon z$$

where ϵ limits the magnitude of noise values and z is distributed according to Bernoulli's principle with parameter $p = 0.5$ on the sample space of elementary events $\{-1, 1\}$.

2) FAST GRADIENT SIGN METHOD (FGSM)

FGSM [41] is a white-box attack based on the gradient of the loss function calculated for the input data. The signs of obtained gradient vector indicate the direction in which the input data should be changed to increase the probability of model error. The attack consists of shifting each value of the data by a step of size ϵ, with a sign corresponding to the gradient

$$x' = x + \epsilon \,\mathrm{sign}[\nabla l(f(x), y)].$$

3) FGSM DISTILLATION

Distillation can be used to create a black-box adversarial attack as proposed in [58]. Based on the input and output data of the model, a neural network classifier with an arbitrary architecture can be trained. Adversarial samples are obtained by attacking the resulting model by any white-box attack. In our study, we used MLP architecture and FGSM attack.

4) PROJECTED GRADIENT DESCENT (PGD)

PGD [42] is an iterative modification of the FGSM white-box attack method. The main difference is that the data shift is done in several steps. After each step, the gradient signs are recalculated

$$x'_{i+1} = \mathrm{Clip}_{\epsilon}\{x'_i + \alpha \,\mathrm{sign}[\nabla \mathrm{l(f(x), y)}]\}$$

where x'_i denotes the changed input data since the previous iteration, Clip{} limits the resulting data shift to no more than ϵ, and α denotes the shift step size at each iteration.

5) DEEPFOOL

DeepFool [43] is a white-box attack which minimizes the difference between the elements of the output vector $f(x)$ that correspond to the correct and incorrect fault type. Among all possible incorrect types, the closest in absolute value of the difference is selected. Minimization occurs in several steps, each defined as follows:

$$x'_{i+1} = x_i + \frac{|D(x_i)|}{||\nabla D(x_i)||_1} \nabla D(x_i)$$

where $D(x) = f(x)_{\mathrm{false}} - f(x)_{\mathrm{true}}$. $f(x)_{\mathrm{false}}$ is the value of the output vector corresponding to the nearest incorrect fault type, which is selected independently at each step. After each iteration, the total adversarial vector x'_{i+1} is limited by ϵ value.

6) CARLINI AND WAGNER (C&W)

C&W [44] is a white-box attack that minimizes the sum of the shift value over the distance metric D and the value of some auxiliary function g. Function g takes negative values in case of incorrect classification. This optimization problem can be represented as

$$\min_{\eta} D(x, x + \eta) + g(x + \eta)$$

where $g(x) = \mathrm{ReLU}(\arg\max(y) - \arg\max(f(x)))$ and D is Chebyshev distance. Minimization is performed by the stochastic gradient descent method or its analogues. For comparison with other attack methods, we constrain η according to the selected ϵ value.

C. DEFENSE METHODS

Another goal of the study was to find out how defense methods behave under attacks with different strengths and for different neural network architectures. The five most popular strategies were implemented: Adversarial training, Autoencoder, Quantization, Regularization, and Distillation. We also proposed to protect models by combination of defense methods.

1) ADVERSARIAL TRAINING

Adversarial training method [41] consists of adding adversarial samples to the training set. The training loss function is given as follows:

$$L = l(f(x), y) + \lambda l(f(x'), y)$$

where x' is adversarial sample and λ is adversarial training coefficient.

2) DEFENSIVE AUTOENCODER

Autoencoder can be used to reconstruct attacked data as proposed in [59]. During its training, the following loss function is minimized:

$$L = ||x_{AE} - x||_1$$

where $x_{AE} = \mathrm{autoencoder}(x + \varepsilon)$ is a reconstructed data and ε is added noise.

3) DATA QUANTIZATION

Quantization is a preprocessing method that converts continuous values into a set of discrete values on a uniform grid [60]. This approach reduces the quality of the input data but can neutralize the impact of adversarial attacks. The fault diagnosis model must be retrained on quantized data.

4) GRADIENT REGULARIZATION

The fault diagnosis model can be protected by training using gradient regularization [61] of the loss function over the input

data

$$L = l(f(x), y) + \lambda \left(\frac{1}{h^2 n} ||f(z) - f(x)||_2^2 \right)$$

where

$$z = x + h \frac{\nabla l(f(x), y)}{||\nabla l(f(x), y)||_2}.$$

h is a quantization step and λ is a regularization coefficient.

5) DEFENSIVE DISTILLATION

Distillation defense method [62] refers to the process of creating a copy of the original neural network model that is more resistant to adversarial attacks. The original neural network is called the teacher, and the new neural network is called the student. When teaching a student, so-called smooth labels are used, which are obtained using the activation function softmax(x, T) on the last layer of the teacher

$$\text{softmax}(x, T)_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$

where T is a temperature constant. At $T = 0$, the function converges to a maximum and at $T \to \infty$, the function converges to a uniform distribution.

6) ADVERSARIAL TRAINING ON QUANTIZED DATA

In recent years, a lot of research has been carried out to develop new defense methods against adversarial attacks. New ideas emerge that are superior to previous approaches in certain conditions. However, there is still no ideal defense method capable of protecting against all types of threats. The vulnerability of protected neural networks is reduced only under certain types of adversarial attacks; in other cases, the accuracy of the models drops noticeably.

In this article, we propose to use a combination of adversarial training and data quantization. As was shown in [60], quantization allows to clean the input from adversarial perturbation due to the grid alignment of discrete values. However, the size of the grid (quantization frequency), has an important role in this type of protection. If the grid is too wide, it reduces the quality of fault diagnosis, if the grid is too narrow, only a fraction of the data can be effectively recovered. On the other hand, adversarial training provides high model robustness, but reduces the quality of diagnosis. This happens because during training, the data contains many adversarial examples that degrade the model's ability to generalize important dependencies in the data that help diagnose faults. Thus, at high values of ϵ in adversarial training, the quality of the model drops significantly, otherwise it does not provide a sufficient level of protection.

We propose to use adversarial training on the data after quantization. Thus, during training, we attack the data with an adversarial attack such as FGSM. We then quantize this data and feed it into the input of the model as a training set. Quantization allows to reduce the strength of the attack, which in turn allows the model to generalize better during adversarial training. As a result, quantization helps the model to achieve better quality in adversarial training.

An additional advantage of this approach is that it does not require a separate model, as is the case with the distillation method or the autoencoder. It is also quite efficient in terms of computational time and memory, since quantization takes place in linear time and requires no additional memory, while adversarial training has the same complexity as training a model on the original data and also requires no additional memory.

D. DATASET

The Tennessee Eastman Process is a very popular dataset for benchmarking fault detection and diagnosis methods. It describes the operation of a chemical production line, where the process smoothly transitions from a normal state to a faulty one. In our study, we used a version of the TEP extended by Reinartz et al. [48] that contained significantly more sensor data than the original (5.2 GB versus 58 MB). This version includes 100 simulation runs for each of the 28 fault types. Each run consists of 52 sensor values for 2000 timestamps, and thus the input samples are in the form of matrices $X^{k \times 52}$, where k is the sliding window size. All data in our experiments were standardized by removing the mean and scaling to unit variance.

IV. EXPERIMENTS

In our study, we wanted to find out how adversarial attacks affect FDD models based on neural networks with different architectures and what defense methods can be used. To analyze the impact of adversarial attacks on fault diagnosis models, the accuracy metric was chosen. This metric well reflects changes in the quality of models when the data is attacked.

The description of our experiments is divided into four subsections. The FDD models subsection describes the training process of neural networks with different architectures. The next subsection shows how the accuracy of the models changes under different types of attack. Further, various methods for protecting models and their properties are shown. Finally, on the basis of the results of experiments, we also proposed and evaluated an approach consisting of a combination of two defense methods. All final results can be found in Fig. 7 and Tables 6–8.

A. FDD MODELS

For our experiments, we used three models of neural networks with different architectures. To make the models differ from each other more, they contain different numbers of parameters and were trained for different numbers of epochs. The first model is a multilayer perceptron (MLP) consisting of two linear layers and containing 3 452 949 parameters. The second one is based on gated recurrent units (GRU) and containing 204 565 parameters. We also used TCN with 151 935 parameters. Data were standardized with a standard deviation of 1. Sliding window size was 32, which is a compromise between the accuracy of the models and the duration of the

TABLE 1 Accuracy of Unprotected Models on Normal Data

Model	Accuracy
MLP	0.8873 ± 0.0002
GRU	0.9067 ± 0.0041
TCN	0.8985 ± 0.0097

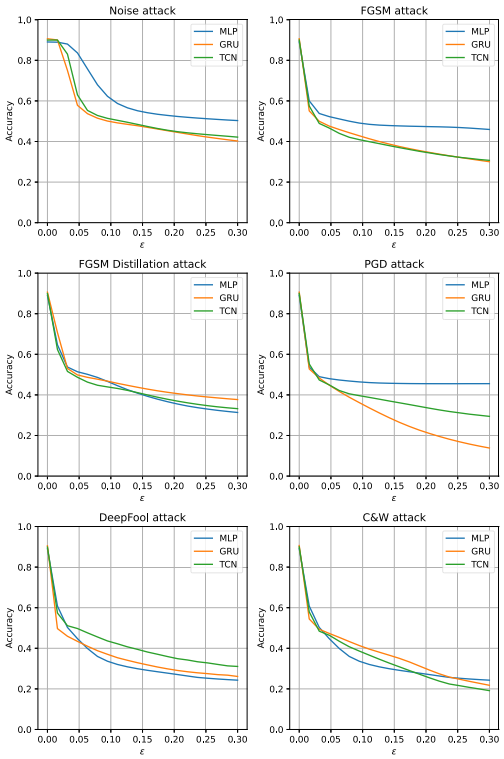

FIGURE 1. Accuracy drop of unprotected models under six different types of attacks depending on the strength of an attack ϵ.

experiments. All models were trained on the TEP dataset for 20, 5, and 10, epochs for MLP, GRU, and TCN, respectively. The accuracy metrics for fault diagnosis on nonattacked data are presented in Table 1. Combinations of the number of parameters and training epochs are selected on the validation set.

The selected neural network architectures showed similar accuracy and can effectively solve the fault diagnosis task.

B. ATTACKS ON FDD MODELS

At the next stage, unprotected models were attacked by six types of attacks with different ϵ. For ϵ values, 20 points were selected in the range from 0 to 0.3 with a step of 0.015. We consider this range to be reasonable given that the data is scaled to a unit variance and the attack should not be detected by both human and automated defense mechanisms. Fig. 1 shows how the model's accuracy degrades depending on the type and strength of the attack. It decreases significantly with small shifts in the attacked data for ϵ values less than 0.05.

To cause potential harm, an attacker does not always need to have access to model architectures and use white-box attacks. Experiments have shown that to create a strong adversarial

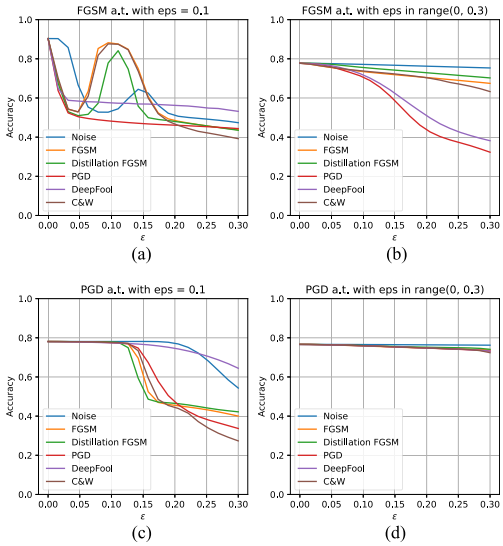

FIGURE 2. Accuracy of the TCN model protected by adversarial training with different settings: (a) training on FGSM samples with fixed $\epsilon = 0.1$; (b) training on FGSM samples with set of ϵ values from the range (0, 0.3); (c) training on PGD samples with fixed $\epsilon = 0.1$; (d) training on PGD samples with set of ϵ values from the range (0, 0.3).

attack, it is enough to have access to the input and output data of the FDD system. This data can be used to train an arbitrary neural network architecture on the basis of which adversarial samples will be created. The distillation FGSM attack showed a similar effect on the accuracy of models as white-box attacks in our study. This type of black-box attacks seems to be the easiest to carry out and potentially the most dangerous.

C. PROTECTION OF FDD MODELS

All three neural network architectures have proven to be highly vulnerable to adversarial attacks and require protection methods. The defense methods studied in our research have many variations and parameters for selecting. It is not possible to conduct experiments for all combinations of settings and models in an adequate period of time. Therefore, we took only the TCN model, which has the fastest inference, to select more optimal settings for defense methods adjustment. Experiments conducted for each type of protection are described in following subsections. After setting up, the defense methods were applied to all FDD models and the final results are presented in Fig. 7 and Tables 6–8.

1) ADVERSARIAL TRAINING

In our study we used equal amounts of normal and attacked data for adversarial training method. Experiments have shown a strong dependence of the model robustness on the set of adversarial samples during the training process. As an example the model trained on attacked data with ϵ value 0.1 is not protected from attacks with ϵ values 0.05 and 0.2. Fig. 2 shows changes in the TCN model's accuracy after adversarial training with different options. The first one is the training with FGSM adversarial samples and fixed ϵ value equal to

TABLE 2 **Accuracy of the TCN Model Protected by Adversarial Training on Normal Data**

Type of adversarial training	Accuracy
None	0.90
FGSM with $\epsilon = 0.1$	0.90
FGSM with ϵ in range(0, 0.3)	0.78
PGD with $\epsilon = 0.1$	0.78
PGD with ϵ in range(0, 0.3)	0.77

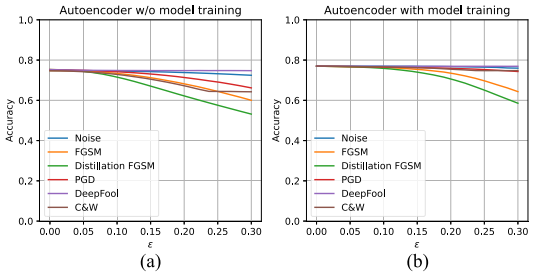

FIGURE 3. **Accuracy of the TCN model protected by autoencoder: (a) model was trained on the original data; (b) model was trained on the data obtained at the output of autoencoder.**

TABLE 3 **Accuracy of the TCN Model Protected by Autoencoder on Normal Data**

Type of autoencoder defense	Accuracy
None	0.90
Autoencoder w/o model training	0.75
Autoencoder with model training	0.77

0.1. Then, the number of ϵ values was expanded to the set with range from 0.015 to 0.3 (ϵ values were randomly selected from the set for every data sample). The same measurements were made for training with PGD adversarial samples.

Adding more different perturbed data to the training process increases the average robustness of the model to adversarial attacks. But the quality with normal data decreases. Table 2 shows the accuracy of the TCN model on nonattacked data before and after adversarial training. Training with PGD adversarial samples showed better robustness from all attack types but worse quality in nonattacked mode. We used this setting for the final comparison of all defense methods.

2) DEFENSIVE AUTOENCODERS

During the experiments, we trained a simple autoencoder with linear layers in the encoder and decoder parts. There are two options for using it in conjunction with the models. The model can be trained on the original dataset data or on autoencoder output data. Both approaches are shown in Fig. 3 using the TCN model as an example. Experiments have shown that when using an autoencoder, a model trained on its output shows better quality and robustness to adversarial attacks. This setting was chosen for the final comparison for all models.

Accuracy metrics on nonattacked data can be seen in Table 3. It is significantly lower than on unprotected model, but there is an opportunity to experiment with advanced autoencoder architectures in further research.

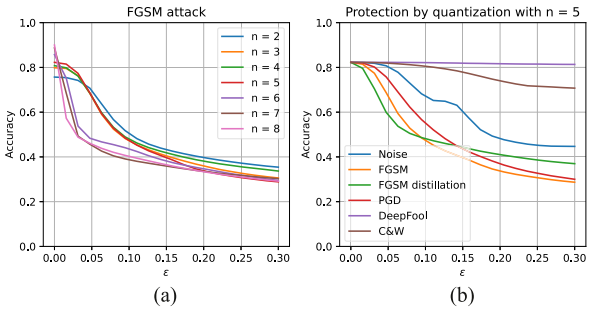

FIGURE 4. **Accuracy of the TCN model protected by quantization: (a) model is under FGSM attack and *n* indicates the number of discrete values during the quantization process (2^n); (b) model is protected by quantization with $n = 5$ under six types of attacks.**

TABLE 4 **Accuracy of the TCN Model Protected by Quantization With Different Parameter *n* on Normal Data**

Number of discrete values $= 2^n$	Accuracy
None	0.90
n = 2	0.76
n = 3	0.80
n = 4	0.81
n = 5	0.82
n = 6	0.86
n = 7	0.87
n = 8	0.90

3) DATA QUANTIZATION

Quantization converts continuous input data into a set of discrete values. To select the number of discrete values in the set, we used different values n for powers of two (from 2^2 to 2^8). The left part of Fig. 4 shows the accuracy of the TCN model protected by quantization method with different sets of discrete values. The attacks where made by FGSM adversarial samples. The remaining types of attacks on the model protected by quantization with $n = 5$ are presented on the right side of the picture.

Table 4 shows the accuracy of the TCN model protected by quantization with different numbers of discrete values on nonattacked data.

For the final comparison, the setting with $n = 5$ was chosen as a tradeoff between the model's robustness to attacks and the accuracy on normal data.

4) GRADIENT REGULARIZATION

The parameters for tuning the regularization method did not show a significant impact on the effectiveness of the protection. Fig. 5(a) shows the change in the accuracy of the protected TCN model after all types of attacks. The quantization step parameter h and regularization coefficient λ were equal to 0.001 and 1, respectively. Regularization turned out to be useful just for small ϵ values. However, it well improves robustness against random noise.

5) DEFENSIVE DISTILLATION

Distillation is a gradient masking technique that protects models against gradient-based adversarial attacks. Changing the

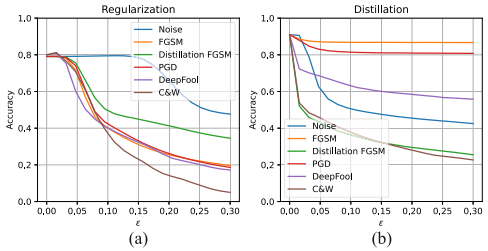

FIGURE 5. Accuracy of the TCN model protected by: (a) regularization defense method; (b) distillation defense method.

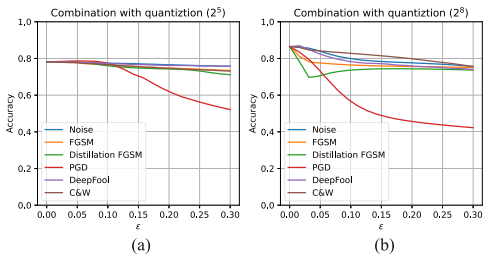

FIGURE 6. Accuracy of the TCN model protected by combination of FGSM adversarial training ($\epsilon = 0.1$) and: (a) quantization having 2^5 discrete values; (b) quantization having 2^8 discrete values.

temperature constant parameter T does not significantly affect the effectiveness against other types of attacks. Fig. 5(b) shows the accuracy of the TCN model protected by distillation defense method with parameter $T = 100$. The results confirm good protection against gradient-based FGSM and PGD adversarial attacks. However, other threats remain relevant when using this protection method.

6) ADVERSARIAL TRAINING ON QUANTIZED DATA

The experimental results showed that various defense methods can be effective against some types of attack and not against the others. This fact suggests the idea of using several defense approaches together. In our study, we used a combination of adversarial training and quantization defense methods.

For the adversarial training setting we chose attack with FGSM samples and $\epsilon = 0.1$ [Fig. 2(a)]. We combined it with the quantization having 2^5 discrete values [Fig. 4(a)]. The results of this combination significantly exceed the effectiveness of these methods separately [Fig. 6(a)]. Moreover, we tried to change the quantization defense setting by increasing the number of discrete values to 2^8. Its combination with FGSM adversarial traning are shown in Fig. 6(b).

Quantization defense method having 2^8 discrete values is more vulnerable to adversarial attacks than the one having 2^5 values. But its combination with adversarial training gives partially better results, especially on normal, nonattacked data (Table 5).

V. DISCUSSION

Our experiments confirmed the vulnerability of fault diagnosis models based on different neural networks to adversarial attacks. We implemented six types of attacks and all of them

TABLE 5 Accuracy of the TCN Model Protected by Combination of Adversarial Training and Quantization Defense Method on Normal Data

Type of combination	Accuracy
None	0.90
Adv. tr. and quantization with $n = 5$	0.79
Adv. tr. and quantization with $n = 8$	0.89

lead to a significant decrease in accuracy of FDD methods. However, a good defense method should be effective against any type of adversarial attack. At the same time, the accuracy of defended models should not drop significantly on normal, nonattacked data. The ϵ parameter, which limits the maximum shift in the attacked data, is common to all types of attacks. The choice of ϵ value range when creating protection for models depends on many factors (such as additional systems for detecting adversarial attacks) and is the subject of discussion. In our work, we investigated five types of defense methods against adversarial attacks with ϵ values in the range (0, 0.3). We also proposed a combination of adversarial training and quantization defense methods.

Adversarial training with PGD samples and defense by autoencoder can be considered as universal methods against adversarial attacks over a wide range of ϵ values. The disadvantage of these approaches is a significant decrease in accuracy on normal nonattacked data. Adversarial training can be done against the attack with a specific ϵ value without losing accuracy on normal data but will be ineffective for attacks with other ϵ values. Adding more variety to adversarial samples degrades the overall accuracy of the model.

The accuracy of the model protected by autoencoder on nonattacked data has noticeably decreased, but was stable after most types of attacks. This approach seems to have great potential and requires further research with different autoencoder architectures. In addition, the vulnerability of the autoencoders themselves should be studied. The disadvantage of this method is the need for additional computing resources.

Other methods such as quantization, regularization, and distillation have shown high protection against some types of attacks and poor results against the others. To address the limitations of individual defense methods, we explored the possibility of combining them. In our study, we combined FGSM adversarial training and quantization defense method. This approach provides good protection against most types of attacks (except for PGD adversarial examples with large ϵ values) with small losses in quality. It is computationally efficient and does not require additional memory. Other combinations of various defense methods can be explored in further research.

VI. CONCLUSION

This study confirmed that adversarial attacks can greatly reduce the quality of FDD models. Such attacks can be quite feasible if attackers have access to the data exchange system. Therefore it is important to know the robustness of models used in real systems to adversarial samples. There are many

TABLE 6 **Accuracy of Protected and Unprotected MLP Model After Adversarial Attacks With Different ϵ Values**

ϵ		0.015	0.03	0.06	0.09	0.12	0.15	0.18	0.21	0.24	0.27	0.30
Noise attack	Unprotected	**0.89**	**0.88**	0.76	0.62	0.57	0.54	0.53	0.52	0.52	0.51	0.50
	Adversarial training	0.77	0.77	0.77	0.77	0.77	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**
	Autoencoder	0.76	0.76	0.76	0.75	0.75	0.75	0.75	0.75	0.74	0.74	**0.73**
	Quantization	0.82	0.82	0.81	0.78	0.77	0.73	0.69	0.62	0.57	0.54	0.51
	Regularization	0.83	0.83	**0.83**	**0.83**	**0.82**	**0.82**	**0.81**	**0.79**	**0.77**	**0.75**	0.72
	Distillation	**0.90**	**0.89**	0.80	0.67	0.60	0.57	0.56	0.54	0.52	0.51	0.50
	Combination (ours)	0.86	0.86	**0.85**	**0.83**	**0.81**	**0.77**	0.75	0.72	0.68	0.65	0.62
FGSM attack	Unprotected	0.60	0.54	0.51	0.49	0.48	0.48	0.48	0.47	0.47	0.47	0.46
	Adversarial training	0.77	0.77	0.77	0.77	0.76	0.76	0.76	0.75	0.75	0.75	0.74
	Autoencoder	0.76	0.75	0.74	0.73	0.70	0.66	0.64	0.61	0.57	0.55	0.53
	Quantization	0.82	0.80	0.63	0.52	0.46	0.37	0.32	0.25	0.21	0.20	0.19
	Regularization	0.79	0.64	0.49	0.45	0.42	0.40	0.39	0.38	0.36	0.35	0.34
	Distillation	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**
	Combination (ours)	**0.83**	**0.82**	**0.83**	**0.84**	**0.84**	**0.84**	**0.84**	**0.84**	**0.84**	**0.84**	**0.84**
FGSM distillation attack	Unprotected	0.65	0.54	0.50	0.47	0.43	0.39	0.38	0.36	0.34	0.32	0.31
	Adversarial training	0.77	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.76**
	Autoencoder	0.75	**0.75**	**0.74**	**0.72**	**0.69**	**0.65**	**0.63**	**0.59**	**0.56**	**0.53**	0.51
	Quantization	**0.81**	0.74	0.57	0.51	0.49	0.49	0.48	0.48	0.47	0.45	0.43
	Regularization	**0.81**	0.74	0.56	0.49	0.47	0.45	0.44	0.42	0.41	0.40	0.39
	Distillation	0.57	0.51	0.46	0.44	0.41	0.38	0.37	0.35	0.34	0.32	0.31
	Combination (ours)	**0.83**	0.66	0.55	0.54	0.54	0.54	0.54	0.53	0.53	**0.53**	**0.52**
PGD attack	Unprotected	0.54	0.49	0.47	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46
	Adversarial training	0.77	0.77	**0.77**	**0.77**	**0.76**	**0.76**	**0.76**	**0.75**	**0.75**	**0.74**	**0.74**
	Autoencoder	0.76	0.75	0.75	0.73	0.71	0.68	0.67	0.64	0.61	0.59	0.56
	Quantization	0.82	**0.80**	0.65	0.53	0.74	0.38	0.34	0.26	0.22	0.20	0.20
	Regularization	0.78	0.57	0.45	0.41	0.38	0.36	0.35	0.34	0.33	0.32	0.31
	Distillation	**0.89**	**0.89**	**0.89**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**
	Combination (ours)	**0.83**	0.76	0.54	0.50	0.49	0.48	0.48	0.47	0.46	0.45	0.44
DeepFool attack	Unprotected	0.61	0.50	0.40	0.34	0.31	0.29	0.28	0.27	0.26	0.25	0.24
	Adversarial training	0.77	0.77	0.77	0.76	0.76	0.75	0.75	0.74	0.74	0.73	0.72
	Autoencoder	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
	Quantization	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.81**	**0.81**	**0.8**	**0.79**	**0.78**	**0.77**
	Regularization	0.77	0.51	0.45	0.39	0.35	0.32	0.3	0.26	0.24	0.22	0.2
	Distillation	0.73	0.73	0.71	0.69	0.68	0.68	0.68	0.67	0.67	0.67	0.67
	Combination (ours)	**0.86**	**0.84**	**0.84**	**0.83**	**0.83**	**0.82**	**0.82**	**0.81**	**0.80**	**0.80**	**0.78**
C&W attack	Unprotected	0.61	0.50	0.40	0.34	0.31	0.29	0.28	0.27	0.26	0.25	0.24
	Adversarial training	0.77	0.77	0.77	0.77	**0.76**	**0.76**	**0.75**	**0.75**	**0.74**	**0.72**	**0.71**
	Autoencoder	0.75	0.75	0.74	0.71	0.67	0.63	0.61	0.56	0.53	0.53	0.53
	Quantization	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**
	Regularization	0.8	0.73	0.56	0.43	0.37	0.33	0.31	0.26	0.23	0.21	0.18
	Distillation	0.63	0.57	0.49	0.41	0.36	0.32	0.3	0.27	0.24	0.24	0.24
	Combination (ours)	**0.86**	**0.84**	**0.79**	**0.78**	0.75	0.71	0.69	0.64	0.58	0.53	0.48

TABLE 7 **Accuracy of Protected and Unprotected GRU Model After Adversarial Attacks With Different ϵ Values**

ϵ		0.015	0.03	0.06	0.09	0.12	0.15	0.18	0.21	0.24	0.27	0.30
Noise attack	Unprotected	**0.90**	0.75	0.54	0.50	0.48	0.47	0.46	0.44	0.43	0.41	0.40
	Adversarial training	0.76	0.76	0.76	0.76	0.76	0.76	0.76	**0.76**	**0.76**	**0.76**	**0.76**
	Autoencoder	0.76	0.76	0.76	0.76	0.75	0.75	0.75	0.75	0.74	0.74	0.73
	Quantization	0.81	0.81	**0.80**	0.76	0.72	0.68	0.65	0.59	0.54	0.51	0.50
	Regularization	0.79	0.79	0.79	**0.79**	**0.79**	**0.79**	**0.78**	0.73	0.62	0.53	0.49
	Distillation	0.86	**0.84**	0.63	0.52	0.50	0.48	0.48	0.47	0.45	0.44	0.43
	Combination (ours)	**0.87**	**0.86**	**0.82**	**0.79**	**0.79**	**0.78**	**0.78**	**0.78**	**0.77**	**0.77**	**0.77**
FGSM attack	Unprotected	0.55	0.50	0.46	0.43	0.40	0.38	0.36	0.35	0.33	0.31	0.30
	Adversarial training	0.76	0.76	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74
	Autoencoder	0.76	0.76	0.75	0.75	0.74	0.73	0.72	0.70	0.67	0.63	0.59
	Quantization	**0.81**	**0.80**	0.73	0.62	0.54	0.50	0.48	0.46	0.44	0.41	0.39
	Regularization	0.79	0.74	0.45	0.38	0.34	0.29	0.27	0.23	0.21	0.19	0.17
	Distillation	**0.85**	**0.84**	**0.84**	**0.83**	**0.83**	**0.83**	**0.83**	**0.83**	**0.83**	**0.83**	**0.83**
	Combination (ours)	**0.81**	0.79	**0.79**	**0.79**	**0.79**	**0.79**	**0.79**	**0.79**	**0.78**	**0.78**	**0.77**
FGSM distillation attack	Unprotected	0.71	0.53	0.49	0.47	0.45	0.43	0.42	0.41	0.39	0.39	0.28
	Adversarial training	0.76	**0.76**	**0.76**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**
	Autoencoder	0.76	0.75	**0.74**	0.73	0.70	0.67	0.65	0.62	0.58	0.55	0.52
	Quantization	**0.80**	0.74	0.58	0.53	0.51	0.50	0.50	0.49	0.48	0.46	0.45
	Regularization	**0.79**	**0.79**	0.70	0.53	0.48	0.46	0.45	0.44	0.43	0.41	0.40
	Distillation	0.65	0.51	0.48	0.46	0.43	0.41	0.40	0.38	0.37	0.35	0.34
	Combination (ours)	0.77	0.69	**0.74**	**0.74**	**0.74**	**0.73**	**0.73**	**0.72**	**0.71**	**0.70**	**0.68**
PGD attack	Unprotected	0.53	0.48	0.42	0.36	0.31	0.26	0.24	0.21	0.18	0.16	0.14
	Adversarial training	0.76	0.76	0.75	0.75	**0.75**	**0.75**	0.74	0.74	0.73	0.73	0.72
	Autoencoder	0.76	0.76	0.76	0.75	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.74**
	Quantization	0.81	**0.81**	**0.80**	**0.78**	**0.75**	0.71	0.68	0.63	0.58	0.54	0.51
	Regularization	0.79	0.72	0.43	0.36	0.31	0.26	0.23	0.19	0.17	0.14	0.12
	Distillation	**0.85**	**0.84**	**0.83**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**
	Combination (ours)	**0.82**	0.75	0.62	0.52	0.47	0.45	0.44	0.43	0.42	0.40	0.39
DeepFool attack	Unprotected	0.50	0.46	0.41	0.37	0.34	0.32	0.31	0.29	0.28	0.27	0.26
	Adversarial training	0.76	0.76	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.73	0.73
	Autoencoder	0.76	0.76	0.76	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**
	Quantization	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**
	Regularization	0.79	0.76	0.45	0.39	0.34	0.31	0.29	0.25	0.21	0.19	0.17
	Distillation	0.76	0.63	0.54	0.53	0.51	0.50	0.49	0.48	0.47	0.46	0.45
	Combination (ours)	**0.86**	**0.85**	**0.79**	0.74	0.72	0.71	0.70	0.69	0.68	0.68	0.67
C&W attack	Unprotected	0.54	0.49	0.45	0.41	0.38	0.35	0.33	0.29	0.26	0.24	0.22
	Adversarial training	0.76	0.76	0.75	0.75	0.75	0.75	0.75	0.74	0.74	**0.74**	**0.73**
	Autoencoder	0.76	0.76	0.75	0.75	0.74	0.72	0.72	0.69	0.67	0.67	0.67
	Quantization	**0.81**	**0.81**	**0.81**	**0.80**	**0.79**	**0.78**	**0.78**	**0.76**	**0.75**	**0.75**	**0.74**
	Regularization	0.78	0.77	0.45	0.39	0.34	0.29	0.27	0.23	0.19	0.17	0.15
	Distillation	0.56	0.50	0.47	0.44	0.42	0.39	0.38	0.36	0.34	0.33	0.32
	Combination (ours)	**0.85**	**0.85**	**0.84**	**0.83**	**0.82**	**0.81**	**0.81**	**0.79**	**0.77**	**0.74**	0.72

types of attacks and quite a few universal defense methods capable of protecting against all of them simultaneously. Also universal defense methods significantly reduce the accuracy of the models on normal nonattacked data. Here, it seems that autoencoders can be improved in further research through the use of more advanced architectures. Many of defense methods protect against certain types of adversarial attacks with a good efficiency. Such methods can be combined into one more powerful defense system, like our experiment with adversarial training and quantization.

APPENDIX A. COMPARISON OF ALL METHODS

In this section, we present the final comparison of all combinations of models, attacks, and defenses (Tables 6, 7, and 8 and Fig. 7).

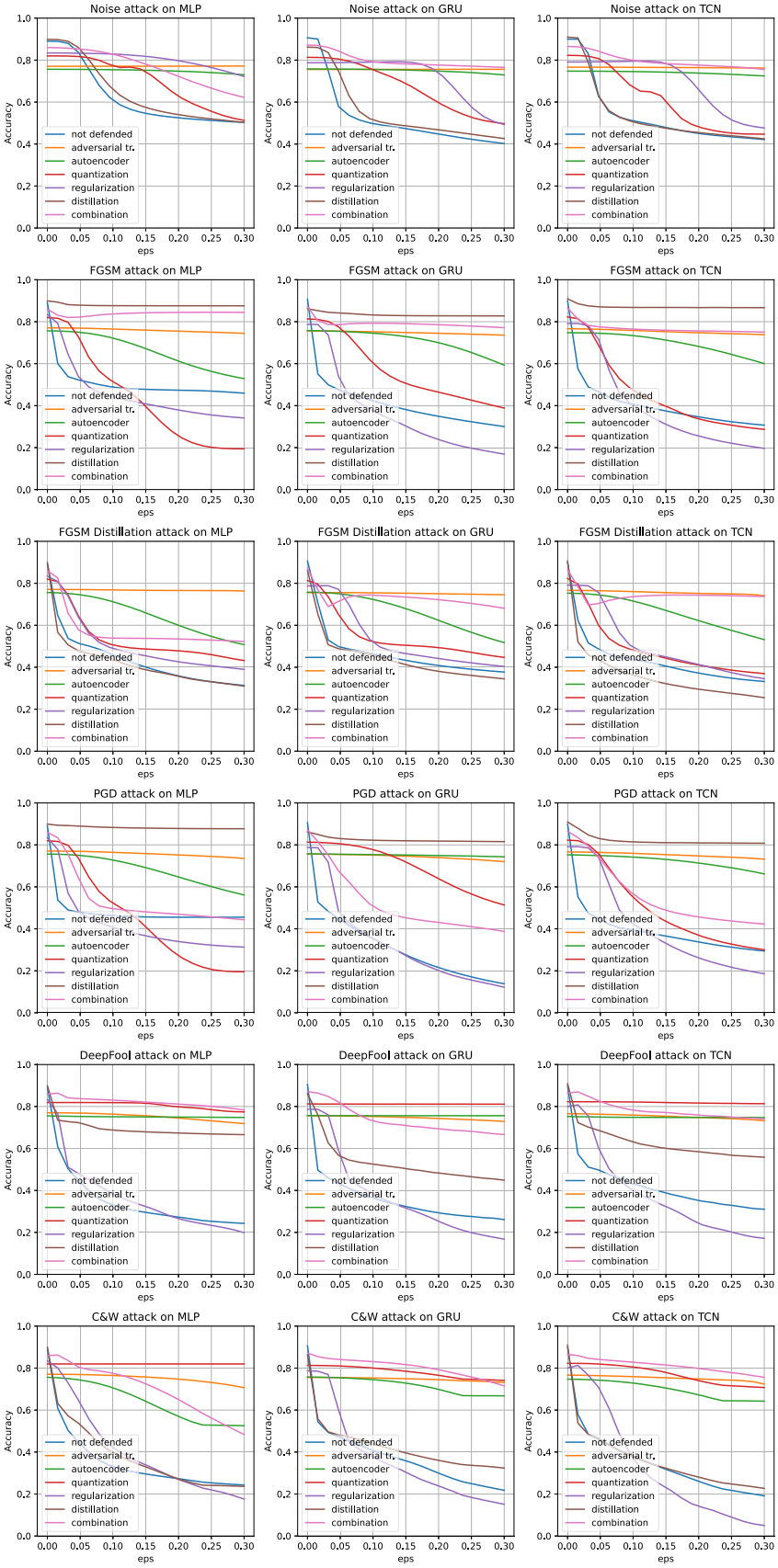

FIGURE 7. Accuracy of protected and unprotected models after adversarial attacks.

TABLE 8 Accuracy of Protected and Unprotected TCN Model After Adversarial Attacks With Different ϵ Values

ϵ		0.015	0.03	0.06	0.09	0.12	0.15	0.18	0.21	0.24	0.27	0.30
Noise attack	Unprotected	**0.89**	**0.83**	0.55	0.51	0.49	0.47	0.46	0.45	0.44	0.43	0.42
	Adversarial training	0.77	0.77	0.77	0.77	0.77	0.76	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**
	Autoencoder	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74	0.73	0.73	0.72
	Quantization	0.82	0.82	0.78	0.68	0.65	0.57	0.52	0.48	0.46	0.45	0.45
	Regularization	0.79	0.79	**0.79**	**0.79**	**0.79**	**0.78**	0.74	0.64	0.54	0.49	0.48
	Distillation	**0.91**	0.79	0.56	0.51	0.49	0.47	0.46	0.45	0.44	0.43	0.42
	Combination (ours)	0.86	**0.86**	**0.83**	**0.80**	**0.79**	**0.78**	**0.78**	**0.78**	**0.77**	**0.76**	**0.76**
FGSM attack	Unprotected	0.57	0.49	0.44	0.41	0.39	0.37	0.36	0.34	0.33	0.32	0.31
	Adversarial training	0.77	0.77	0.76	0.76	**0.76**	0.75	0.75	0.75	0.74	0.74	0.74
	Autoencoder	0.75	0.75	0.74	0.74	0.72	0.71	0.7	0.68	0.65	0.63	0.6
	Quantization	**0.81**	0.77	0.59	0.49	0.43	0.39	0.36	0.33	0.31	0.3	0.29
	Regularization	0.79	**0.78**	0.57	0.41	0.35	0.3	0.28	0.25	0.23	0.21	0.2
	Distillation	**0.89**	**0.88**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**	**0.87**
	Combination (ours)	**0.81**	**0.78**	**0.77**	**0.77**	**0.76**	**0.76**	**0.76**	**0.76**	**0.75**	**0.75**	**0.75**
FGSM distillation attack	Unprotected	0.62	0.52	0.46	0.44	0.42	0.40	0.39	0.37	0.35	0.34	0.33
	Adversarial training	0.77	**0.77**	**0.76**	**0.76**	**0.76**	**0.76**	**0.75**	**0.75**	**0.75**	**0.75**	**0.74**
	Autoencoder	0.75	0.75	**0.74**	0.72	0.69	0.66	0.65	0.62	0.59	0.56	0.53
	Quantization	**0.8**	0.7	0.54	0.49	0.46	0.44	0.43	0.41	0.39	0.38	0.37
	Regularization	**0.79**	**0.79**	0.66	0.51	0.47	0.44	0.43	0.41	0.38	0.36	0.35
	Distillation	0.52	0.46	0.41	0.37	0.34	0.32	0.31	0.29	0.28	0.27	0.26
	Combination (ours)	0.78	0.70	0.72	**0.73**	**0.74**	**0.74**	**0.74**	**0.74**	**0.74**	**0.74**	**0.74**
PGD attack	Unprotected	0.55	0.47	0.42	0.40	0.38	0.36	0.35	0.33	0.32	0.30	0.29
	Adversarial training	0.77	0.77	**0.76**	**0.76**	**0.76**	**0.75**	**0.75**	**0.75**	**0.74**	**0.74**	**0.73**
	Autoencoder	0.75	0.75	0.75	0.74	0.74	0.73	0.72	0.71	0.7	0.68	0.66
	Quantization	0.82	**0.8**	0.69	0.57	0.48	0.42	0.4	0.36	0.34	0.32	0.3
	Regularization	0.79	0.79	0.61	0.43	0.37	0.32	0.29	0.26	0.23	0.2	0.19
	Distillation	**0.88**	**0.85**	**0.82**	**0.82**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**
	Combination (ours)	**0.84**	0.79	0.68	0.58	0.52	0.48	0.47	0.45	0.44	0.43	0.42
DeepFool attack	Unprotected	0.57	0.51	0.48	0.44	0.41	0.38	0.37	0.35	0.33	0.32	0.31
	Adversarial training	0.77	0.76	0.76	0.76	0.76	0.75	0.75	0.75	0.74	0.74	0.73
	Autoencoder	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	**0.75**	**0.75**	**0.75**
	Quantization	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.81**	**0.81**	**0.81**
	Regularization	0.81	0.75	0.5	0.41	0.36	0.31	0.29	0.24	0.21	0.19	0.17
	Distillation	0.72	0.7	0.67	0.64	0.61	0.6	0.59	0.58	0.57	0.56	0.56
	Combination (ours)	**0.87**	**0.85**	**0.81**	**0.79**	**0.77**	**0.77**	**0.76**	**0.76**	**0.75**	**0.75**	0.74
C&W attack	Unprotected	0.58	0.48	0.43	0.39	0.35	0.31	0.29	0.25	0.22	0.21	0.19
	Adversarial training	0.77	0.77	0.76	0.76	0.76	0.75	0.75	**0.75**	**0.74**	**0.74**	**0.72**
	Autoencoder	0.75	0.74	0.74	0.73	0.72	0.7	0.69	0.67	0.64	0.64	0.64
	Quantization	**0.82**	**0.82**	**0.82**	**0.81**	**0.79**	**0.77**	**0.76**	0.74	0.72	0.71	0.71
	Regularization	0.81	0.77	0.61	0.4	0.28	0.22	0.18	0.14	0.1	0.07	0.05
	Distillation	0.54	0.48	0.43	0.38	0.35	0.32	0.3	0.28	0.25	0.24	0.23
	Combination (ours)	**0.86**	**0.85**	**0.84**	**0.83**	**0.82**	**0.81**	**0.81**	**0.80**	**0.78**	**0.77**	**0.76**

REFERENCES

[1] Y.-J. Park, S.-K. S. Fan, and C.-Y. Hsu, "A review on fault detection and process diagnostics in industrial processes," *Processes*, vol. 8, no. 9, 2020, Art. no. 1123.

[2] S. Yousefi, S. Yin, and M. G. Alfarizi, "Intelligent fault diagnosis of manufacturing processes using extra tree classification algorithm and feature selection strategies," *IEEE Open J. Ind. Electron. Soc.*, vol. 4, pp. 618–628, 2023.

[3] F. Winkel, O. Wallscheid, P. Scholz, and J. Böcker, "Pseudo-labeling machine learning algorithm for predictive maintenance of relays," *IEEE Open J. Ind. Electron. Soc.*, vol. 4, pp. 463–475, 2023.

[4] B. Brenner et al., "Better safe than sorry: Risk management based on a safety-augmented network intrusion detection system," *IEEE Open J. Ind. Electron. Soc.*, vol. 4, pp. 287–303, 2023.

[5] M. Bertolini, D. Mezzogori, M. Neroni, and F. Zammori, "Machine learning for industrial applications: A comprehensive literature review," *Expert Syst. Appl.*, vol. 175, 2021, Art. no. 114820.

[6] T. Khoualdia, A. Lakehal, Z. Chelli, K. Khoualdia, and K. Nessaib, "Optimized multi layer perceptron artificial neural network based fault diagnosis of induction motor using vibration signals," *Diagnostyka*, vol. 22, pp. 65–74, 2021.

[7] I. Lomov, M. Lyubimov, I. Makarov, and L. E. Zhukov, "Fault detection in tennessee eastman process with temporal deep learning models," *J. Ind. Inf. Integration*, vol. 23, 2021, Art. no. 100216.

[8] H. Liang, E. He, Y. Zhao, Z. Jia, and H. Li, "Adversarial attack and defense: A survey," *Electronics*, vol. 11, no. 8, 2022, Art. no. 1283.

[9] G. Zizzo, C. Hankin, S. Maffeis, and K. Jones, "Adversarial attacks on time-series intrusion detection for industrial control systems," in *Proc. IEEE 19th Int. Conf. Trust Secur. Privacy Comput. Commun.*, 2020, pp. 899–910.

[10] M. Kravchik and A. Shabtai, "Efficient cyber attack detection in industrial control systems using lightweight neural networks and PCA," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2179–2197, Jul./Aug. 2021.

[11] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Comput. Chem. Eng.*, vol. 17, no. 3, pp. 245–255, 1993.

[12] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic poisoning attacks on and defenses for machine learning in healthcare," *IEEE J. Biomed. Health Inform.*, vol. 19, no. 6, pp. 1893–1905, Nov. 2015.

[13] B. Biggio et al., "Evasion attacks against machine learning at test time," in *Proc. Mach. Learn. Knowl. Discov. Databases: Eur. Conf.*, 2013, pp. 387–402.

[14] H. Teryak, A. Albaseer, M. Abdallah, S. Al-Kuwari, and M. Qaraqe, "Double-edged defense: Thwarting cyber attacks and adversarial machine learning in IEC 60870-5-104 smart grids," *IEEE Open J. Ind. Electron. Soc.*, vol. 4, pp. 629–642, 2023.

[15] M. A. Ayub, W. A. Johnson, D. A. Talbert, and A. Siraj, "Model evasion attack on intrusion detection systems using adversarial machine learning," in *Proc. 54th Annu. Conf. Inf. Sci. Syst.*, 2020, pp. 1–6.

[16] W. Jiang, H. Li, G. Xu, and T. Zhang, "Color backdoor: A robust poisoning attack in color space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 8133–8142.

[17] Z. Yang et al., "Data poisoning attacks against multimodal encoders," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 39299–39313.

[18] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–35, 2022.

[19] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.

[20] J. Niu et al., "A survey on membership inference attacks and defenses in machine learning," *J. Inf. Intell.*, 2024.

[21] D. S. Shaikhelislamov, K. Lukyanov, N. N. Severin, M. D. Drobyshevskiy, I. A. Makarov, and D. Y. Turdakov, "A study of graph neural networks for link prediction on vulnerability to membership attacks," *J. Math. Sci.*, vol. 530, pp. 113–127, 2023.

[22] X. Yuan, K. Chen, W. Huang, J. Zhang, W. Zhang, and N. Yu, "Data-free hard-label robustness stealing attack," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 6853–6861.

[23] J. Zhang, S. Peng, Y. Gao, Z. Zhang, and Q. Hong, "APMSA: Adversarial perturbation against model stealing attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 1667–1679, 2023.

[24] Y. Wang, J. Liu, X. Chang, R. J. Rodríguez, and J. Wang, "DI-AA: An interpretable white-box attack for fooling deep neural networks," *Inf. Sci.*, vol. 610, pp. 14–32, 2022.

[25] R. Singhal, M. Soni, S. Bhatt, M. Khorasiya, and D. C. Jinwala, "Enhancing robustness of malware detection model against white box adversarial attacks," in *Proc. Int. Conf. Distrib. Comput. Intell. Technol.*, 2023, pp. 181–196.

[26] A. N. Bhagoji, W. He, B. Li, and D. Song, "Exploring the space of black-box attacks on deep neural networks," 2017, *arXiv:1712.09491*.

[27] F. Zhang, S. P. Chowdhury, and M. Christakis, "DeepSearch: A simple and effective blackbox attack for deep neural networks," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2020, pp. 800–812.

[28] Y. Bai, Y. Wang, Y. Zeng, Y. Jiang, and S.-T. Xia, "Query efficient black-box adversarial attack on deep neural networks," *Pattern Recognit.*, vol. 133, 2023, Art. no. 109037.
[29] Y. Xiang, Y. Xu, Y. Li, W. Ma, Q. Xuan, and Y. Liu, "Side-channel gray-box attack for DNNs," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 68, no. 1, pp. 501–505, Jan. 2021.
[30] R. Al-qudah, M. Aloqaily, B. Ouni, M. Guizani, and T. Lestable, "An incremental gray-box physical adversarial attack on neural network training," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 45–50.
[31] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks," *Proc. IEEE*, vol. 108, no. 3, pp. 402–433, Mar. 2020.
[32] A. Aldahdooh, W. Hamidouche, S. A. Fezza, and O. Déforges, "Adversarial example detection for DNN models: A review and experimental comparison," *Artif. Intell. Rev.*, vol. 55, no. 6, pp. 4403–4462, 2022.
[33] P. Laykaviriyakul and E. Phaisangittisagul, "Collaborative defense-GAN for protecting adversarial attacks on classification system," *Expert Syst. Appl.*, vol. 214, 2023, Art. no. 118957.
[34] R. Hou, S. Ai, Q. Chen, H. Yan, T. Huang, and K. Chen, "Similarity-based integrity protection for deep learning systems," *Inf. Sci.*, vol. 601, pp. 255–267, 2022.
[35] G. Zizzo, C. Hankin, S. Maffeis, and K. Jones, "Adversarial machine learning beyond the image domain," in *Proc. 56th Annu. Des. Automat. Conf.* 2019, pp. 1–4.
[36] F. Specht, J. Otto, O. Niggemann, and B. Hammer, "Generation of adversarial examples to prevent misclassification of deep neural network based condition monitoring systems for cyber-physical production systems," in *Proc. IEEE 16th Int. Conf. Ind. Inform.*, 2018, pp. 760–765.
[37] A. Hamdi, M. Müller, and B. Ghanem, "SADA: Semantic adversarial diagnostic attacks for autonomous applications," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 10901–10908.
[38] Á. L. P. Gómez, L. F. Maimó, A. H. Celdrán, F. J. G. Clemente, and F. Cleary, "Crafting adversarial samples for anomaly detectors in industrial control systems," *Procedia Comput. Sci.*, vol. 184, pp. 573–580, 2021.
[39] Á. L. P. Gómez, L. F. Maimó, F. J. G. Clemente, J. A. M. Morales, A. H. Celdrán, and G. Bovet, "A methodology for evaluating the robustness of anomaly detectors to adversarial attacks in industrial scenarios," *IEEE Access*, vol. 10, pp. 124582–124594, 2022.
[40] Y. Zhuo, Z. Yin, and Z. Ge, "Attack and defense: Adversarial security of data-driven FDC systems," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 5–19, Jan. 2023.
[41] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
[42] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.
[43] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.
[44] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 39–57.
[45] A. Sinha, Z. Chen, V. Badrinarayanan, and A. Rabinovich, "Gradient adversarial training of neural networks," 2018, *arXiv:1806.08028*.
[46] M. Balunovic and M. Vechev, "Adversarial training and provable defenses: Bridging the gap," in *Proc. Int. Conf. Learn. Representations*, 2019.
[47] S. Szyller and N. Asokan, "Conflicting interactions among protection mechanisms for machine learning models," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 15179–15187.
[48] C. Reinartz, M. Kulahci, and O. Ravn, "An extended tennessee eastman simulation dataset for fault-detection and decision support systems," *Comput. Chem. Eng.*, vol. 149, 2021, Art. no. 107281.
[49] C. Lessmeier, J. K. Kimotho, D. Zimmer, and W. Sextro, "Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification," in *Proc. PHM Soc. Eur. Conf.*, 2016.
[50] M. Z. Ali, M. N. S. K. Shabbir, X. Liang, Y. Zhang, and T. Hu, "Machine learning-based fault diagnosis for single-and multi-faults in induction motors using measured stator currents and vibration signals," *IEEE Trans. Ind. Appl.*, vol. 55, no. 3, pp. 2378–2391, May/Jun. 2019.
[51] M. Unal, M. Onat, M. Demetgul, and H. Kucuk, "Fault diagnosis of rolling bearings using a genetic algorithm optimized neural network," *Measurement*, vol. 58, pp. 187–196, 2014.
[52] V. N. Ghate and S. V. Dudul, "Optimal MLP neural network classifier for fault detection of three phase induction motor," *Expert Syst. Appl.*, vol. 37, no. 4, pp. 3468–3481, 2010.
[53] C. Li, C. Shen, H. Zhang, H. Sun, and S. Meng, "A novel temporal convolutional network via enhancing feature extraction for the chiller fault diagnosis," *J. Building Eng.*, vol. 42, 2021, Art. no. 103014.
[54] H. Zhang, B. Ge, and B. Han, "Real-time motor fault diagnosis based on TCN and attention," *Machines*, vol. 10, no. 4, 2022, Art. no. 249.
[55] J. Zhang, Y. Wang, J. Tang, J. Zou, and S. Fan, "MS-TCN: A multi-scale temporal convolutional network for fault diagnosis in industrial processes," in *Proc. Amer. Control Conf.*, 2021, pp. 1601–1606.
[56] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.
[57] J. Yuan and Y. Tian, "An intelligent fault diagnosis method using GRU neural network towards sequential data in dynamic processes," *Processes*, vol. 7, no. 3, 2019, Art. no. 152.
[58] W. Cui, X. Li, J. Huang, W. Wang, S. Wang, and J. Chen, "Substitute model generation for black-box adversarial attack based on knowledge distillation," in *Proc. IEEE Int. Conf. Image Process.*, 2020, pp. 648–652.
[59] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 135–147.
[60] C. Guo, M. Rana, M. Cisse, and L. V. D. Maaten, "Countering adversarial images using input transformations," 2017, *arXiv:1711.00117*.
[61] C. Finlay and A. M. Oberman, "Scaleable input gradient regularization for adversarial robustness," *Mach. Learn. Appl.*, vol. 3, 2019, Art. no. 100017.
[62] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 582–597.

VITALIY POZDNYAKOV received the master's degree in data science from HSE University, Moscow, Russia, in 2021.

He is currently a Junior Research Scientist with the Artificial Intelligence Research Institute and the ISP RAS Research Center for Trusted Artificial Intelligence, Moscow, Russia. His research interests include the forecasting of multivariate time series by deep generative models and fault diagnosis in large industrial processes by deep learning methods.

ALEKSANDR KOVALENKO received the specialist degree in microelectronics from Saint Petersburg Electrotechnical University "LETI," St. Petersburg, Russia, in 2010, and the master's degree in data science from HSE University, Moscow, Russia, in 2022.

Since 2012, he has worked with industrial equipment as an Electronics Service Engineer. He is currently a Junior Research Scientist with the Artificial Intelligence Research Institute, Moscow, Russia. His research interests include digital twins and deep learning for sensor data analysis.

ILYA MAKAROV (Member, IEEE) received the specialist degree in mathematics from Lomonosov Moscow State University, Moscow, Russia, in 2011, and the Ph.D. degree in computer science from the University of Ljubljana, Ljubljana, Slovenia, in 2021.

Since 2011, he has been a Lecturer with the School of Data Analysis and Artificial Intelligence, HSE University, Moscow, Russia, where from 2012 to 2016, he was the School Deputy Head, and is Associate Professor and Senior Research Fellow. He was also the Program Director of BigData Academy MADE from VK, and a Researcher with Samsung-PDMI Joint AI Center, St. Petersburg Department of V.A, Steklov Mathematical Institute, Russian Academy of Sciences, Saint Petersburg, Russia. He is currently a Senior Research Fellow with Artificial Intelligence Research Institute (AIRI), Moscow, Russia, where he leads the research in industrial AI. He became the Head of AI Center and Data Science Tech Master Program in NLP with the National University of Science and Technology MISIS, Moscow, Russia.

MIKHAIL DROBYSHEVSKIY received the Ph.D. degree in physico-mathematical sciences from the Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow, Russia, in 2019.

He is currently a Research Fellow with the Ivannikov Institute for System Programming. He also participates in an ISP RAS Research Center for Trusted Artificial Intelligence. His research interests include explainable artificial intelligence, methods of attacks and defenses for machine learning models, graph neural networks and their applications, and random graph models.

Dr. Drobyshevskiy is a Member of program committee at Ivannikov ISP RAS Open Conference.

LUKYANOV KIRILL received the master's degree in physico-mathematical sciences from the Moscow Institute of Physics and Technology, MIPT, Phystech, Moscow, Russia, in 2023.

He is currently a Graduate Student Fellow with the Ivannikov Institute for System Programming, Moscow, Russia. He also participates in an ISP RAS Research Center for Trusted Artificial Intelligence. His research interests include machine learning, trusted artificial intelligence, explainable artificial intelligence, methods of attacks and defenses for machine learning models, and graph neural networks and their applications.