DETECTION OF STEALTHY FALSE DATA INJECTION ATTACKS AGAINST STATE ESTIMATION IN ELECTRIC POWER GRIDS USING DEEP LEARNING TECHNIQUES

by

Qingyu Ge

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Engineering

at

The University of Wisconsin-Milwaukee

August 2020

ABSTRACT

DETECTION OF STEALTHY FALSE DATA INJECTION ATTACKS AGAINST STATE ESTIMATION IN ELECTRIC POWER GRIDS USING DEEP LEARNING TECHNIQUES.

by

Qingyu Ge

The University of Wisconsin-Milwaukee, 2020
Under the Supervision of Dr. Lingfeng Wang

Since communication technologies are being integrated into smart grid, its vulnerability to false data injection is increasing. State estimation is a critical component which is used for monitoring the operation of power grid. However, a tailored attack could circumvent bad data detection of the state estimation, thus disturb the stability of the grid. Such attacks are called stealthy false data injection attacks (FDIAs). This thesis proposed a prediction-based detector using deep learning techniques to detect injected measurements. The proposed detector adopts both Convolutional Neural Networks and Recurrent Neural Networks, making full use of the spatial-temporal correlations in the measurement data. With its separable architecture, three discriminators with different feature extraction methods were designed for the predictor. Besides, a measurement restoration mechanism was proposed based on the prediction. The proposed detection mechanism was assessed by simulating FDIAs on the IEEE 39-bus system. The results demonstrated that the proposed mechanism could achieve a satisfactory performance compared with existing algorithms.

To
my parents,
mentor
and advisor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I appreciate my advisor Dr. Wang for the mentorship. I also learned a lot from his course on cyber and physical systems and got to know machine learning could be applied in this field as well.

Second, I am thankful to Dr. Liu, who gave me many useful advices during my research. He was very nice and was always patient in answering my questions.

Finally, thanks to my parents, who supported me to study abroad; and thanks to Dr. Ma, who was my neighbor and helped me; and thanks to everyone who cared about me during this pandemic.

CHAPTER 1 INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

It is an inevitable trend to develop the smart grid in the 21st century. According to the IEEE Grid Vision 2050, the primary expectancy of the smart grid is to have the control and automation processes distributed over the whole power grid to enable efficient and reliable bidirectional power flow [1, 2]. This objective is realized through the integration of the Information and communication technologies (ICT) and supervisory control and data acquisition (SCADA) into the power grid, which is becoming a cyber-physical system (CPS) [1, 3]. ICT relies on common infrastructure like the internet, which may bring up a new challenge. Due to the strong coupling between the physical network and the communication network, attackers can extract and modify information flowing through the communication network, which may influence the physical network and further lead to severe accidents. Such cases have been seen throughout the last decades. On December 23, 2015, a wide blackout took place in Kyiv, Ukraine for several hours since attackers penetrated the SCADA system and opened several circuit breakers in the distribution system [4]. Similarly, there are instances of other attacks, such as the Slammer [5] and the Aurora [6]. There are various forms of attacks, for example, time synchronization attack [7], Denial of Service Attack (DOS) [8], which causes disruptions at the communication system, and False Data Injection Attacks (FDIA) [9], which causes disruptions at the physical system level.

The smart grid relies heavily on SCADA. The most vulnerable part of SCADA systems is the

State Estimation (SE) [10]. The control center obtains real-time data from the SCADA system, estimates the state of the grid, and then takes appropriate actions to ensure regular and stable operation of the power grid peter [11]. However, the state estimator can be fooled by FDIAs to predict wrong states without getting detected [12, 13]. Generally, intruders need to know the complete configuration of the grid to launch such attacks, but recent researches have proven it is possible even if the information is incomplete [14].

Bad Data Detection (BDD) is currently deployed in power grids to secure the integrity of measurement data. BDD filters measurement errors caused by device fault or malicious attack, where it is assumed that bad data will necessarily lead to high residual error [9]. However, the residual error would keep the same as the usual case when a successful FDIA is launched. Thus, it passes by the BDD.

Deep learning is part of a broader family of machine learning methods based on artificial neural networks. Deep belief networks and convolutional neural networks have demonstrated great potential in computer vision, speech recognition, and natural language recognition [15]. The reason why deep learning is so successful is that neural networks have the ability to learn complex structures, and data are exploding in this age. Deep learning is data-driven. It is an excellent function approximator trained using gradient descent algorithm over a specific dataset. Encouraged by its application and effectiveness in time series prediction and anomaly detection, its potential to detect false data injection attacks in the electric power grid is explored in this paper.

1.2 Literature Review

1.2.1 Detection Mechanisms

Since the stealthy FDIAs discussed in this paper are designed to bypass BBD during state estimation, conventional estimation-based methods fail to detect them [13]. With the assumption that the attack vector is sparse, sparse matrix reconstruction methods are employed to identify compromised devices [16, 17]. However, such methods cannot guarantee excellent performance for stealthy FDIAs in certain conditions [18]. Therefore, different approaches need to be explored for this problem. Ozay et al. [18] took the lead in applying machine learning techniques to tackle this anomaly detection problem by regarding it as a classification task. The author tested a group of machine learning algorithms, including Sparse Logistic Regression, k-Nearest Neighbors (KNN), and Support Vector Machines (SVM), whose performance is impressive. However, these models are slow and not scalable for large power systems. Besides, these methods are supervised, which means various scenarios need to be considered as many as possible in the training phase. Therefore, semi-supervised SVM is also applied in [18], which is a good fit when labeled data is little. This approach would take into consideration unlabeled data when being training. Besides, it is mentioned in [19] that the probability density function of the regular classes can be modeled by any density estimation algorithm when only regular data is known, for example, Gaussian Mixture Models [20] or Kernel Density Estimation [21].

Unsupervised learning is another approach where unlabeled data are delivered to the machine for finding classification schemes and patterns, and it is assumed that anomalies are rare in a dataset compared to regular instances [19]. This approach has been applied in many practical

problems, such as network intrusion detection and fraud detection, and can be fit into the detection of FDIAs on power grids. Goldstein et al. [19] evaluated dozens of such algorithms. Most of these algorithms are based on density estimation using KNN or k-means clustering. Points lying outside the margin of density clusters are marked as anomalies [19]. KNN is the most straightforward global unsupervised anomaly detection algorithm. If one point is so far away from its k neighbors that it is classified as an outlier. K-means clustering is another basic algorithm, which is to separate observations into k clusters. If one point does not belong to any cluster, then it is an abnormal sample. This method has been applied to detecting FDIA in energy theft [22]. However, for a dynamic system like the power gird, non-anomalous points can exist in various clusters of density-clouds in an n-dimensional hyperspace. Thus, it is necessary to detect local anomalies [23]. Reference [19] discussed a series of local anomaly detection algorithms, for example, Local Outlier Factor (LOF) and Local Outlier Probability (LoOP). However, since these methods are based on KNN, they cost much time. Apart from that, another common techniques has been utilized in the detection of FDIAs, such as Principal component analysis (PCA) [24] and Isolation Forest [25, 26]. Additionally, one-class SVM can be trained unsupervisedly to address this problem with a soft-margin [27], but it is a challenge to find the best set of hyper-parameters.

Artificial Neural Networks (ANN) have drawn enormous attention in recent years and achieve significant performance in fields of object recognition [28], speech recognition [29], and anomaly detection [30]. No matter in supervised [31], semi-supervised [32], or unsupervised manner [33], deep neural networks have proven itself in anomaly detection [23]. References [34, 35] proposed an extended DBN architecture called Conditional Deep Belief Network

(CDBN) to extract high-dimensional temporal features of FDIAs. Stacked auto-encoder is employed in [36] to extract the nonlinear and nonstationary features of electric load data, which facilitates detection. Apart from that, autoencoders can also be used to reconstruct input data, and then the similarity between original data and reconstructed ones determines whether the input is abnormal or not. Generative models like Generative Adversarial Networks (GANs) [37] and Variational Auto-Encoders (VAEs) [38] are also promising. The discriminator in GANs is a great detector to judge whether the new data are different from learned data. VAEs can transform new data into latent space to see whether they keep in accordance with the distribution of historical data.

Recently, Recurrent Neural Networks (RNNs) are favored to capture the temporal correlations. References [39–41] trained an RNN in a supervised setting to predict the existence of anomalies, and reference [42] adopted the discrete wavelet transform (DWT) as spatial-temporal features extraction tool before using an RNN to make a prediction. RNNs can also be built as autoencoders, and the reconstruction error is the metric of anomaly [41]. Additionally, RNNs are deployed in prediction-based detectors. For example, reference [43] used mean squared error (MSE) between estimated measurements by RNNs and actual measurements as the metric of an anomaly. An anomaly exists if MSE at specific measurements is unreasonably high. Niu et al. [44] followed this idea but added a CNN in front of RNN to adjust the dimensionality of data. Wang et al. [45] introduced the residual architecture into RNNs to improve the performance. The model in [45] divides input time series to the linear part which is predicted by RNNs, and the nonlinear part which is predicted by vector autoregressive processes (VAR). It employs the Weibull distribution to fit SSE between the predicted values

and the observed values to determine the detection threshold. In this paper, the detection mechanism is explored based on prediction using CNNs and RNNs.

1.2.2 Convolutional Neural Networks

Convolutional Neural Network (CNN), a category of feedforward Neural Network (FNN), has succeeded in processing video signals and images, for example, style transfer and recognition of objects. A critical point of CNNs is the local connectivity between neurons in adjacent layers, which are motivated by the structure of the animal visual cortex, whose individual neurons are organized in such a manner as to respond to overlapping regions of the visual field [46, 47]. Since CNNs currently are mainly focused on images, it is unrealistic to connect neurons to all neurons in the previous layer when coping with high-dimensional tensors. In CNNs, every neuron is linked to only a small region of the neurons of the last layer, and hence the network is capable of leveraging the spatially local structure of the data. Such functionality can be borrowed as a preceding procedure to provide spatial features to the following networks.

CNNs are typically made up of three forms of layers: (1) fully-connected layer, (2) convolutional layer (3), and pooling layer. Various instances of CNNs can be roughly defined as the following process, as shown in Figure 1-1:

1. Convolve the source image with a group of filters.

2. Subsample the result of coevolution after being through activation functions.

3. Repeat steps 1 and 2 until sufficiently high-level features are obtained.

4. Attach fully connected layers, a FNN, to the resulting features.

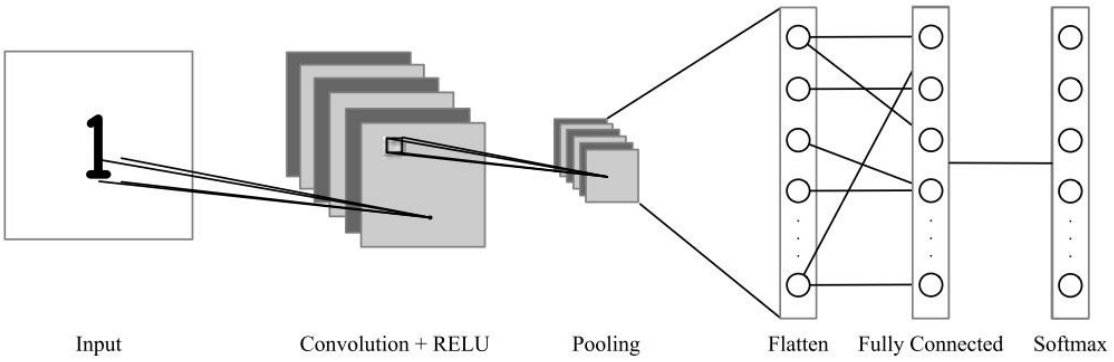

Figure 1-1: An example of an architecture for image classification with a convolutional neural network [44, p. 2]

1.2.2.1 Convolutional Layer

Convolutional layers are critical to a CNN which serve as feature extractors. A convolutional layer is built on a group of learnable kernels, which are 1D or 2D arrays of numbers reflecting how a pixel's neighbors impact on its convolution value. These kernels would be convolved across the input features during the forward propagation, producing a new feature map. Kernels do not need to keep the same height and width. A kernel can be 1D or 2D according to problems. However, dimensions, whether 1D or 2D, must be odd numbers to center the kernel over the region.

Figure 1-2 reveals how convolution works. To calculate the value of each transformed pixel, add the products of each surrounding pixel value with the corresponding kernel value; for instance, in Figure 1-2, -8 is derived because the only product of right-bottom corners is non-

zero. During a convolution operation, the kernel moves across every possible position in the image to repeat this procedure and put the effect to the whole image [48].

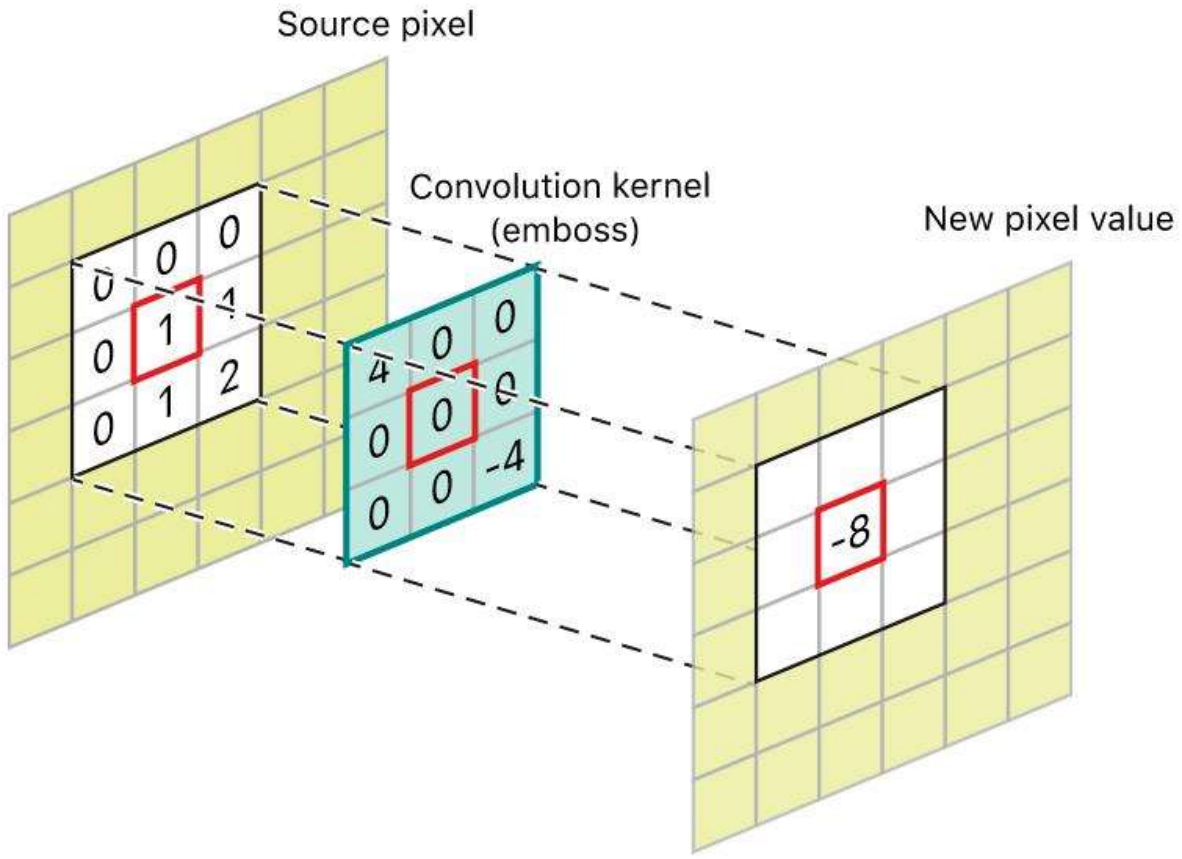

Figure 1-2: Convolution [48]

1.2.2.2 Pooling Layer

Pooling can be regarded as a nonlinear down-sampling, in which various nonlinear functions such as the maximum, the minimum, and the average are implemented. What is used the most common is the Max pooling, which parts the image into non-overlapping patches and picks up the maximum value as the final output for each patch. Figure 1-3 is an example of this procedure.

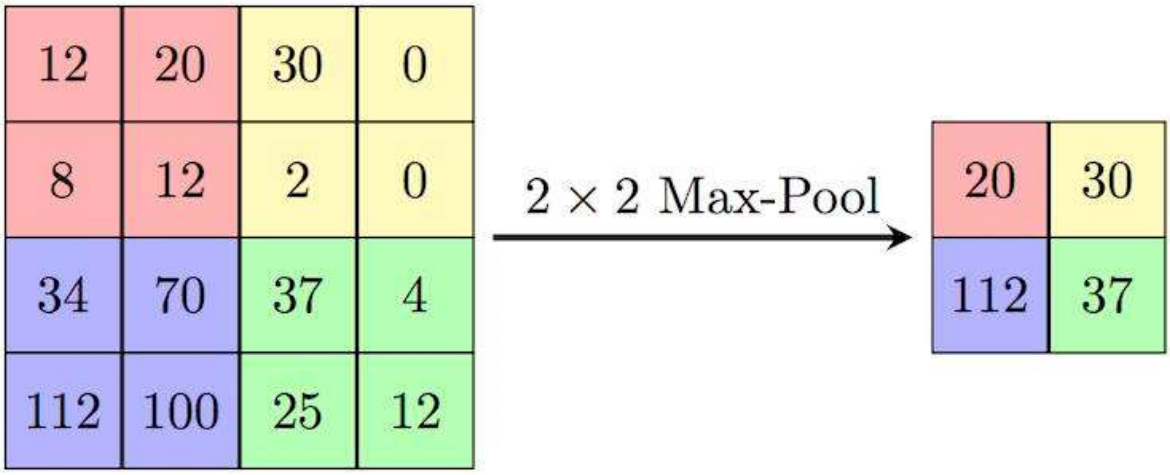

Figure 1-3: Max-pooling [49]

There are three main reasons why max-pooling is useful [50, 51]:

(1) It reduces the computational burden for the following layers by eliminating non-maximal values.

(2) It provides a form of translation invariance so that robustness is assured.

(3) It is a simple approach to reducing the dimensionality of intermediate representations.

1.2.2.3 Fully-Connected Layer

Fully connected layers, which connect every neuron between two layers, play a supporting role in CNN. It is the same as the conventional multi-layer perceptron (MLP) neural network. Generally, it is added between CNNs and other types of neuron networks where the 2D or 3D output would be flattened into a 1D shape.

1.2.3 Recurrent Neural Networks

1.2.3.1 Simple Recurrent Neural Networks

Recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence, which is shown in Figure 1-4. This structure allows it to exhibit temporal dynamic behavior [52].

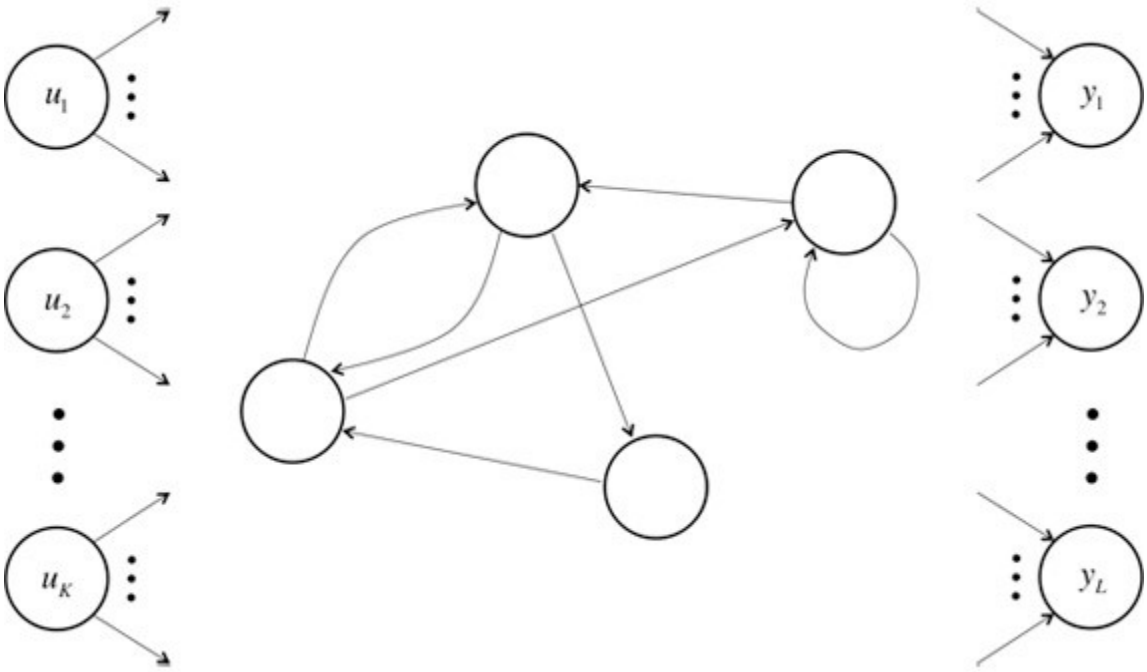

Figure 1-4 Diagram of an RNN [53]

RNNs are fundamentally different from the FNN. For example, the neurons in each hidden layer are connected, and the input of the hidden layer includes not only the output of the input layer but also the output of itself at the previous moment. This structure allows RNNs to process sequence data better and has achieved great success and wide application in Natural Language Processing (NLP).

The training of RNNs is divided into two steps, which are forward propagation and backpropagation. The forward propagation is to calculate the output value. Outputs are generated through activation functions one by one as a time series, which depends on the current input and the previous hidden states. The following Equation illustrates how to calculate the output and hidden state at each time step.

$$\begin{aligned} h_t &= \sigma_h(W_h h_{t-1} + W_i x_t + b_h) \\ o_t &= \sigma_o(W_o h_t + b_o) \end{aligned} \tag{1-1}$$

The meaning of symbols is listed below:

x_t: input vector
h_t: hidden layer vector
o_t: output vector
W b: parameter matrices and vector
σ_h σ_o: Activation functions

Back-propagation uses the chain rule to propagate prediction-error gradients to modify all network weights [54]. An RNN can be seen as deep feedforward networks in which all the layers share the same weights when it is unfolded in time (Figure 1-5) [55].

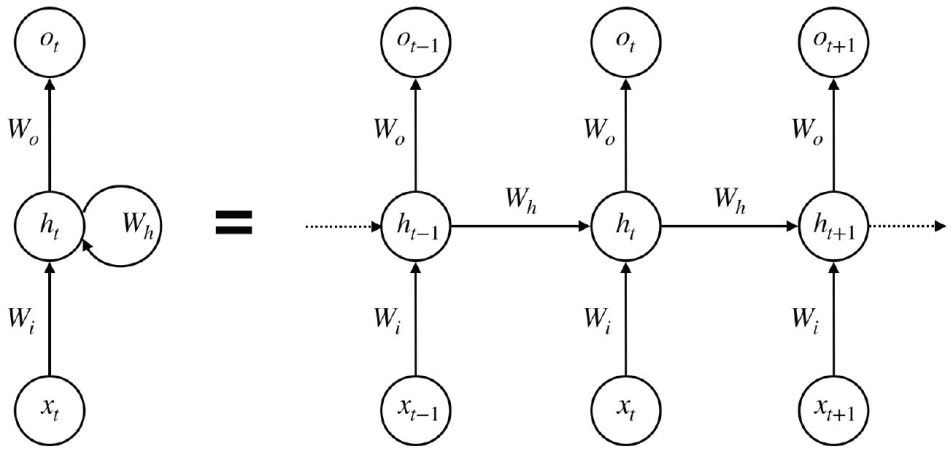

Figure 1-5 An unrolled RNN cell [23]

When backpropagation is applied to this unrolled chain, it is referred to as Back Propagation Through Time (BPPT) [54, 56]. The loss function of RNNs is defined as an overall summation of losses in each time step [57, 58], as shown below.

$$\mathcal{L}(y, o) = \sum_{t=1}^{T} \mathcal{L}_t(y_t, o_t) \tag{1-2}$$

In Equation (1-2), y is the target output. By considering the network parameters in Figure 1-5(1-5 as the set $\theta = \{W_h, W_i, W_o, b_h, b_O\}$ and h_t as the hidden state of the network at time t, gradients can be written as

$$\frac{\partial \mathcal{L}}{\partial \theta} = t = \sum_{k=1}^{t} \left(\frac{\partial \mathcal{L}_t}{\partial \theta} \right) \tag{1-3}$$

where the expansion of loss function gradients at time t is

$$\begin{aligned} \frac{\partial \mathcal{L}_t}{\partial \theta} &= \sum_{k=1}^{t} \left(\frac{\partial \mathcal{L}_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial h_k} \cdot \frac{\partial h_k}{\partial \theta} \right) \\ &= \sum_{k=1}^{t} \left(\frac{\partial \mathcal{L}_t}{\partial h_t} \cdot \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \cdot \frac{\partial h_k}{\partial \theta} \right) \\ &= \sum_{k=1}^{t} \left(\frac{\partial \mathcal{L}_t}{\partial h_t} \cdot \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \cdot \frac{\partial h_k}{\partial \theta} \right) \end{aligned} \tag{1-4}$$

It can be seen from Equation (1-4) that calculation of gradient involves calculating a product of gradients of all hidden states concerning their previous hidden states. Thus, RNNs may suffer a problem of vanishing gradients or exploding gradients with the association of nonlinear activation functions such as “tanh” and “sigmoid”, which nullifies the influence of initial inputs on the final output during the training phase [59]. This phenomenon causes memory of the

network to ignore long term dependencies and hardly learn the correlation between temporally distant events [57]. Fortunately, Long Short Term Memory (LSTM) [60] has been developed to get rid of this problem.

1.2.3.2 Long Short-Term Memory

LSTM is one of the most common and effective ways of reducing the effects of vanishing and exploding gradients [57, 60]. This approach changes the structure of hidden units from "sigmoid" or "tanh" functions to memory components, in which their inputs and outputs are managed by gates. Such gates control the stream of information to hidden neurons and retain features derived from previous time steps [57, 60, 61]. A typical LSTM cell is shown in Figure 1-6.

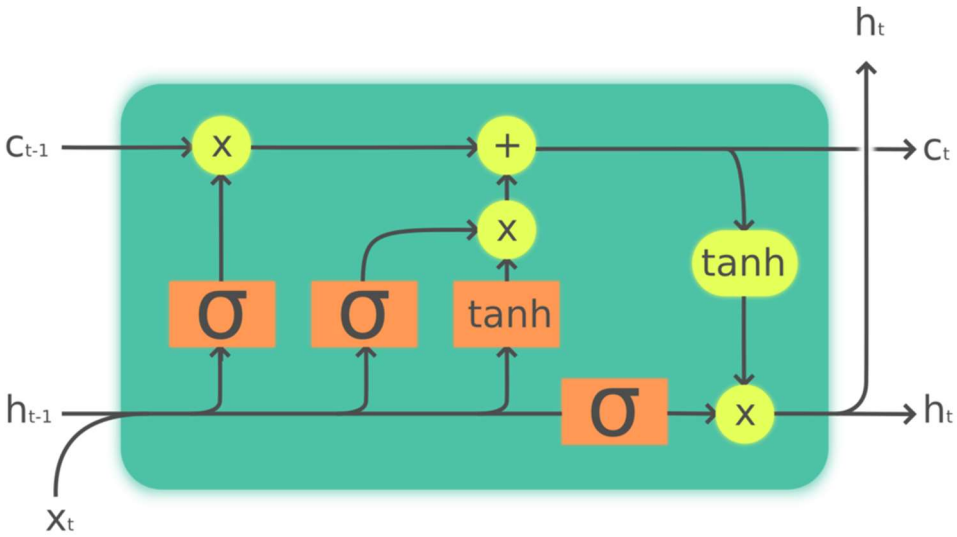

Figure 1-6 The LSTM cell[62]

Compared with simple RNNs, LSTM networks have an extra state variable called cell state. A typical LSTM cell is made of an input gate, a forget gate, and an output gate, and a cell

activation component. The compact forms of the equations for the forward pass of an LSTM unit are: [60, 63, 64]

$$\begin{aligned} f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t \\ h_t &= o_t * \sigma_h(c_t) \end{aligned} \tag{1-5}$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator "$*$" denotes the element-wise product. And other definitions are:

x_t:	input vector to the LSTM unit
f_t :	forget gate's activation vector
i_t:	input/update gate's activation vector
h_t:	output gate's activation vector
o_t:	hidden state vector also known as output vector of the LSTM unit
$\tilde{c}_t$:	cell input activation vector
c_t:	cell state vector
W, U, b:	weight matrices and bias vector parameters which need to be learned during training
σ_g:	activation function for gates
σ_h:	activation function for hidden state

As known from Equation (1-5), if the forget gate is closed, i.e. $f_t = 0$, the historical cell state cannot be involved in the current time step; if the forget gate is open, i.e. $f_t \neq 0$, the historical cell state can be remembered in the current time step; if the input gate is closed, i.e. $i_t = 0$, the input cannot be passed down to the current cell state; if the input gate is open, i.e. $i_t \neq 0$, the input can be passed to the current cell state; if the output gate is closed, i.e. $o_t = 0$, the hidden state of the current cell is prohibited from being passed down; if the output gate is open, i.e. $o_t \neq 0$, the hidden state of the current cell can be passed down. LSTMs are fundamental blocks of our temporal predictor of the whole detector.

1.3 Anomaly Detection Metrics

In general, for binary classification tasks such as anomaly detection, common metrics to judge the performance of a model are Accuracy, Precision, Recall, F1-Score, Receiver Operating Characteristics (ROC) curve and Area Under the Curve (AUC), and Precision-Recall curve [65]. Considering that the frequency of anomalies, in reality, is much smaller than the frequency of non-anomalous conditions, the data set itself is unbalanced. Therefore, metrics such as Accuracy will not be a good choice for evaluation, while Precision and Recall can better evaluate the performance of the model. Both ROC-AUC and Precision-Recall curves can evaluate the overall performance of a model and help to find the optimal threshold. However, the ROC curve is not sensitive to the imbalance of the dataset and can maintain stability when the proportion of positive and negative samples changes. Therefore, six metrics are used in this paper to verify the effectiveness and feasibility of the intrusion detection mechanism: Precision, Recall, F1-Score, ROC-AUC, training time, and test time. In this problem, anomalies are treated as positive cases, and normal samples are treated as negative cases.

Definition:

(1) TN (True Positive) indicates the number of normal data correctly identified as normal data.

(2) FN (False Negative) indicates the number of normal data recognized as anomalies.

(3) TP (True Positive) indicates the number of anomalies correctly identified as anomalies.

(4) FP (False Positive) indicates the number of anomalies recognized as normal data.

Therefore, the first three metrics can be represented as follows:

(1) Precision：

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1-6}$$

It represents the proportion of true anomalies in the samples predicted as anomalies. The higher the Precision is, the better the algorithm is.

(2) Recall：

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1-7}$$

It represents the proportion of the abnormal samples that are correctly predicted. The higher the Recall is, the better the algorithm is.

(3) F1 Score ：

$$\text{F1} = \frac{\text{Precision} \times \text{Recall} \times 2}{\text{Precision} + \text{Recall}} \tag{1-8}$$

It represents the harmonic mean of Precision and Recall. The larger the F1 Score is, the better the overall performance of the detector is.

ROC curve and AUC will be discussed in Chapter 3, where the threshold determination will be

detailed. AUC, along with the F1 Score, is used to reflect the overall performance of the detector.

1.4 Contribution and Roadmap

The contribution of this paper is as follows,

1. An FDIA detection framework, which includes a predictor and a discriminator, is proposed.

2. Three ANN-based discriminators are developed for the FDIA detector.

3. A measurement restoration mechanism is proposed along with a method to determine the optimal threshold for the proposed mechanism.

The rest of this paper is composed as follows: The generation and implementation algorithm of FDIAs are discussed in Chapter 2. Then, the detection model and recovery methodology proposed in this paper are discussed in Chapter 3. Lastly, the experimental results are analyzed and summarized in Chapter 4, followed by a summary in Chapter 5.

CHAPTER 2 PROBLEM FORMULATION AND ATTACK MODELS

2.1 State Estimation

The key to maintaining the reliability of the power system is to monitor the power flows and voltages. The control center receives readings from redundant meters and estimates the state of power system variables from these meter measurements. For example, state variables include bus voltage angles and magnitudes and state estimation problem is to estimate power system state variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ based on meter measurements $\mathbf{z} = (z_1, z_2, \ldots, z_m)^{\mathrm{T}}$, where n and m are positive integers and $x_i, z_j \in R$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$ [66]. To be more specifically, $\mathbf{e} = (e_1, e_2, \ldots, e_m)^{\mathrm{T}}$ with $e_j \in R$, $j = 1, 2, \ldots, m$, are measurement errors. Following formula connects state variables and related measurements.

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{e} \tag{2-1}$$

In Equation (2-1), $\mathbf{h}(\mathbf{x}) = (h_1(x_1, x_2, \ldots, x_n), h_2(x_1, x_2, \ldots, x_n), \ldots, h_m(x_1, x_2, \ldots, x_n))^{\mathrm{T}}$ and $h_i(x_1, x_2, \ldots, x_n)$ is a function of $x_1, x_2, \ldots, x_n$. The state estimation problem is finding an estimate $\hat{\mathbf{x}}$ to $\mathbf{x}$ that is the best match of the measurement $\mathbf{z}$ according to Equation (2-1). Furthermore, Equation (2-1) can be written as a linear regression formula with state estimation using DC power flow model, i.e.,

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{e} \tag{2-2}$$

where $\mathbf{H} = \left(h_{i,j}\right)_{\mathrm{m\times n}}$. Three common statistical estimation criteria have been used in state

estimation: the maximum likelihood criterion, the weighted least-square criterion, and the minimum variance criterion [66]. These three criteria lead to the same estimator with the following matrix solution (2-3), if meter errors are believed to be normally distributed with zero mean.

$$\hat{\mathbf{x}} = (\mathbf{H}^{\mathbf{T}}\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^{\mathbf{T}}\mathbf{W}\mathbf{z} \tag{2-3}$$

In Equation (2-4), $\mathbf{W}$ is a diagonal matrix, which elements are reciprocals of the variances of meter errors, which means

$$\mathbf{W} = \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_m^{-2} \end{bmatrix} \tag{2-4}$$

where σ_i^2 the variance of the i-th meter ($1 \leq i \leq m$).

2.2 Bad Data Detection

Bad measurements can be caused for various reasons, for instance, meter failures or malicious attacks. Strategies for bad measurement detection have been established to secure the state estimation [66]. In general, measurements from regular sensors typically provide an approximation of state variables to their actual values. However, estimated state variables may be shifted away from their actual values by abnormal measurements, which means inconsistency generally exists between good and bad measurements. Because of that, some researchers proposed that the presence of bad measurements can be spotted by calculating the measurement residual $\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}$ between vector of estimated measurements and observed ones and its L2-norm $\|\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}\|$. More precisely, the L2-norm $\|\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}\|$ has to be compared with

a threshold τ which is set by humans, and the existence of bad measurements can be inferred when $\|\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}\| > \tau$. Now the problem is transmitted into another form, the selection of τ, which is a critical part.

Before solving this problem, some assumptions need to be set up: (1) all state variables are mutually independent, and (2) meter errors obey the normal distribution. Sequentially, $\|\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}\|^2$ which is denoted as $\mathcal{L}(\mathbf{x})$, satisfies a $\chi^2(v)$ distribution, where $v = m - n$ is the degree of freedom. Reference [66] gives out that the threshold τ can be determined through a hypothesis test with a significance level of α which means the probability of $\mathcal{L}(\mathbf{x}) \geq \tau$ is equivalent to α. Thus, the existence of bad measurements is suggested by $\mathcal{L}(\mathbf{x}) \geq \tau$ with false alarm of probability α.

The detection mechanism by the L-2 norm is widely used in power systems, which causes a loophole for attackers. A smart attacker can design false data deliberately to deceive the estimator without activating any alarm. Notably, the credibility of the state estimator is undermined by the attacker through compromising a subset of meters and then submitting modified readings [13]. Denoting $\mathbf{c}$ as the deviation vector of the estimated state variables before and after the attack, the estimated state vector being hacked can be expressed as $\hat{\mathbf{x}}_{\text{bad}} = \hat{\mathbf{x}} + \mathbf{c}$. Additionally, denoting $\mathbf{a} = (a_1, \ldots, a_m)^{\text{T}}$ as the nonzero vector injected to the measurement data $\mathbf{z} = (z_1, \ldots, z_m)^{\text{T}}$, the measurement vector being hacked can be expressed as $\mathbf{z}_{\text{bad}} = \mathbf{z} + \mathbf{a}$.

With the DC power system model, the estimated state variables after FDIAs are as follows [66]:

$$\begin{aligned}\hat{\mathbf{x}}_{\text{bad}} &= (\mathbf{H^T W H})^{-1}\mathbf{H^T W z}_{\text{bad}} \\ &= (\mathbf{H^T W H})^{-1}\mathbf{H^T W}(\mathbf{z}+\mathbf{a}) \\ &= \hat{\mathbf{x}} + (\mathbf{H^T W H})^{-1}\mathbf{H^T W a} \\ &= \hat{\mathbf{x}} + \mathbf{c}\end{aligned} \tag{2-5}$$

and the new $\mathcal{L}(\mathrm{x})$ can be computed as:

$$\begin{aligned}\mathcal{L}(\mathbf{x})_{\text{bad}} &= \|\mathbf{z}_{\text{bad}} - \mathbf{H}\hat{\mathbf{x}}_{\text{bad}}\|^2 \\ &= \|\mathbf{z}+\mathbf{a}-\mathbf{H}(\hat{\mathbf{x}} + (\mathbf{H}^{\mathrm{T}}\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{W}\mathbf{a})\|^2 \\ &= \|\mathbf{z}-\mathbf{H}\hat{\mathbf{x}} + (\mathbf{a}-\mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{W}\mathbf{a})\|^2 \\ &= \|\mathbf{z}-\mathbf{H}\hat{\mathbf{x}} + (\mathbf{a}-\mathbf{H}\mathbf{c})\|^2\end{aligned} \tag{2-6}$$

If $\mathbf{a} = \mathbf{Hc}$, then $\mathcal{L}(\mathbf{x})_{\text{bad}} = \mathcal{L}(\mathbf{x})$, which means bad data are successfully injected into meter measurements while leaving the residual value unchanged.

2.3 False Data Injection Attack

Yao Liu etc. proposed the concept of such stealthy FDIAs in [13] in 2011. He sorted the FDIAs as random FDIA and targeted FDIA, considering the possible attack goals. Random FDIA, in which the attacker seeks to find any attack vector as long as it is able to result in an inaccurate estimation of state variables. For targeted FDIA, the attacker seeks to find an attack vector which is able to inject a specific error into certain state variables. While the latter type of attack can potentially cause more damage to the system, the former one is easier to launch. In a targeted false data injection attack, two cases are considered: a constrained case and an unconstrained case. In the constrained case, the attacker wants to launch a targeted false data injection attack that only changes the target state variables but does not contaminate the other state variables. The constrained case reflects the situations where the control center has the ability to verify estimates of the other state variables. In the unconstrained case, the attacker

does not have any concern about the influence on the other state variables while compromising the chosen ones. All these situations are under the consideration of limited access to meter devices from the attacker's perspective, which bothers attackers a lot, because attackers have to try many times to construct a satisfying attack vector. Whether easy or difficult to launch successful FDIAs depends on how many devices the attacker can approach. The more meters are accessed by the attacker, the larger the probability of success is.

Since the detection mechanism focused on in this paper is to defend such attacks, a strong adversary is expected. Thus, assuming the attacker, who is eager to cause a severe impact on the system, has the best knowledge of the entire power system and construct attack vectors based on that. Therefore, targeted attacks under the constrained case are chosen in this paper to test the detection mechanism. In constrained case, the attacker can construct an attack vector $\mathbf{a}$ by substituting fixed $\mathbf{c}$ into the relation $\mathbf{a} = \mathbf{Hc}$.

2.4 Attack Generation Algorithms

When generating targeted attack as training data, in order to make the simulation more realistic, the following factors need to be considered:

1. The proportion of violated state variables $\frac{k}{n}$, where k is the number of compromised state variables, and n is the total number of state variables. Typically, the ratio from 10% to 100% with a 10% step is of interest.

2. The duration of the attack t_1. An attack may last for a while, interfering with measurement data at several points of time.

3. The period between every attack t_2. The shorter the interval is, the more frequently the grid is hacked, while the longer the interval is, the safer the grid is, and fewer attacks occur.

Furthermore, two forms of targeted attacks are considered, the normal targeted attacks and the playback targeted attacks.

1. **The normal targeted attacks**: The attacker would change the states to any values he/she wants. It is impossible to guess the false data the attacker plans to inject, but the distribution of changes in state variables can be assumed beforehand. In this way, attacks of different intensities can be generated as many as possible. Normal distribution or uniform distribution is a common choice.

2. **The playback targeted attacks**: False data can be constructed deliberately based on a real historical event and then injected into the system. Sequentially, such kind of attack may not be likely to be detected successfully only if a static method was applied [44].

At the same time, considering the extreme case, we assume that the attacker can access all the measurement devices and have a full understanding of the structure of the entire power grid. It is difficult to achieve this, because there are always some meters that are not easy to access, such as the measurement points in a substation [13], and the structural parameters of the power grid cannot be easily obtained by outliers. The advantage of considering extreme conditions is that it can cover all the situations that may occur in reality. The labels for measurements are set to 0 when it is safe while 1 when it is attacked. As for the obtainment of measurements, real-world load curves are used in the simulation, and thus measurements can be generated at every

time step, and then the noise is added.

Based on what is discussed before, the algorithm of the generation of attacks is designed as follows:

Algorithm 1 Generation of Attacks

Procedure GENERATE_ATTACK ($attack\_type$, $measurements$, $states\_variables$, k/n, $distribution$)
  for i in range (size_of ($measurements$)) **do**
    $hacked\_state \leftarrow$ choice ($states$, int ($k/n \times$ size_of ($states$)))
  $noise \leftarrow \text{Norm}(\mu, \sigma^2)$
    $z[i] \leftarrow measuremtns[i] + noise$
    $t = 0$
    if $t < (t_1 + t_2)$ **do**
      if $t = 0$ **do**
        if $attack\_type \neq$ "playback" **then**
          $a \leftarrow$ get_normal_targeted_attack ($hacked\_state$, t_1, $distribution$)
        else
          $a \leftarrow$ get_playback_targeted_attack ($hacked\_state$, t_1)
        end if
      end if
      if $t \leq t_1$ **do**
        $z[i] \leftarrow z[i] + a$
        $label[i] \leftarrow 1$
      else
        $label[i] \leftarrow 0$
      end if
      $t \mathrel{+}= 1$
    else
      $t = 0$
    end if
    save ($z[i]$)
  end for
end procedure

CHAPTER 3 DETECTION AND RESTORATION

3.1 Detection

As mentioned earlier, FDIAs detection can be treated as a problem of detecting anomalies in a time series. It is because sparse attack vectors are added to the actual measurements [17], thereby violating the time structure of the data. Therefore, by predicting the measurement value at the next moment and comparing it with the actual measurement value, an abnormality can be detected. As shown in Figure 3-1, the detector in this paper consists of two parts. The first part is called predictor, which is based on the model proposed in [44], takes in measurements over a window of length k in time step $[t-k, t-1]$ to predict the measurement value at time step t. The second part is called discriminator, which compares the predicted measurement value with the actual one to judge the current condition of power system whether being hacked or not. Since predictor needs to be trained through unsupervised learning, that is, only historical measurement data are required to make predictor obtain the ability of prediction, while the discriminator needs to take in historical data and corresponding labels to learn the difference between predicted measurement value and the actual value, which is supervised learning. When two parts are combined, they make up a semi-supervised detector.

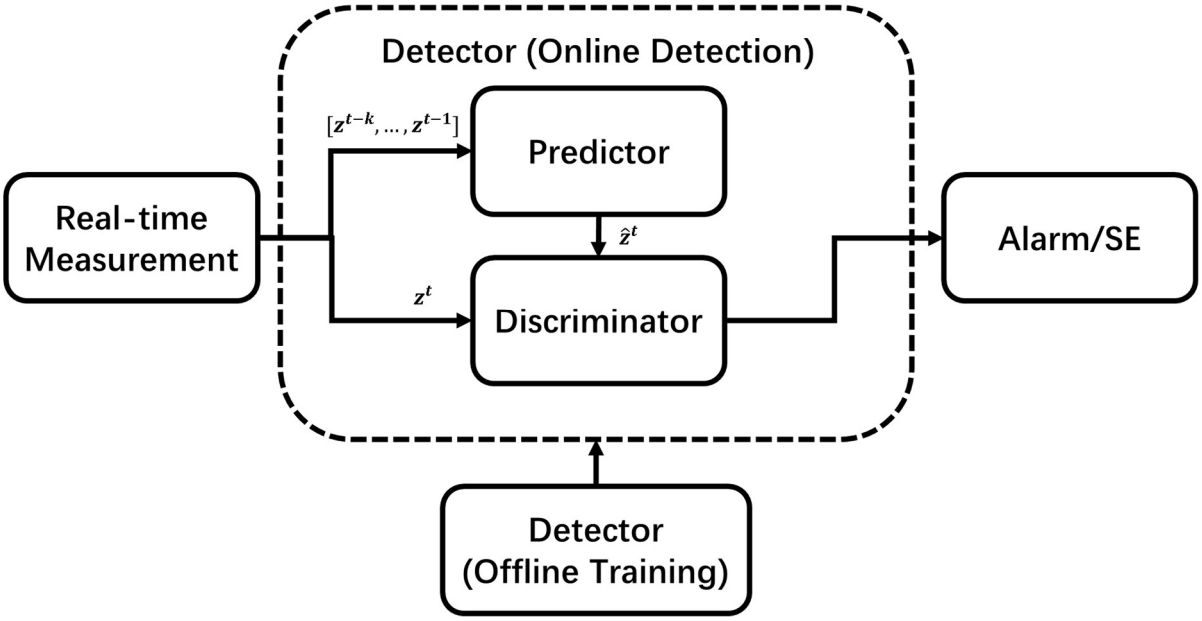

Figure 3-1 Architecture of Semi-Supervised Detector

3.1.1 Predictor

The predictor takes in the measurement data in the time steps $[t-k, t-1]$ as input, and outputs the predicted measurement value at time t. The predictor consists of two parts, the first half is a CNN, and the second half is a bidirectional LSTM network. The CNN extracts the features of the sequence by learning the spatial structure of the measurement data, which is hidden under the noise at every moment, and then compresses and maps the features into the low-dimensional latent space. It plays the role of dimensionality reduction during this process. The bidirectional LSTM network learns the temporal structure of the compressed data so that the data at the next moment can be inferred in the latent space according to the characteristics of the previous $(k-1)$ time steps. Finally, predicted data are upgraded from low dimension to the actual dimension through a linear layer. The architecture of the predictor is shown in Figure 3-2.

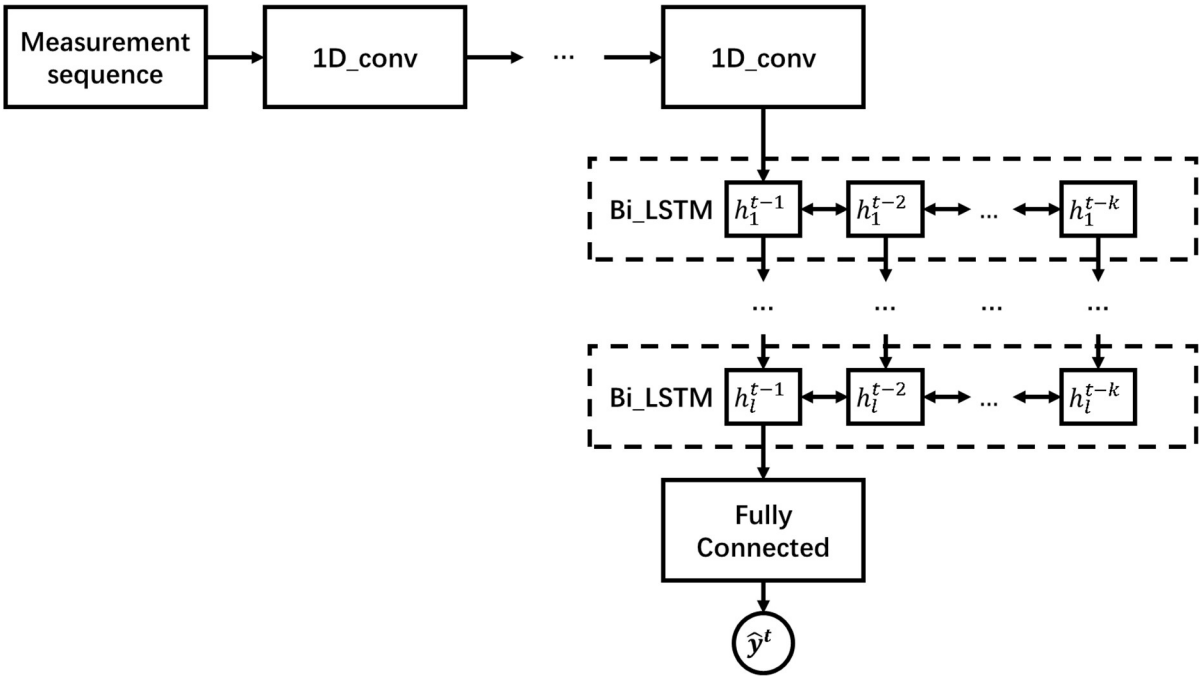

Figure 3-2 Architecture of the predictor

The network is trained based on historical measurement data and does not need any labels. Therefore, it is unsupervised learning. Since it is a regression problem to predict future data based on historical data, mean square error (MSE) is selected as loss function, as shown in Equation (3-1), which is being minimized when the predicted value is approaching the actual value.

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3-1}$$

In Equation (3-1), y_i is the i-th actual value and $\hat{y}_i$ is the i-th predicted value, n is the number of samples in a batch.

3.1.2 Discriminator

The discriminator is primarily a Multi-layer Perceptron (MLP). It is a feedforward artificial

neural network model that maps multiple input data sets to a single output data set. The MLP is fully connected between layers. The discriminator takes in the predicted value and the actual value, finds out differences between the two vectors, and then learns the nonlinear mapping relationship between the differences and the labels through the multiple hidden layers. Finally, a sigmoid function is used to calculate the probability of whether the system being hacked at the current moment or not. Figure 3-3 shows the architecture of the discriminator.

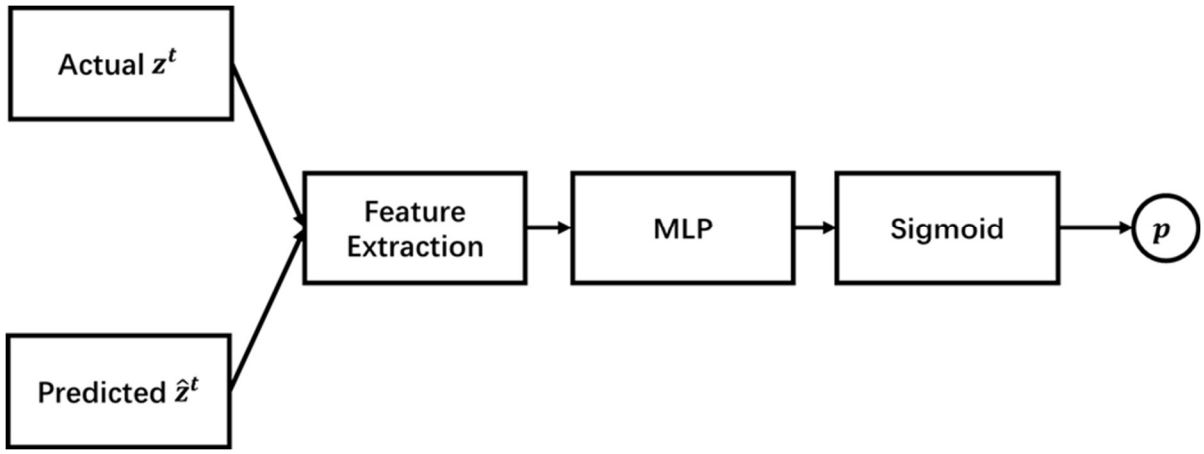

Figure 3-3 architecture of the discriminator

To be able to distinguish the difference between the predicted value and the actual value after the power grid being attacked, three designs of the neural network are proposed, which varies in the feature extraction part in the proceeding stage. They are concatenation-based discriminator, convolution-based discriminator, and square-error-vector-based discriminator.

3.1.2.1 Concatenation-Based Discriminator

The predicted vector and the actual vector are spliced together to obtain a new vector with a double-length, which is used as the feature vector to be further processed by the subsequent MLP, as shown in Figure 3-4. The MLP can automatically pair off each measurement with its

counterpart and recognize the correspondence between these pairs and final decision through training.

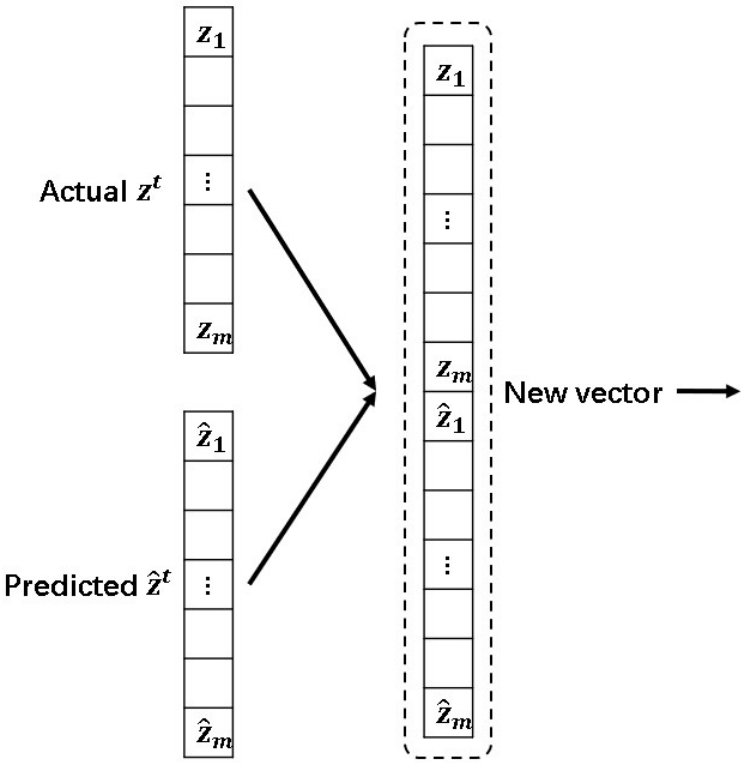

Figure 3-4 Diagram of concatenation

3.1.2.2 Convolution-Based Discriminator

As mentioned earlier, one of the advantages of CNNs is feature extraction. Thus it is widely used in image processing. Inspired by this, CNNs are used to extract the difference between the two vectors. Since what is of interest is the difference between corresponding elements of the two vectors, rather than the difference between different elements, a special way is proposed in this paper to make the two vectors suitable for convolution processing. According to the format of an image, the elements of a vector are regarded as pixels, and the two vectors are regarded as two channels. The synthesized "image" is scanned by a kernel with the size of (1,1)

to perform the convolution, and the result goes through activation functions, which guarantees that only “channels” on a single “pixel” participate in each calculation. With the “image” passing through multiple convolutional layers, the number of channels increases. It should be noted that pooling is not required in this process. All channels need to be expanded into a one-dimensional vector before going to the MLP. Figure 3-5 illustrates this process. In this way, the difference between the corresponding elements of the predicted vector and the actual vector can be extracted and stored in this flatten vector from which the MLP can figure out whether the system being attacked or not.

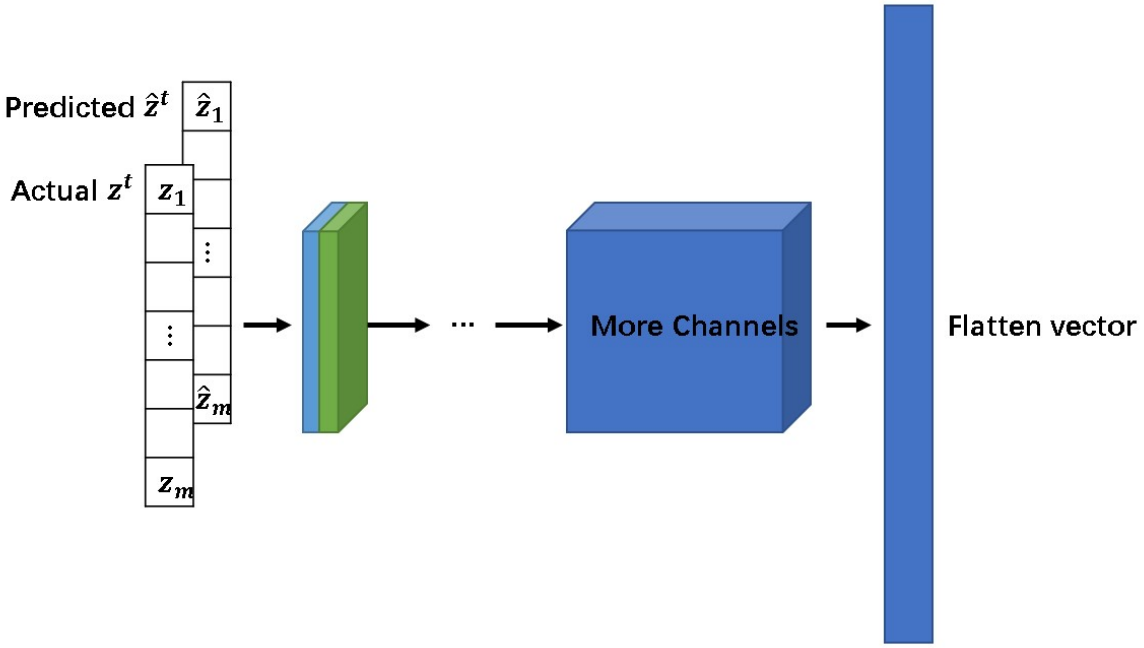

Figure 3-5 Diagram of convolution

3.1.2.3 Squared-Error-Vector-Based Discriminator

This scheme is to directly calculate the squared errors of the predicted vector and the actual vector, which are used as features to be passed down to the following MLP. Through the

squaring operation, any error greater than one is amplified while the one less than one is reduced, making the difference more distinct.

3.1.2.4 Difference Between Three Discriminators

From the concatenation-based discriminator to the square-error-vector-based discriminator, the burden on the MLP is decreasing. In concatenation-based discriminator, the MLP shoulders the responsibility of analyzing concatenated vector and making the judgment. In convolution-based discriminator, the CNN takes on the responsibility of analyzing the deviation between predicted vector and actual vector while the MLP focuses on making decisions. However, what features the CNN extracts are unclear, while in squared-error-vector-based discriminator, it is assumed that squared error can be a good fit that CNN can be removed. Additionally, CNN provides multiple features, while the squared error is a single feature. The performance of these three designs is presented in Chapter 4.

3.1.2.5 Training for Discriminator

Training discriminator is slightly complicated. First, the predictor gives out the predicted value at each moment to form a prediction data set, which is used as the training data set for the discriminator together with the actual data set and the labels obtained during the attack generating stage. For the 0-1 binary classification task, binary cross-entropy loss is the most appropriate estimator, which is presented in Equation (3-2) [67].

$$L(c,p) = -\frac{1}{n}\sum_{n} c\,log(p) + (1-c)\,log(1-p) \tag{3-2}$$

In Equation (3-2), c is the true label ($c = 1$ for the attacked case and $c = 0$ for secure case), and p is the predicted label, which can also be seen as the probability of positive class (being attacked), n is the number of samples in a batch.

It should be noted that, during the training process, the neural network minimizes the empirical loss. Thus, unbalanced data set where the number of anomalies is much less than the normal ones will make the discriminator impossible to learn how to judge anomalies. Such occasions may happen when realistic labeled data is used as the training set, which needs particular attention when training network. However, the occurrence of such a situation can be avoided by weighting the loss function: the penalty can be increased when anomalies are missed (false negatives) to make loss larger than when discriminator fails to predict the normal situations (false positive) [41]. The weighted loss function, for example, is described in Equation (3-3), where a is the penalty multiplier for false negatives. Also, Due to the neural network is trained using mini-batch stochastic gradient descent, another solution is to ensure that the normal data and the abnormal data are evenly distributed for the network to learn when constructing mini-batches. However, this approach will distort the actual distribution of the data and cause deviations for the discriminator.

$$L(c,p) = -\frac{1}{n}\sum_{n} a\, c \log(p) + (1-c)\log(1-p) \tag{3-3}$$

However, in the training task of this paper, the data set can be carefully designed so that the number of positive and negative samples is roughly equal to ensure the balance of the data set. Even Equation (3-2) can still ensure that the discriminator is perfectly trained.

3.1.3 Measurement Restoration

With the predicted measurements from the predictor at the current time point, if the alarm was triggered by the discriminator in which the power grid was under attack, the damaged measurements could be repaired by predicted ones. For the measurement vector that needs to be repaired, firstly, the element that has the largest error from the predicted one needs to be found out, then replaced by the corresponding predicted value. The MSE is required to be calculated after every substitution to see whether it is less than a certain threshold. If the predetermined accuracy is reached, the repair work is done. The key is to determine the proper threshold. First of all, the prediction results given by the predictor are not 100% perfect, including errors caused by the unpredictability of noise. Therefore, the average error in the safe state can be used as the threshold for restoration. Accordingly, **Algorithm 2** is defined, and Figure 3-6 illustrates the process.

Algorithm 2 Measurements Restoration

1: **Procedure** REPAIR_MEASUREMENTS ($real\_z$, $pred\_z$, $threshold$)
2: **while** mse_of ($real\_z$, $pred\_z$) > $threshold$ **do**
3: $index \leftarrow$ arg_max(get_error ($real\_z$, $pred\_z$))
4: $real\_z[index] \leftarrow pred\_z[index]$
5: **end while**

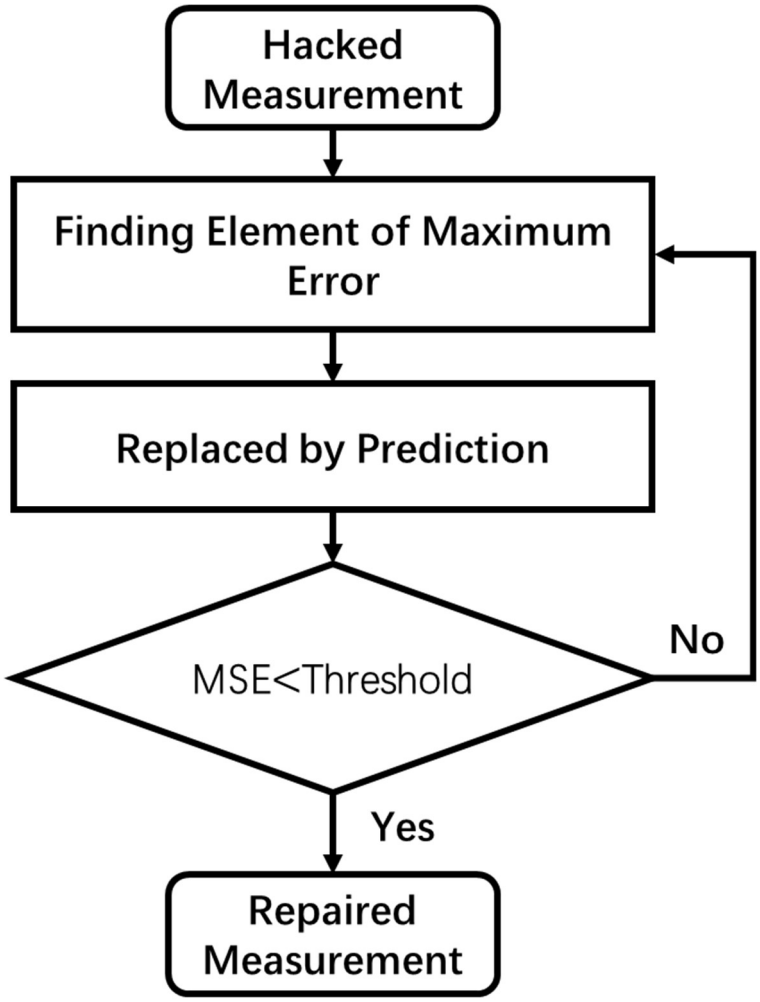

Figure 3-6 Flow chart of measurement restoration

3.1.4 Detection and Restoration Mechanism

The detection method and restoration method mentioned above are combined to obtain the comprehensive detection and restoration mechanism proposed in this paper. The real-time measurement data is continuously inputted into the detector, and the detector gives out the prediction of the current state of the power grid. If the grid is attacked, the alarm sets off, and the damaged data are handed over to the restorer for repair. After the repair is completed, the repaired measurement data are passed to the state estimator; If the power grid is safe, the measurement data are directly used for state estimation. In both branches, the measurement

data are stored for offline training of the detector. Figure 3-7 details the architecture of such an integrated mechanism.

Since the structure of power grids and load characteristics are changing with time, it is necessary to train and update the detector periodically. In other words, short training time and prediction time are of importance, because the time required for training a neural network and prediction is directly related to the depth and scale of the network, which means that the detector cannot have a too complex structure.

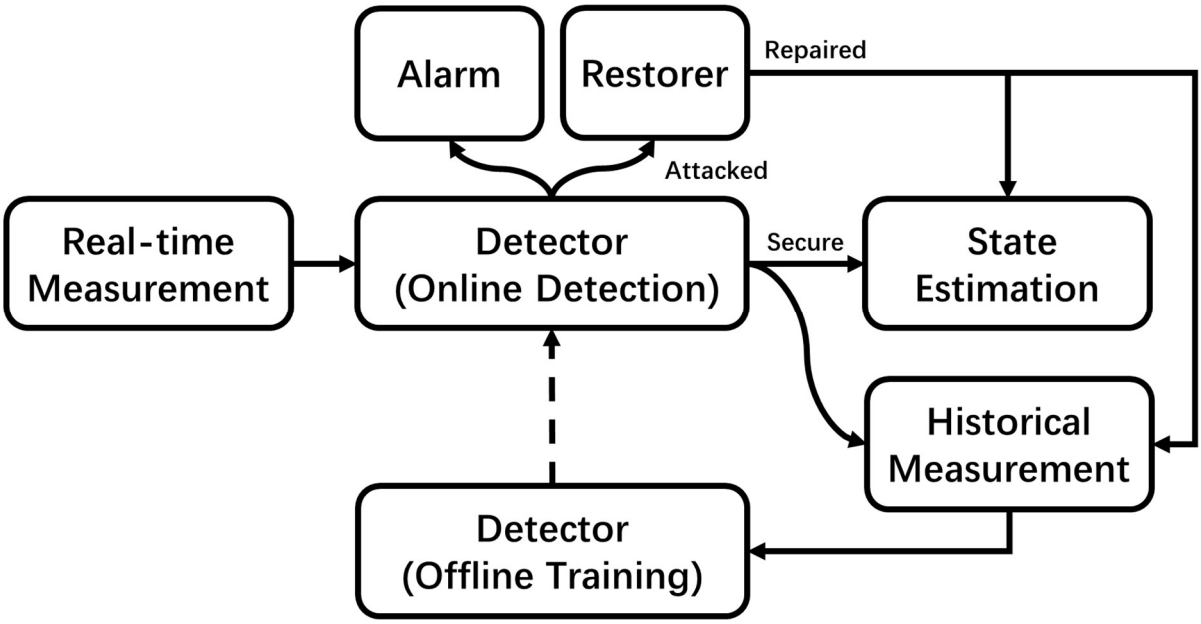

Figure 3-7 Architecture of detection and restoration mechanism

3.1.5 Data Preparation and Training

Since this is a many-to-one model of sequence prediction, batches of time series and the corresponding state of the system for the last time step of the series need to be prepared. A fixed-length rolling window has been used to create training pairs. The inputs for detector are the measurements within a time window (predictor's input), and the measurement of the next

time step to the window (along with the predictor's predicted measurement as the discriminator's input). The output of the detector is the label corresponding to the system state at a single time step. The detector is trained by the ADAM [68] optimizer using mini-batch gradient descent. Dropouts are used to prevent the overfitting of the model [69]. The dropouts technique randomly drops out the neural connections between layers, thereby improving the generalization ability of the detector and making it more robust to noise.

3.1.1 Optimal Threshold Determination

The optimal threshold can be determined through the ROC curve and then be used to calculate Precision, Recall, and F1 Score for the proposed detector.

ROC curve is a graphical plot which illustrates the diagnostic ability of a binary classifier system when its discrimination threshold varies. The ROC curve is defined by depicting the true-positive rate (TPR) against the false-positive rate (FPR) at various threshold settings, which plots relative tradeoff between false positives (costs) and true positives (benefits). The TPR is also known as the sensitivity, Recall, or probability of detection in machine learning. The FPR, also known as the probability of false alarm, is equal to (1 − specificity), where the specificity means true-negative rate (TNR), which measures the proportion of actual negatives that are correctly detected [70, 71].

In ROC space, a diagonal line from the left bottom corner to the top right corner, also known as the line of no-discrimination, represents points set by a random guess. Points under the diagonal represent poor classification results (worse than the random guess), while points

above the line indicate great results (better than the random guess). The best potential classifier would yield a point in the upper left corner or coordinate (0,1) of the ROC region, indicating 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a perfect classification. It should be kept in mind that the output of a consistently bad classifier could simply be reversed to get a good classifier [70].

As a result, the point near the perfect classification should be picked up as an optimal cutoff point, which can be screened out by minimum Euclidean distance from (0,1) or maximum Youden index J, which is the vertical distance between the point on the ROC curve and the diagonal line. The formula to calculate Youden index J is as follows [72],

$$J = \text{sensitivity} + \text{specificity} - 1 \tag{3-4}$$

which can be expanded as follows by notations in section 1.3 of Chapter 1, that is

$$J = \frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}} - 1 \tag{3-5}$$

Youden index is used to search out the best threshold in this paper.

CHAPTER 4 EXPERIMENTS AND OBSERVATIONS

4.1 Data Generation, Test Cases, and Model's Parameters.

In order to make the simulation more realistic, actual load data from the New York Independent System Operator (NYISO) [73] were used, which was a total of 9094 points sampled every 5 minutes in a month. These load data were applied to the IEEE 39-bus case [74] (Figure 4-1),

and the AC power flow was calculated by MATPOWER [75]toolbox to obtain measurements. Since there were only 11 load profiles from the NYISO data set, all the curves were scaled and replicated to match the IEEE 39-bus system while inactive power injections kept unchanged. Part of the ideal operating conditions of bus 1 is shown in Figure 4-2. The obtained measurements include active power flow of 39 buses and 56 branches, 85 measurements in total. Besides, gaussian noise with a mean of zero and a standard deviation of 0.02 was added to obtained measurements.

Based on the measurements, the algorithm discussed in Chapter 2 was used to generate attacks. For ordinary targeted FDIAs, changes with a uniform distribution $[-10\%, 10\%]$ were applied to state variables; for playback FDIAs, a delay of 3 time steps was applied to state variables, i.e., real load data 15 minutes ago were used to generate attack vectors. Besides, the width of the rolling window to create training pairs was set to three time steps, along with the duration of each attack was four time steps, and the break between every attack was three time steps. The reason why it was set up in this way is that these parameters can simulate various scenarios where how many hacked measurement vectors account for each time window. For example, the grid being attacked at all time steps or only one time step in a window results in different errors for prediction. As mentioned before, different attack levels were considered, in which the number of attacked state variables increases from 10% to 100% by 10%. All measurements were repeatedly applied to each attack level. Additionally, assume all devices can be accessed by an attacker so that situations can be covered as many as possible.

In this experiment, the detector was trained for only three epochs to avoid overfitting and tuned

up by Bayes Optimization. The predictor contained two convolutional layers with ReLU activation functions, which reduced the dimensionality of measurements from 85 to 32, and 3 bidirectional RNN layers with LSTM cells along with a dense layer at the end to output predictions. The discriminator contained two hidden layers with ReLU activation functions. All dropout rates were set to 0.5 during tuning up.

In order to verify the effectiveness of the proposed detection mechanism, results were compared with other machine learning algorithms, and the effectiveness of the detection mechanism was analyzed in different attack scenarios.

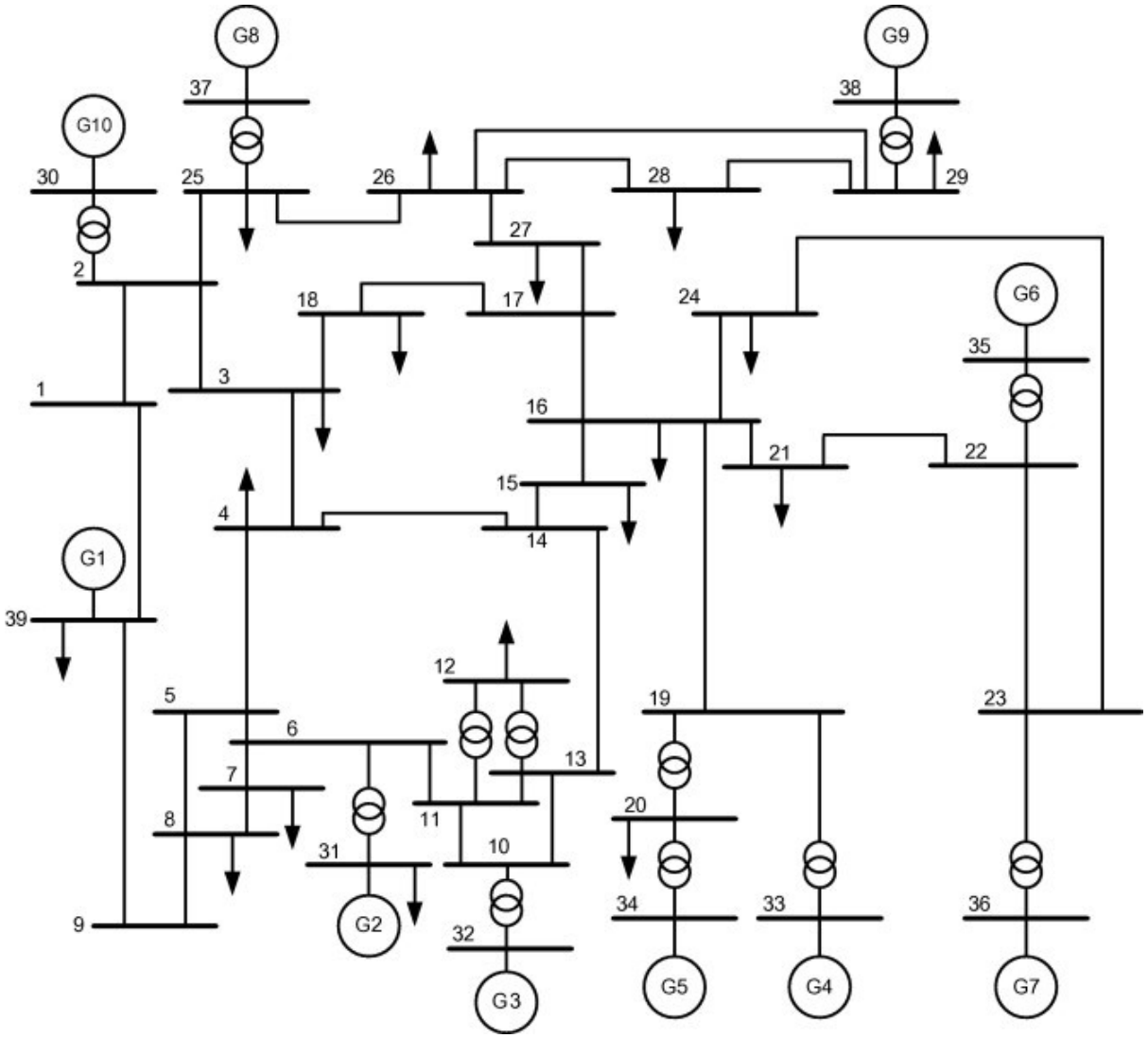

Figure 4-1 IEEE 39-Bus Power System

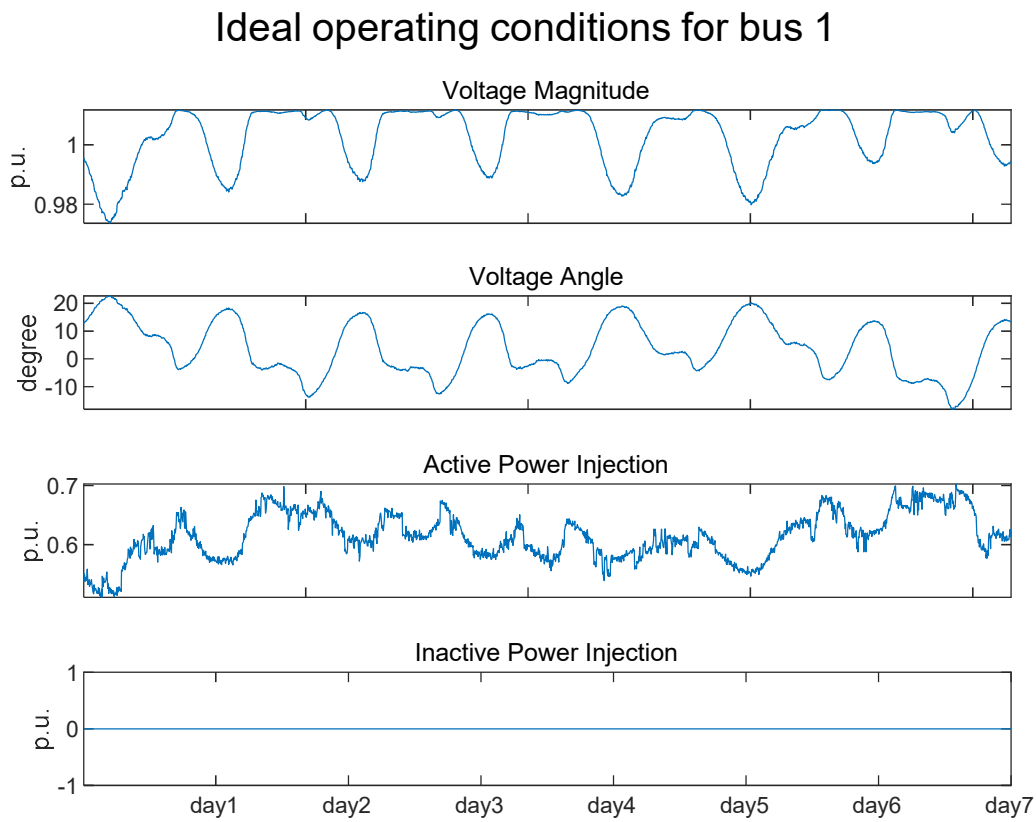

Figure 4-2 Ideal Operating Conditions for Bus 1

4.2 Analysis of Data Set

The strength of the attack vectors depends on the value of the state change $\mathbf{c}$. Random values of $\mathbf{c}$ can cover various conditions of malicious data injection attacks, which can successfully bypass the traditional residual detection mechanism. After the test system being injected into the aforementioned normal targeted attacks and playback targeted attacks respectively with $k/n = 50\%$, histograms of the measurements of the test system before and after attacks are compared together, as shown in Figure 4-3 and Figure 4-4.

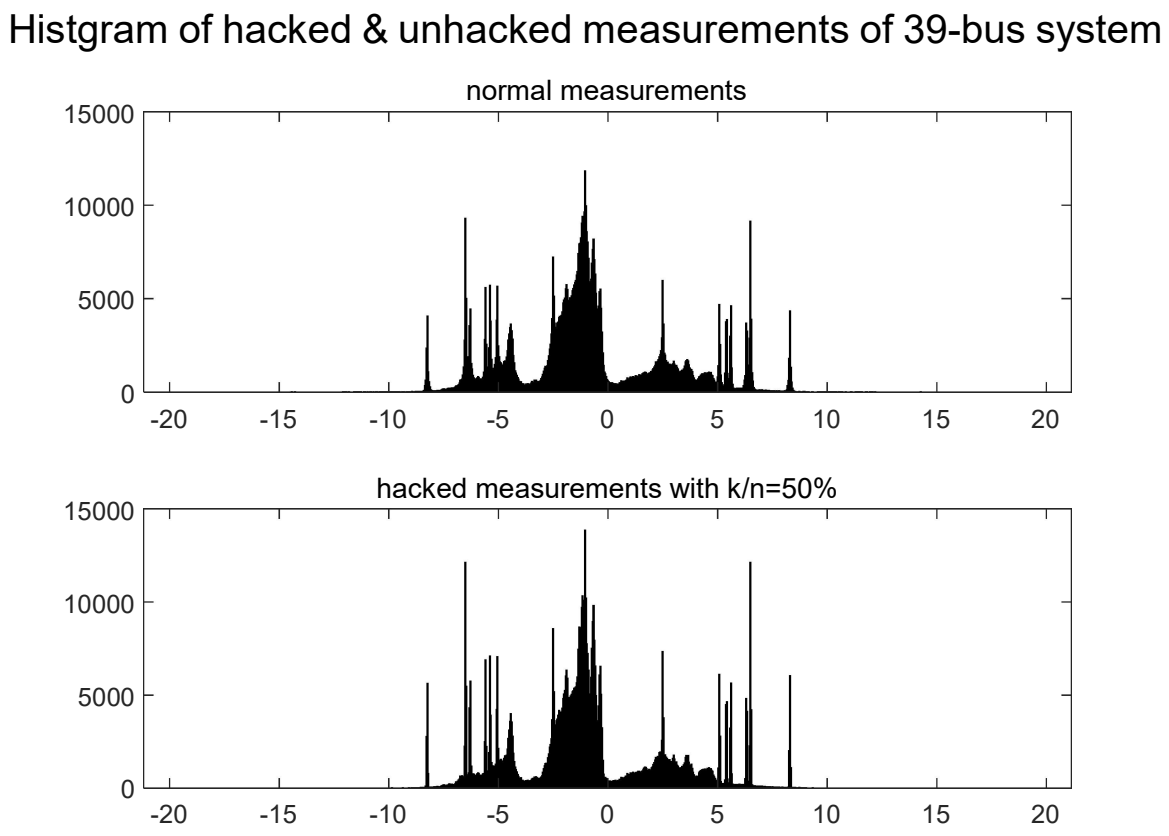

Figure 4-3 Histograms of hacked and unhacked measurements of IEEE 39 bus system by normal targeted attacks

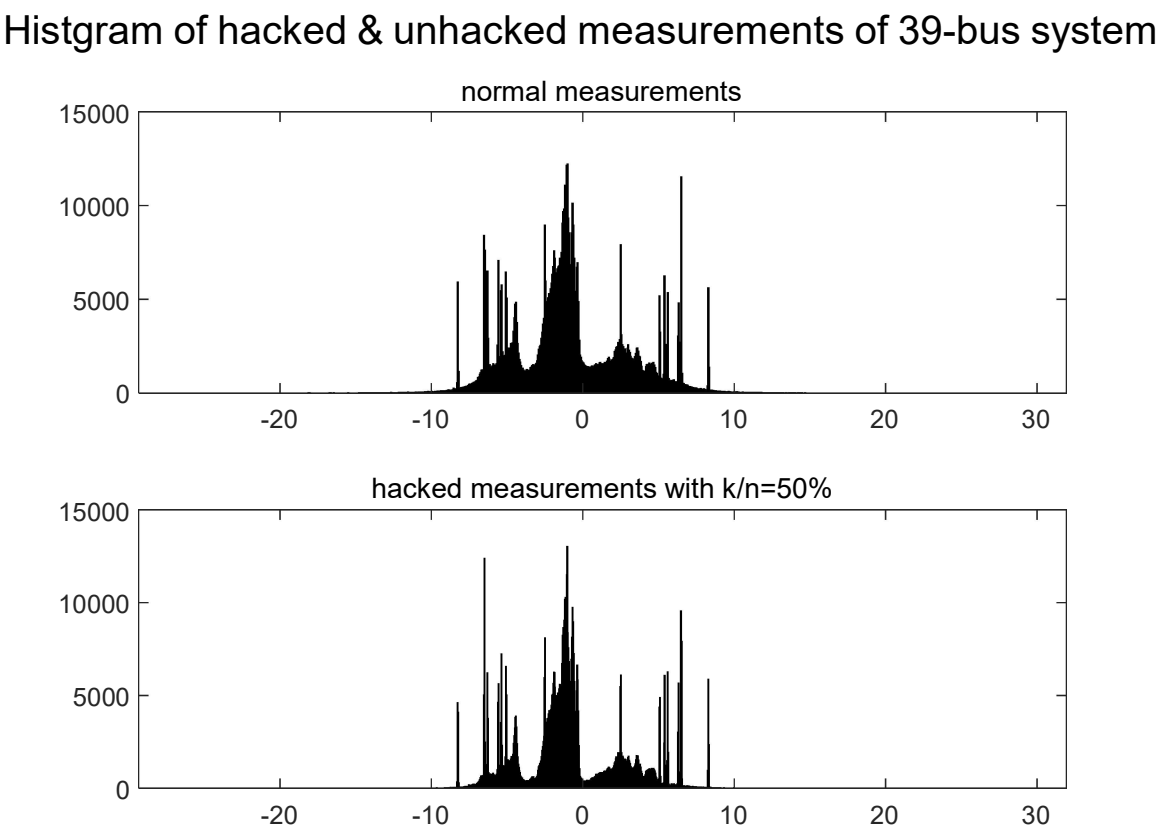

Figure 4-4 Histograms of hacked and unhacked measurements of IEEE 39 bus system by playback targeted attacks

It can be seen from Figure 4-3 and Figure 4-4 that after stealthy attacks were injected into the test system, the probability distribution of the measurements changed, but the change was slight. Therefore, it is difficult to detect malicious data injection attacks through statistic methods. Later the proposed detector is tested for such attacks.

4.1 Prediction

70% of ideal baseline data, 6364 samples, were spilled to train the predictor, and the remaining 30% was used as a test dataset. After being trained for three epochs, the MSE of prediction reached 0.0071 on the test dataset, and the training time was 39s, and the test time was 0.68s.

Next, the predicted data were used to train and test discriminators together with the attacked dataset. The performance of discriminators based on such a predictor was evaluated considering both mixed attack levels and each attack level, respectively.

4.2 Detection

Conventionally, samples are separated according to their Euclidean distance and a threshold. In this section, three ANN-based discriminators were compared with such a Euclidean-distance-based approach, as well as three other common classifiers in machine learning: Random Forest, K-Nearest Neighbors (KNN), and support vector machine (SVM). It is worth noting that three non-ANN-based classifiers take in only attacked data, which means they have nothing to do with the predictor. The ratio between a training set and a test set was still 70%/30%, and attacks were from different attack levels uniformly.

4.2.1 Normal Targeted Attacks

The ANN-based discriminators were trained for three epochs. With the classification threshold being determined by the algorithm in Chapter 3, the metrics on the test set are shown in Table 4-1. The ROC curves are shown in Figure 4-5 to Figure 4-11.

As can be seen from these figures and the table, the ANN-based discriminators had a robust performance compared to other methods, especially achieved both high Precision and Recall. The top three were occupied by them in ROC-AUC and top four in F1-Score, with acceptable training and detection time. Although SVM's F1-Score exceeded convolution-based and squared-error-vector-based approaches, the time required by SVM in the training phase was

almost 43 times the average time 17.12s of the neural network classifiers, which was the longest training time among all classifiers. Similarly, the detection time required by the KNN was almost 209 times the average value 1.87s of the neural network classifiers, which was the longest detection time among all classifiers. Random Forest was the one with the shortest detection time, but its ROC-AUC was the lowest. The method based on Euclidean distance had the shortest training and detection time in total because it did not require training, but its F1-Score was the lowest.

At last, even if the time consumption caused by the predictor had been added, the proposed detector would be still tempting. However, the discriminator and the predictor do not have to be trained together. On the whole, if time cost is important, it is recommended to choose the Euclidean-distance-based method; if time is sort of ample, the discriminators based on neural networks demonstrate the best performance.

	ROC-AUC	**Precision**	**Recall**	**F1-Score**	**Training Time/s**	**Test Time/s**
Concatenation	0.9877	0.9945	0.9542	0.9739	11.94	1.39
Convolution	0.9922	0.9856	0.9690	0.9773	25.22	2.58
Squared Error Vector	0.9962	0.9976	0.9715	0.9844	14.22	1.64
Euclidean Distance	0.9798	0.9419	0.9215	0.9316	0	1.39
Random Forest	0.9202	0.8982	0.9904	0.9420	20.27	0.10
KNN	0.9465	0.9979	0.8956	0.9440	2.86	390.16
SVM	0.9787	0.9947	0.9643	0.9792	729.43	21.18

Table 4-1 Detection using Various Approach under normal targeted attacks with mixed levels

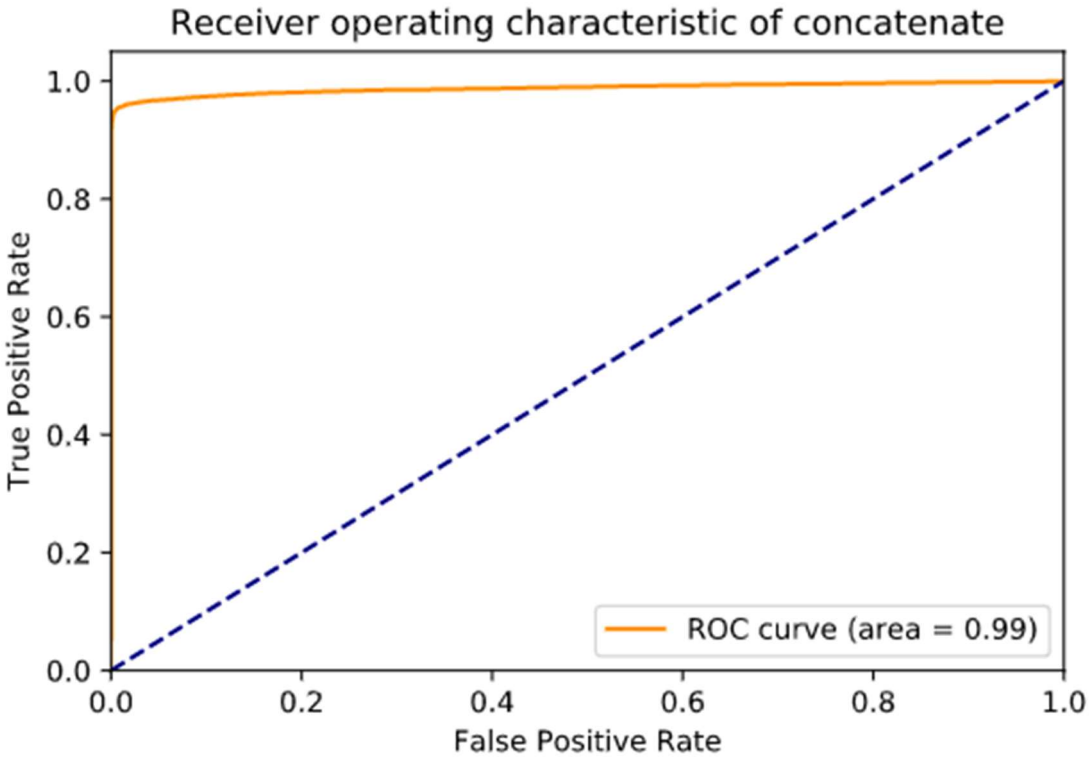

Figure 4-5 ROC Curve for concatenation discriminator under normal targeted attacks

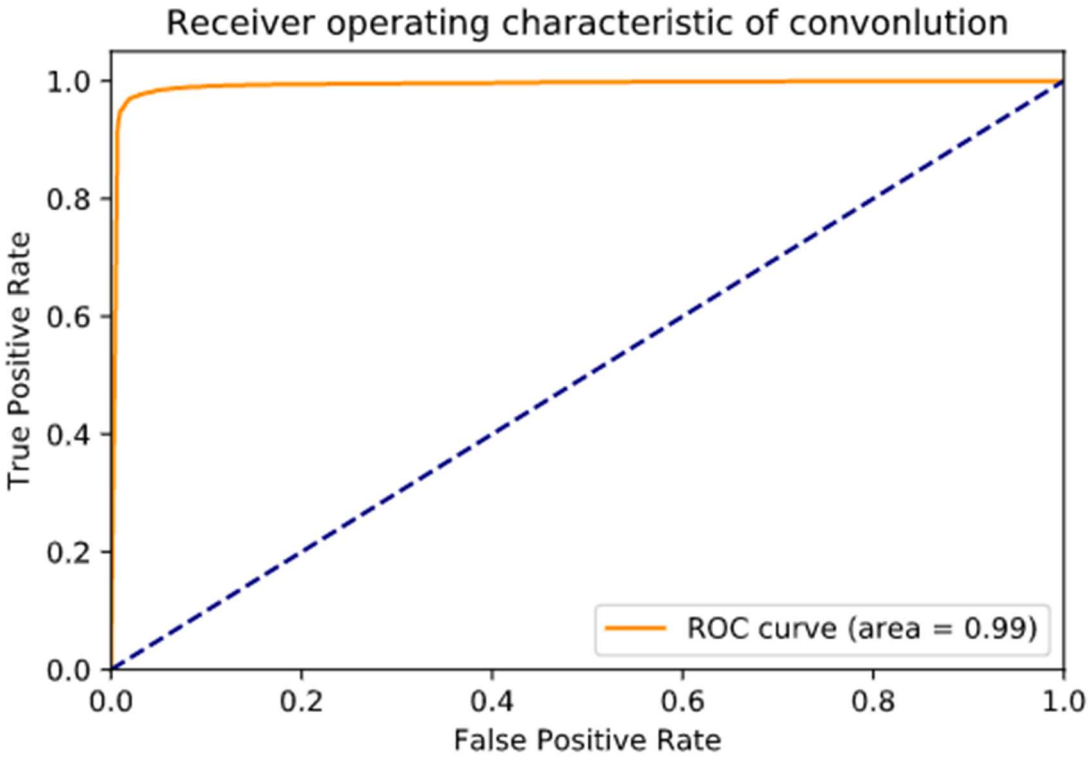

Figure 4-6 ROC Curve for convolution discriminator under normal targeted attacks

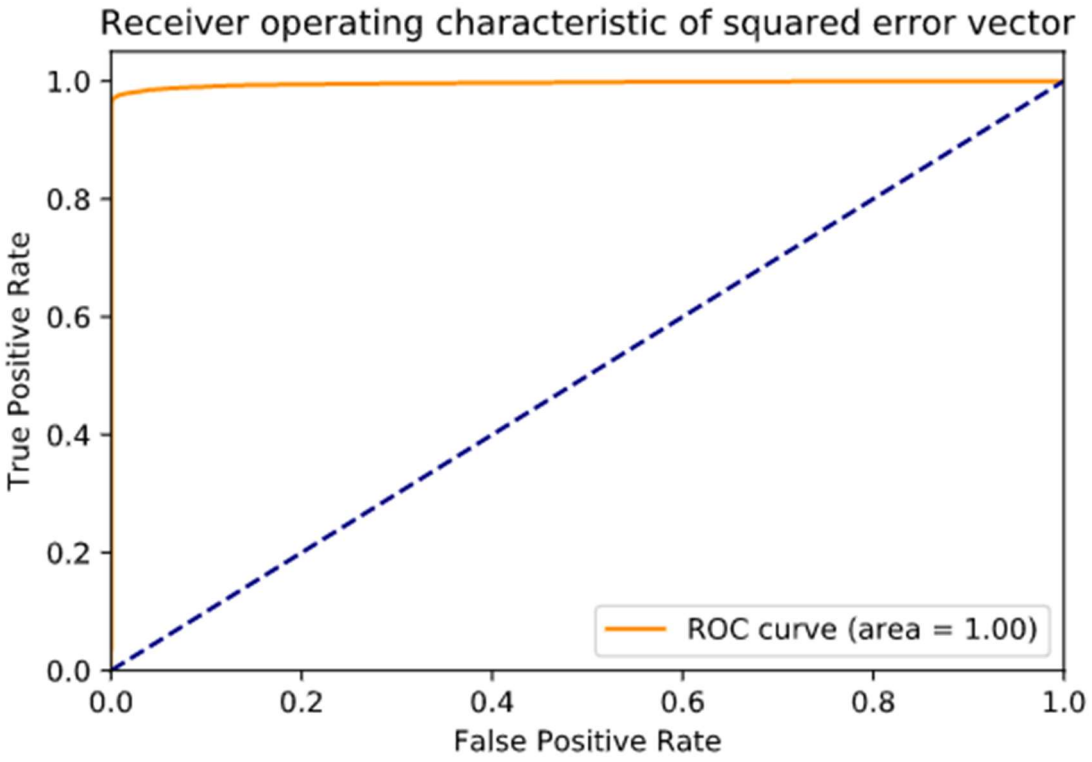

Figure 4-7 ROC Curve for Squared error vector discriminator under normal targeted attacks

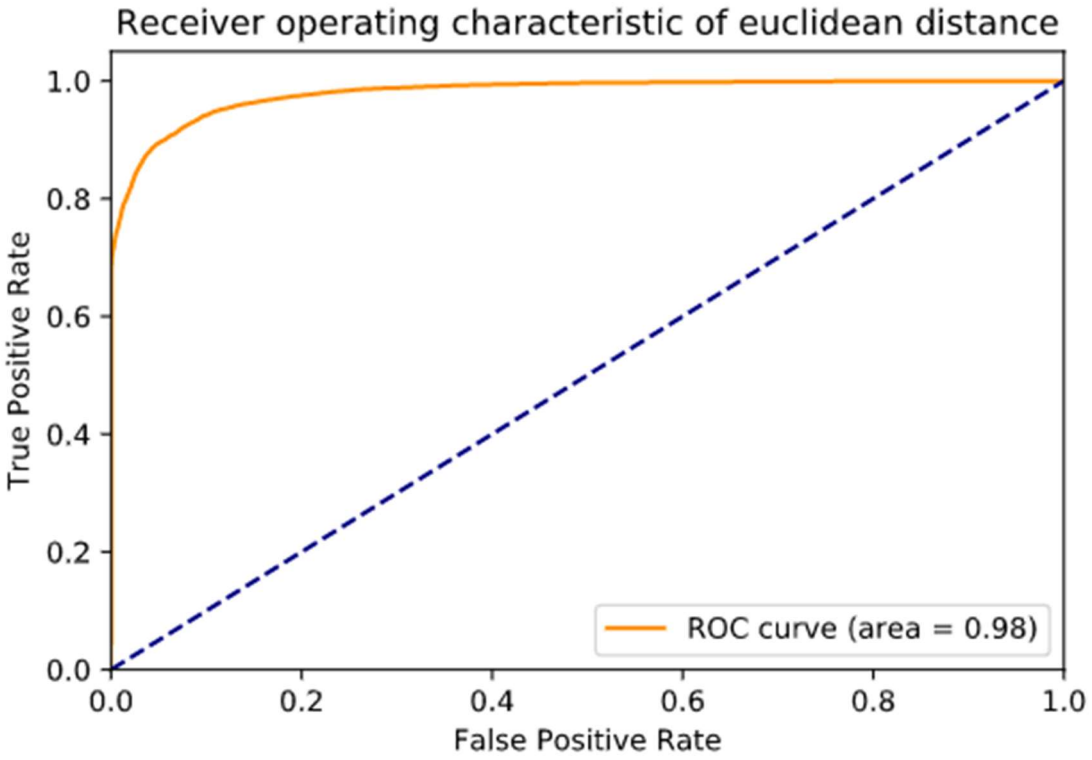

Figure 4-8 ROC Curve for Euclidean distance method under normal targeted attacks

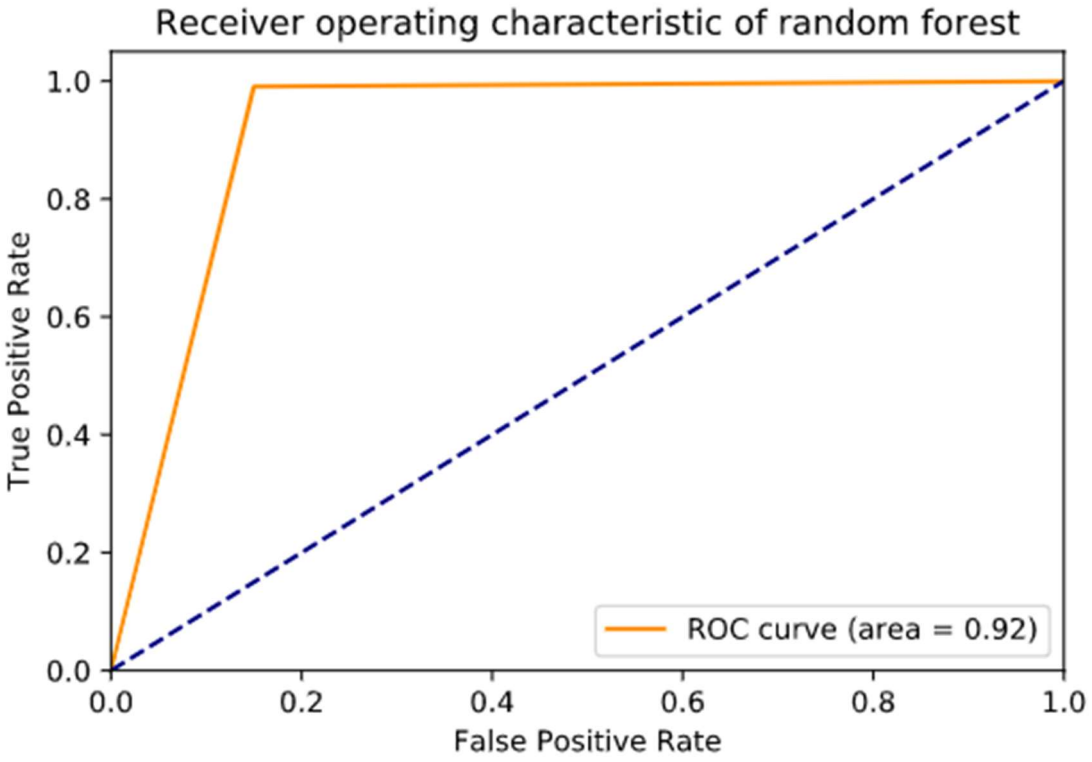

Figure 4-9 ROC Curve for random forest classifier under normal targeted attacks

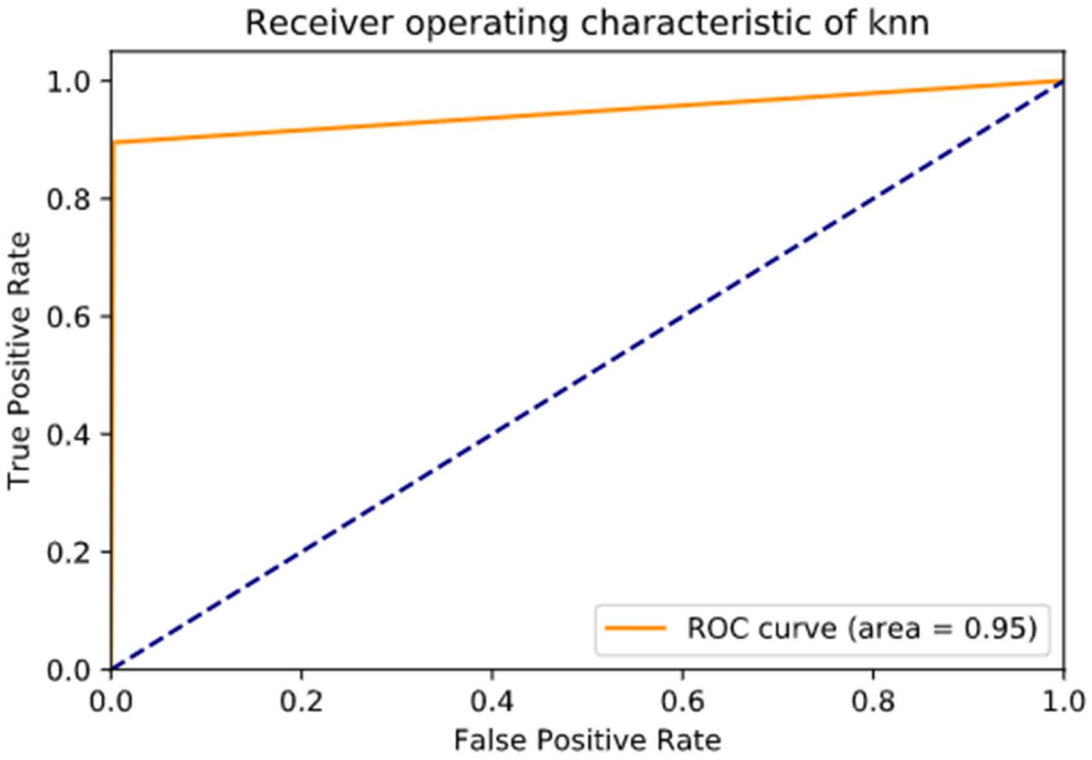

Figure 4-10 ROC Curve for KNN classifier under normal targeted attacks

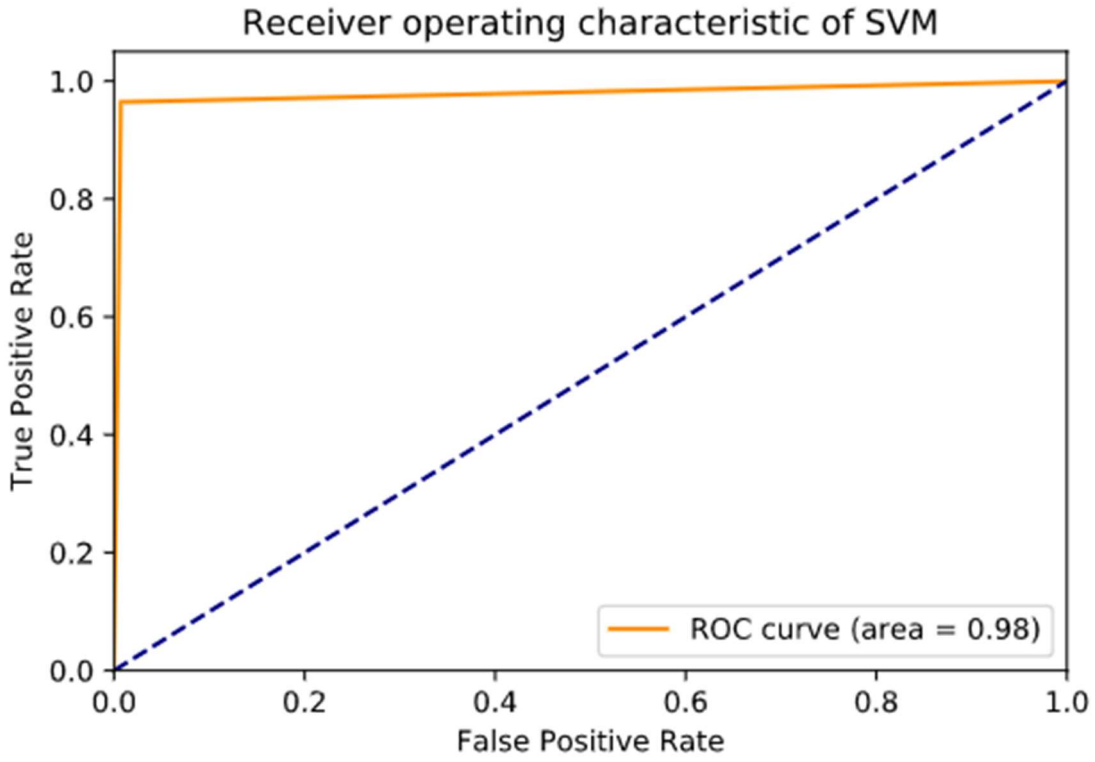

Figure 4-11 ROC Curve for SVM classifier under normal targeted attacks

To observe the performance of discriminators under attacks of different levels, they were tested on different attack levels, and the results are shown in Figure 4-12 to Figure 4-14. All classifiers had a lower performance when the attack level was low because the low proportion of tampered state variables changed measurement data too slightly to be detected, or the change was even drowned in the noise.

From Figure 4-12, KNN and squared-error-vector-based discriminator ranked first in Precision. SVM and the concatenation-based discriminator ranked second. The convolution-based discriminator was at the third level above 0.98. The Precision for the Euclidean-distance-based method changed dramatically from about 0.89 to around 0.94. Random Forest was the last never more than 0.90.

From Figure 4-13, it can be seen that difference occurred at low attack levels. Even though Random Forest achieved the highest Recall, it paid a great price that the Precision was the lowest. To the contrast, KNN was the last due to its high Precision. The ANN-based discriminators' Recall was within 0.6 and 0.8 when $k/n = 0.1$ and then went up to 0.90 or above when $k/n > 0.1$, among which the squared-error-vector-based one was the best. The SVM joined the group of the ANN-based discriminators because of similar performance.

From Figure 4-14, with the tradeoff between Precision and Recall, the ANN-based discriminators still performed well at all attack levels in F1-Score. Among them, the squared-error-vector-based discriminator was the best, the convolution-based one was the second, and the concatenation-based one was the last. When k/n was greater than 0.4, their F1-Score approached close to 1. Although when $k/n = 0.1$, Random Forest achieved the best value of 0.9, its F1-score failed to exceed 0.95 when $k/n > 0.2$. Also, F1-Score of SVM exceeded the concatenation-based discriminator when k/n was less than 0.4.

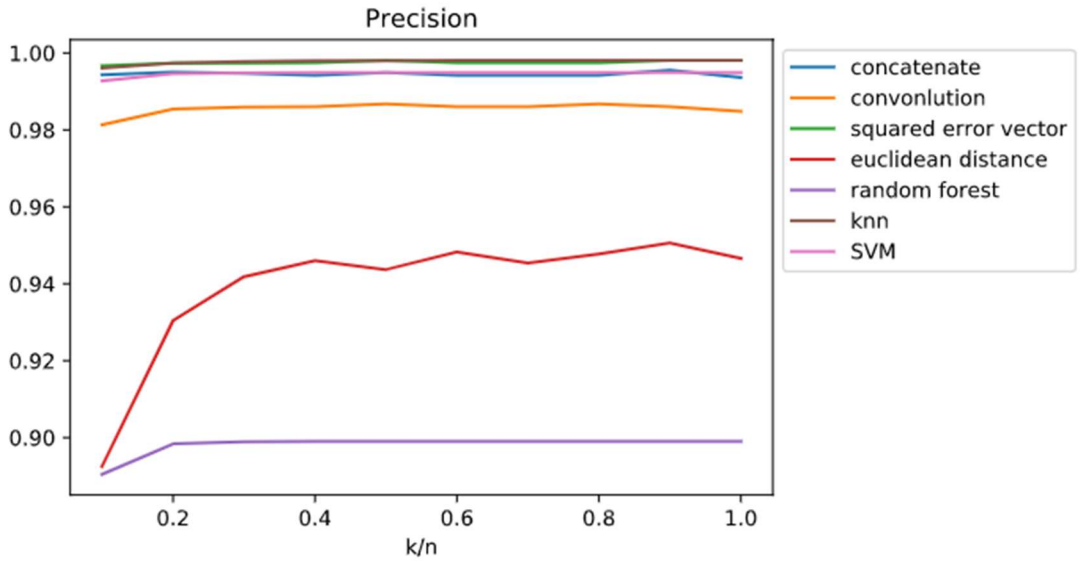

Figure 4-12 Precision for various classifiers under normal targeted attacks with different levels

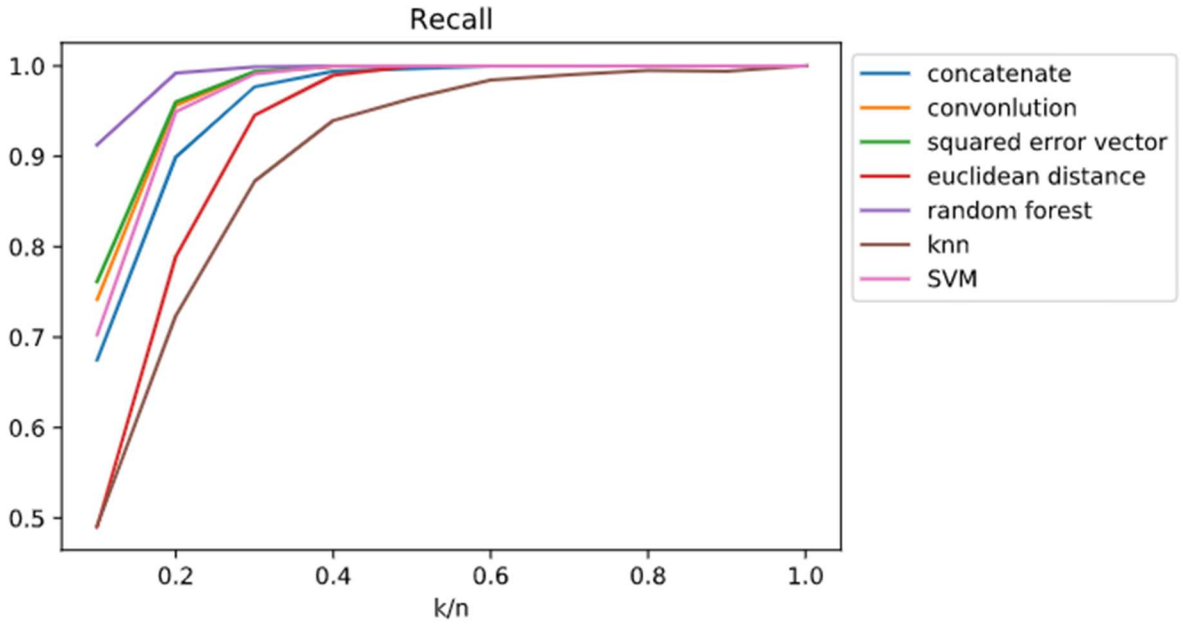

Figure 4-13 Recall for various classifiers under normal targeted attacks with different levels

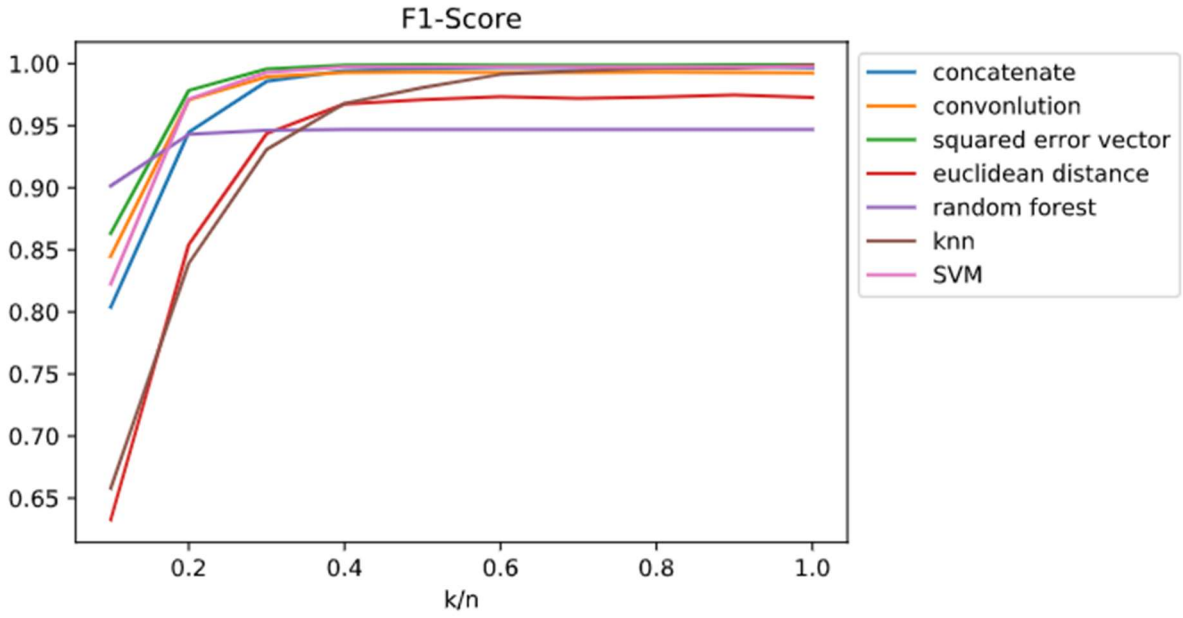

Figure 4-14 F1-Score for various classifiers under normal targeted attacks with different levels

4.2.2 Playback Targeted Attacks

Playback targeted attacks were from real load data 15 minutes ago, i.e., a delay of 3 time steps. The performance of the proposed discriminators was analyzed under different attack levels. The discriminators based on the neural network were trained three epochs. The classification threshold was determined according to the algorithm in Chapter 3, and the metrics on the test dataset are shown in Table 4-2. ROC curves are shown from Figure 4-15 to Figure 4-21. As can be seen from Table 4-2, compared with the normal targeted attack scenario, all classifiers' performance was improved. The ANN-based classifiers were top three in the comprehensive metrics such as ROC-AUC and F1-Score, where particularly the convolution-based discriminator was the champion. All ANN-based discriminators kept high in both Precision and Recall while the other methods not.

	ROC-AUC	**Precision**	**Recall**	**F1-Score**	**Training Time/s**	**Test Time/s**
Concatenation	0.9964	0.9987	0.9742	0.9863	15.60	1.70
Convolution	0.9981	0.9987	0.9841	0.9913	28.97	2.79
Squared Error Vector	0.9980	0.9967	0.9827	0.9897	15.83	1.74
Euclidean Distance	0.9638	0.9747	0.8601	0.9138	0	1.71
Random Forest	0.9635	0.9559	0.9879	0.9716	11.56	0.08
KNN	0.9694	0.9906	0.9508	0.9703	3.26	441.11
SVM	0.9455	1.0	0.8910	0.9424	1671.35	45.44

Table 4-2 Detection using Various Approaches under playback targeted attacks with mixed levels

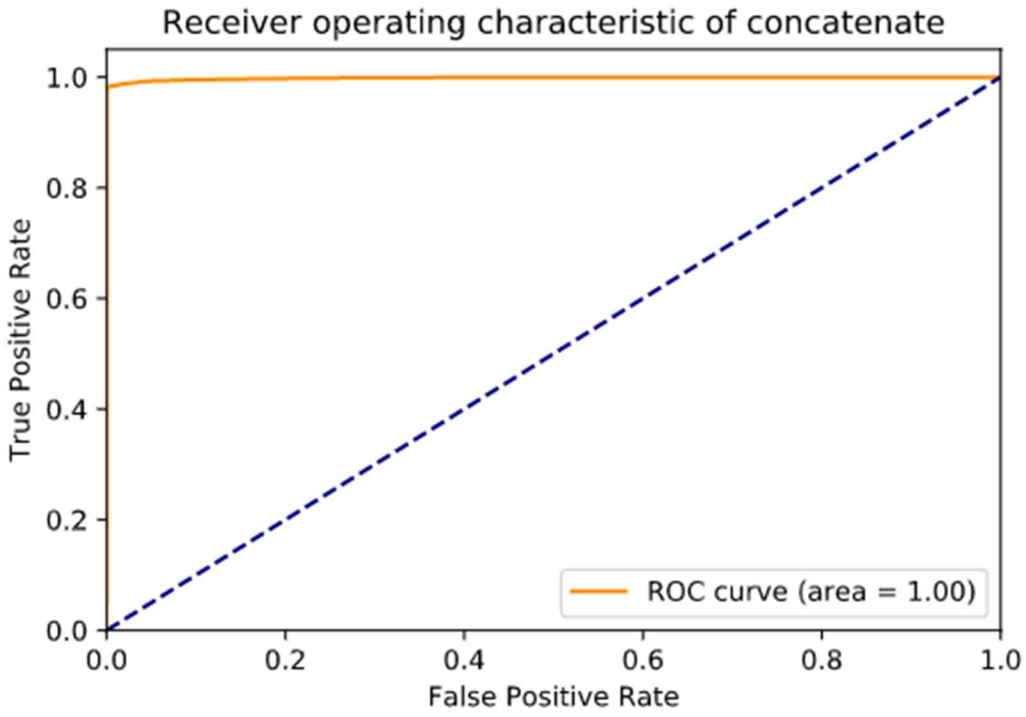

Figure 4-15 ROC Curve for concatenation discriminator under playback targeted attacks

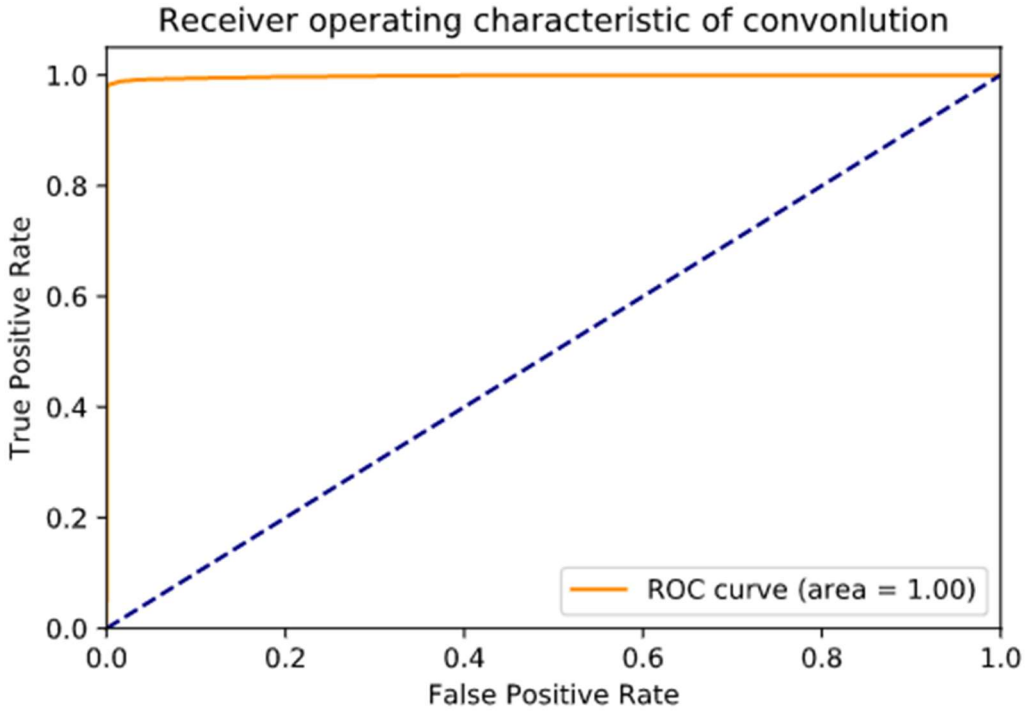

Figure 4-16 ROC Curve for convolution discriminator under playback targeted attacks

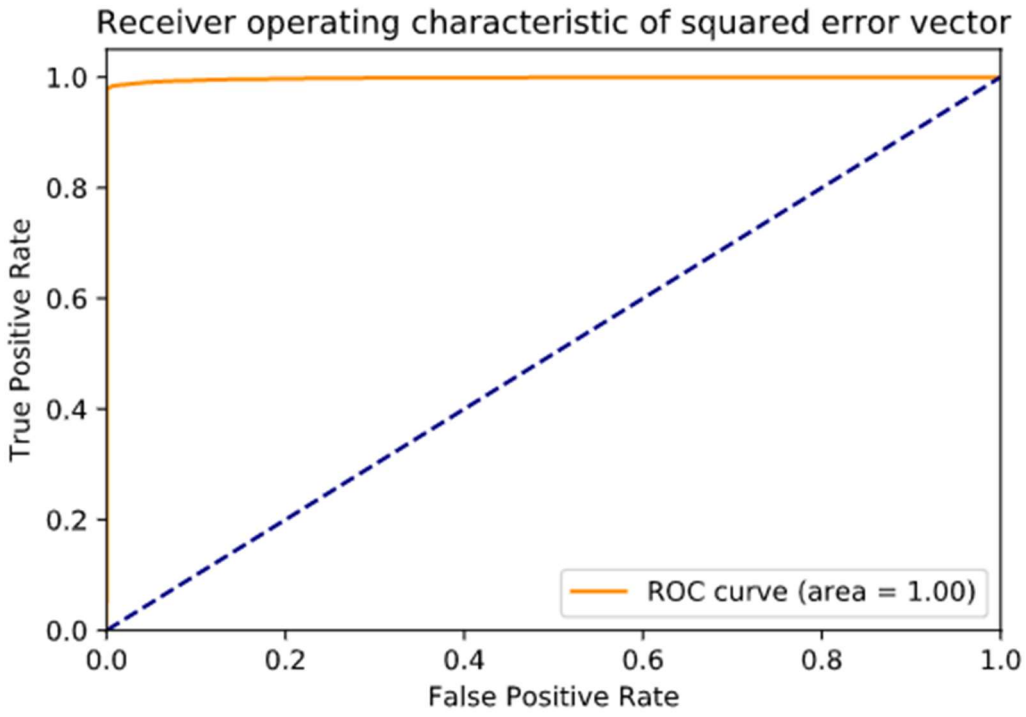

Figure 4-17 ROC Curve for Squared error vector discriminator under playback targeted attacks

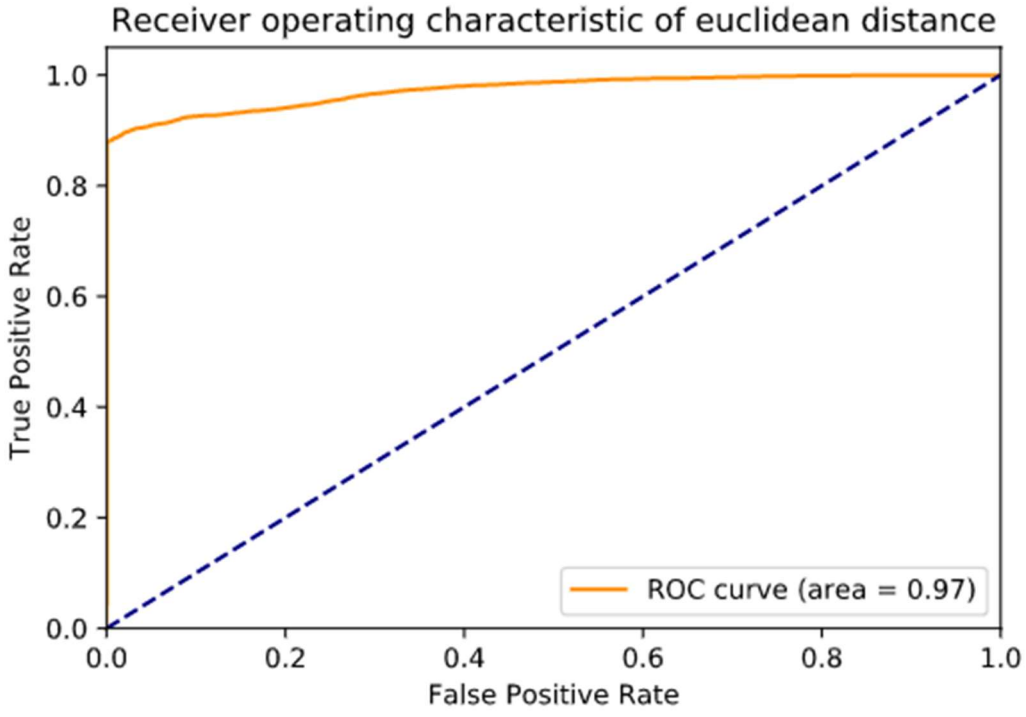

Figure 4-18 ROC Curve for Euclidean distance method under playback targeted attacks

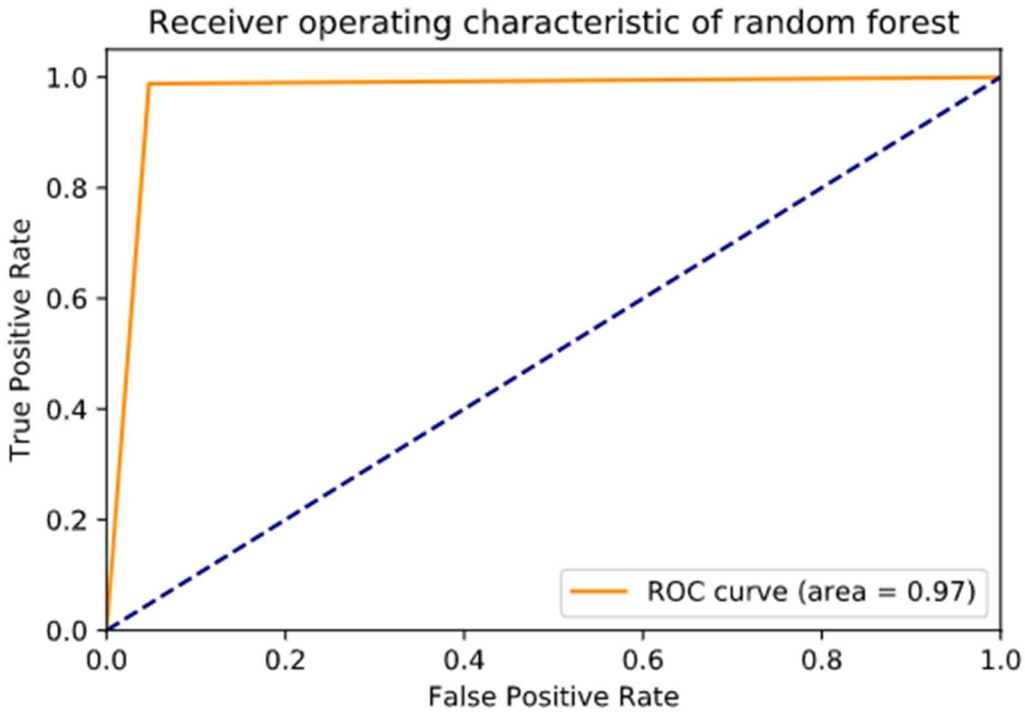

Figure 4-19 ROC Curve for random Forest under playback targeted attacks

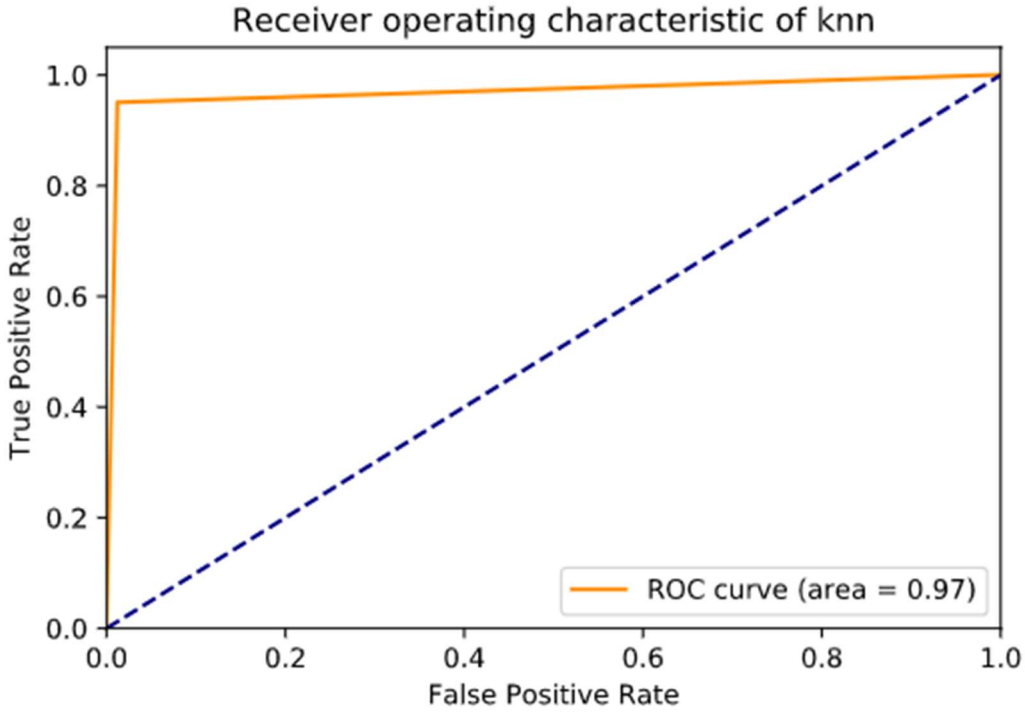

Figure 4-20 ROC Curve for KNN under playback targeted attacks

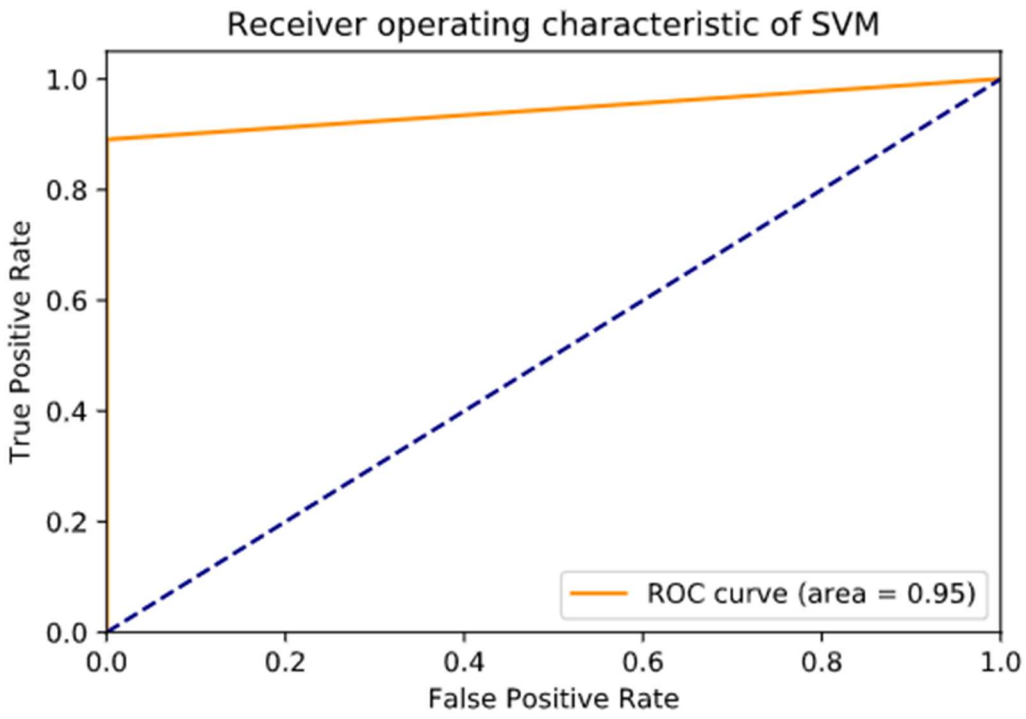

Figure 4-21 ROC Curve for SVM under playback targeted attacks

Figure 4-22 to Figure 4-24 shows the performance of different classifiers under different attack levels separately. It can be seen that all classifiers performed better than on the normal targeted attacks dataset. Similar to the scenario of normal targeted attacks, the ANN-based discriminators were still the best at any level of attacks and demonstrated a perfect balance between Precision and Recall. Particularly, the convolution-based discriminator was the best.

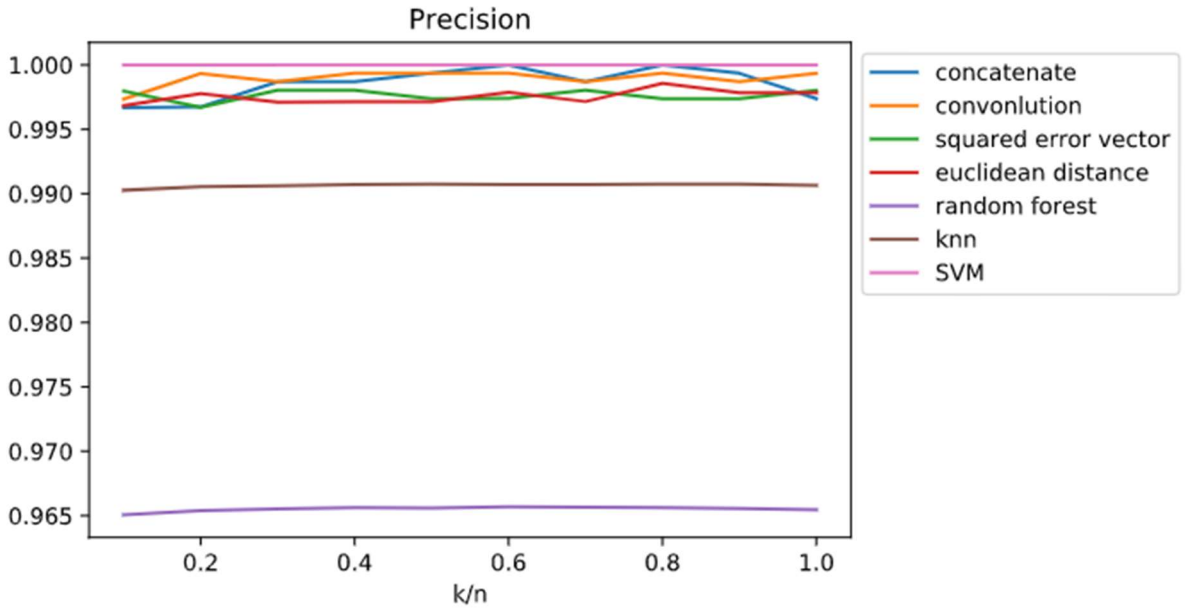

Figure 4-22 Precision for various classifiers under playback targeted attacks with different levels

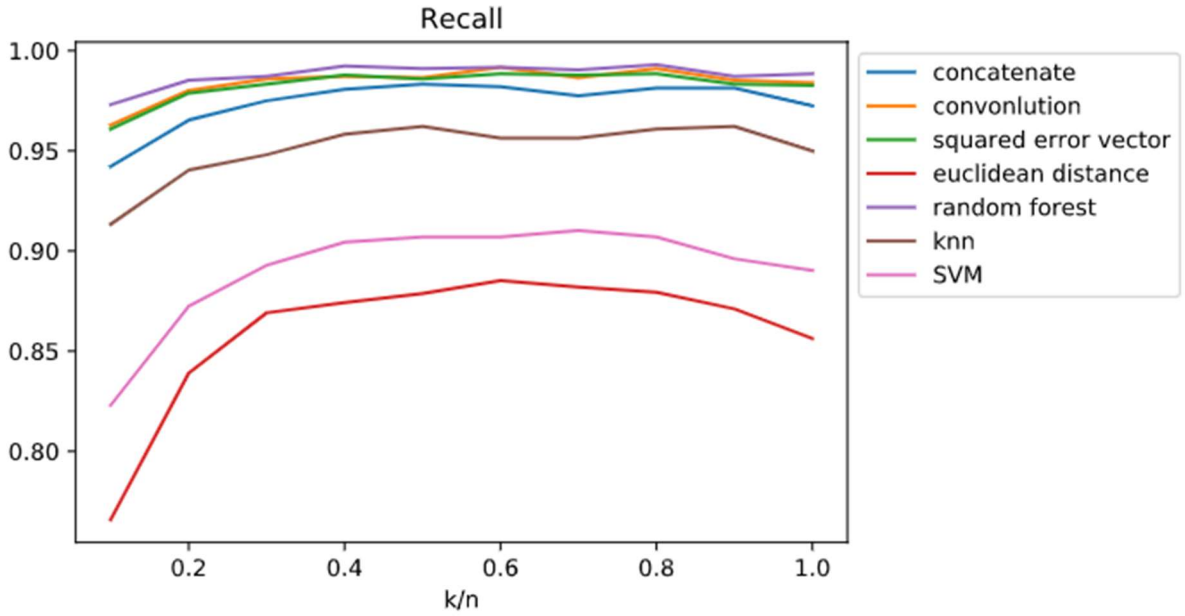

Figure 4-23 Recall for various classifiers under playback targeted attacks with different levels

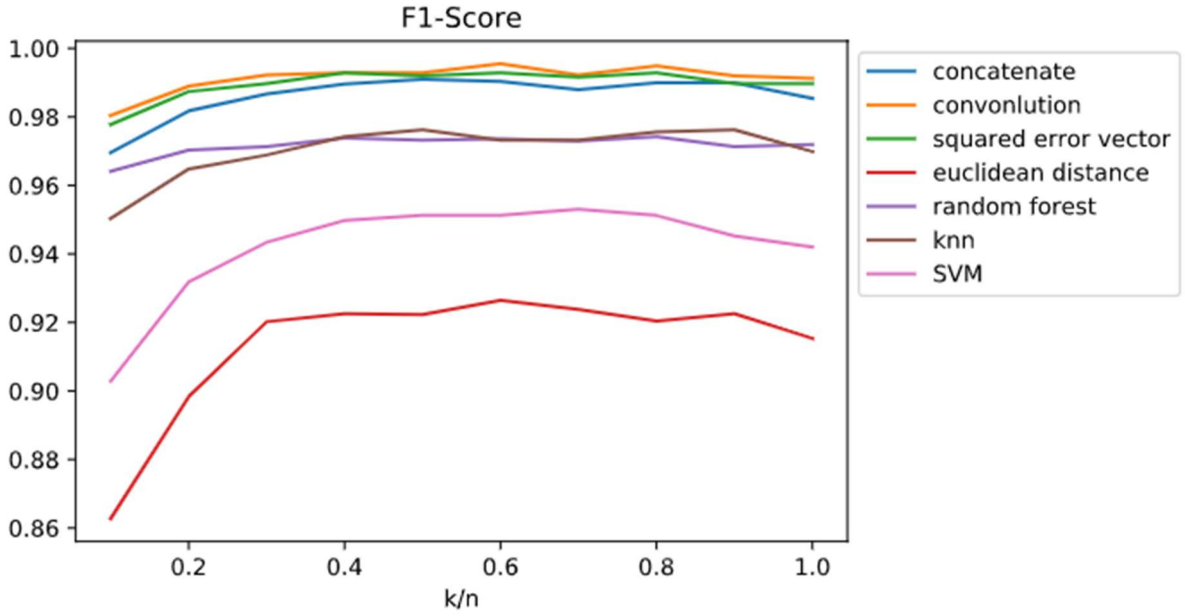

Figure 4-24 F1-Score for various classifiers under playback targeted attacks with different levels

4.2.3 Further experiment

It is found that ANN-based discriminators could also achieve satisfactory results even if they were trained on the normal targeted attack dataset but tested on the playback targeted attack dataset, as shown in Table 4-1. Among them, discriminator that used squared error vectors as features had the highest ROC-AUC and F1-Score. It also shows that playback targeted attacks can be regarded as a subset of normal targeted attacks with random changes on state variables. Thus, only the latter one needs to be focused in future research.

	ROC-AUC	Precision	Recall	F1-Score
Concatenation	0.9955	0.9982	0.9745	0.9862
Convolution	0.9969	0.9964	0.9854	0.9858
Squared Error Vector	0.9973	0.9975	0.9790	0.9881

Table 4-3 Detection using discriminators when the dataset is changed

4.2.4 Analysis of The Difference Between Precited Data and Actual Data

4.2.4.1 Normal Targeted Attacks

In order to explore the difference in discriminators' performance under different attack levels, this paper analyzed the distribution of the Euclidean distance between the predicted measurements and the actual measurements. Figure 4-25 shows the distance distribution under all attack levels. Figure 4-26 to Figure 4-35 show the distance distribution at different levels. From Figure 4-25, actual measurements deviated from predicted ones in a wide range of about 0.5 to 5 in attacked cases, and in a small range of 0.5 to 1.5 in secure cases because of the presence of prediction errors. Because of that, the distributions in the two cases overlapped, making it difficult for various classifiers to make accurate judgments. However, because of the powerful learning ability of neural networks, data in the overlap can be distinguished as much as possible by proposed discriminators compared with other non-ANN -based classifiers.

It has been found from the distribution of distances under different attack levels that the lower k/n was, the more severe the overlapping was. When $k/n = 1.0$, data in the two cases were almost entirely separated, which is why all classifiers performed perfectly when k/n was high while not very well when k/n was low. In reality, an attacker is unlikely to control too many devices, which means that the attack level may not be very high, and attacks are difficult to detect. For this reason, ANN-based discriminators proposed in this paper are more recommended.

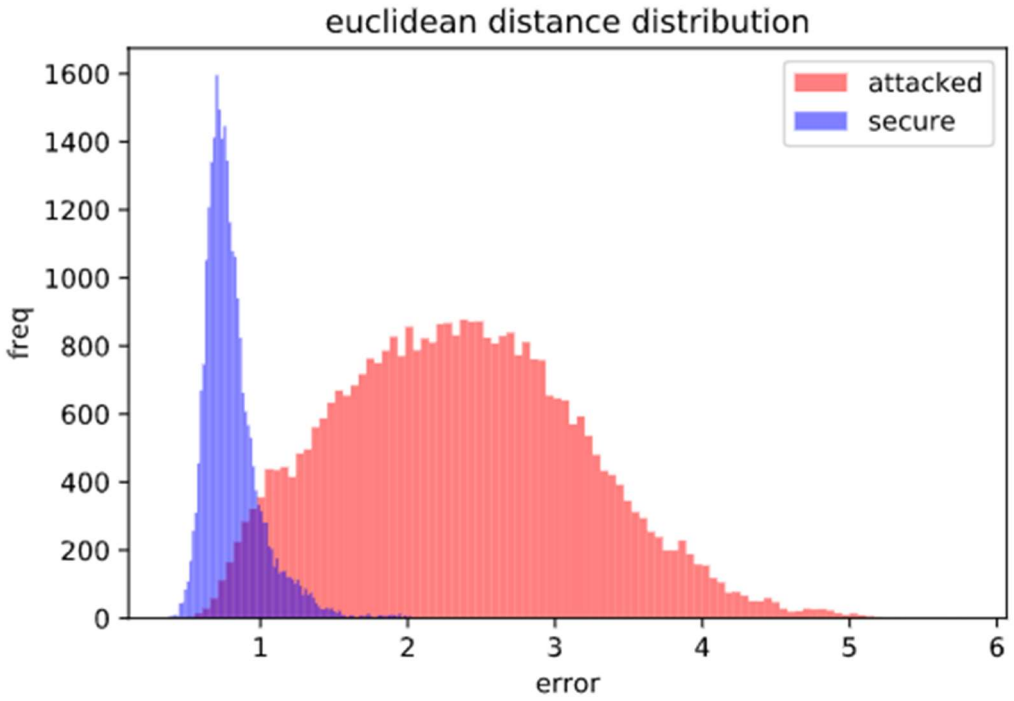

Figure 4-25 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks of mixed levels

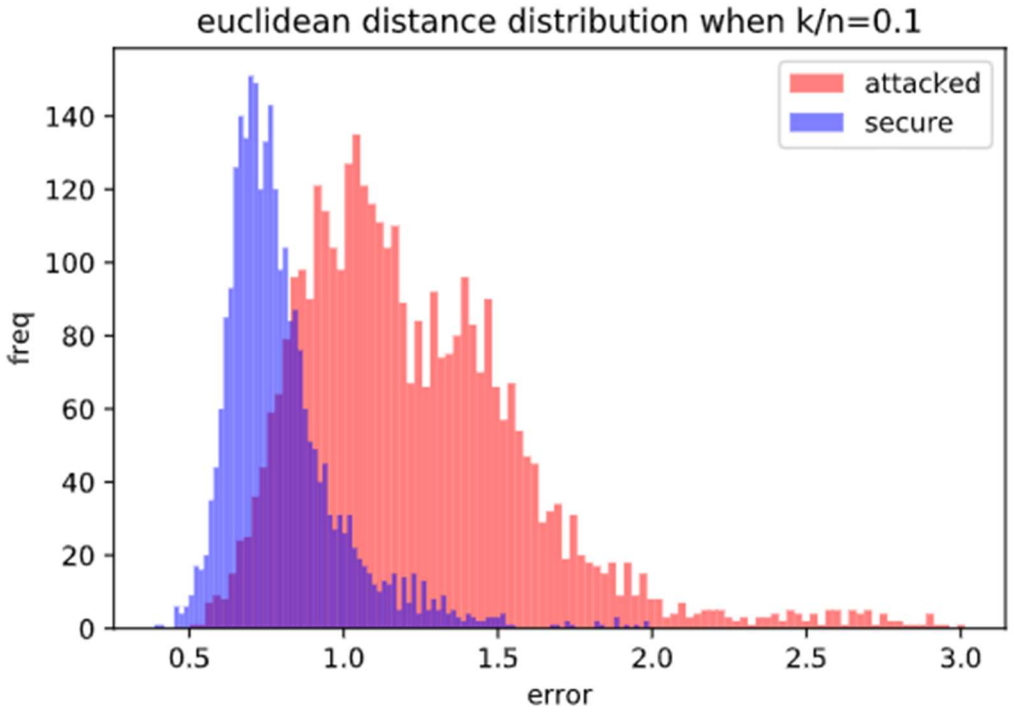

Figure 4-26 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.1

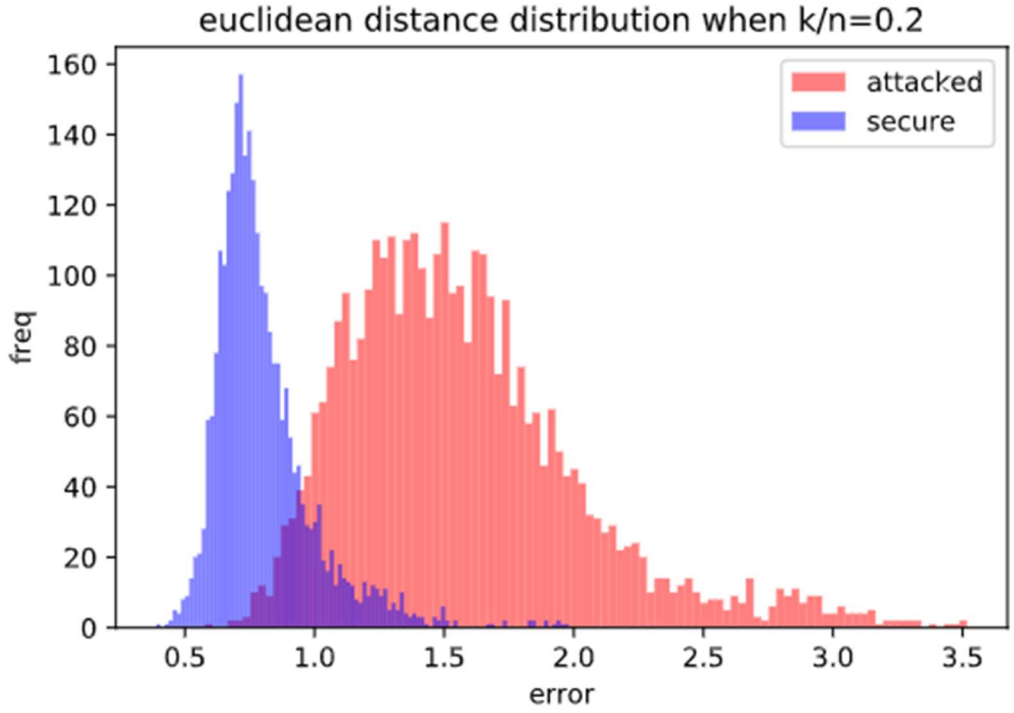

Figure 4-27 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.2

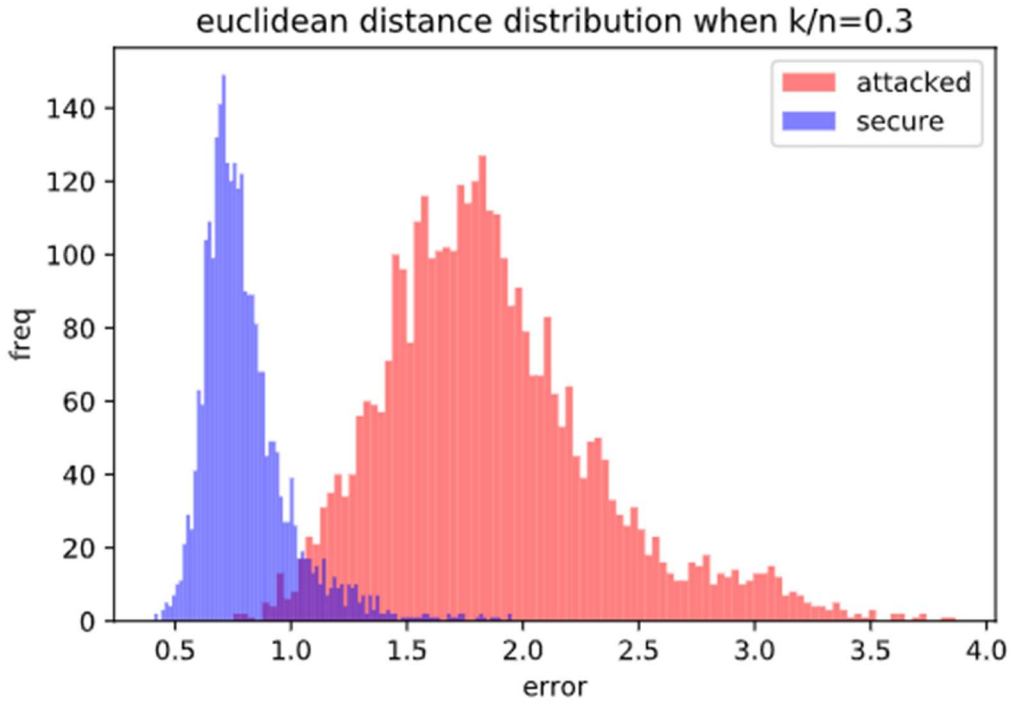

Figure 4-28 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.3

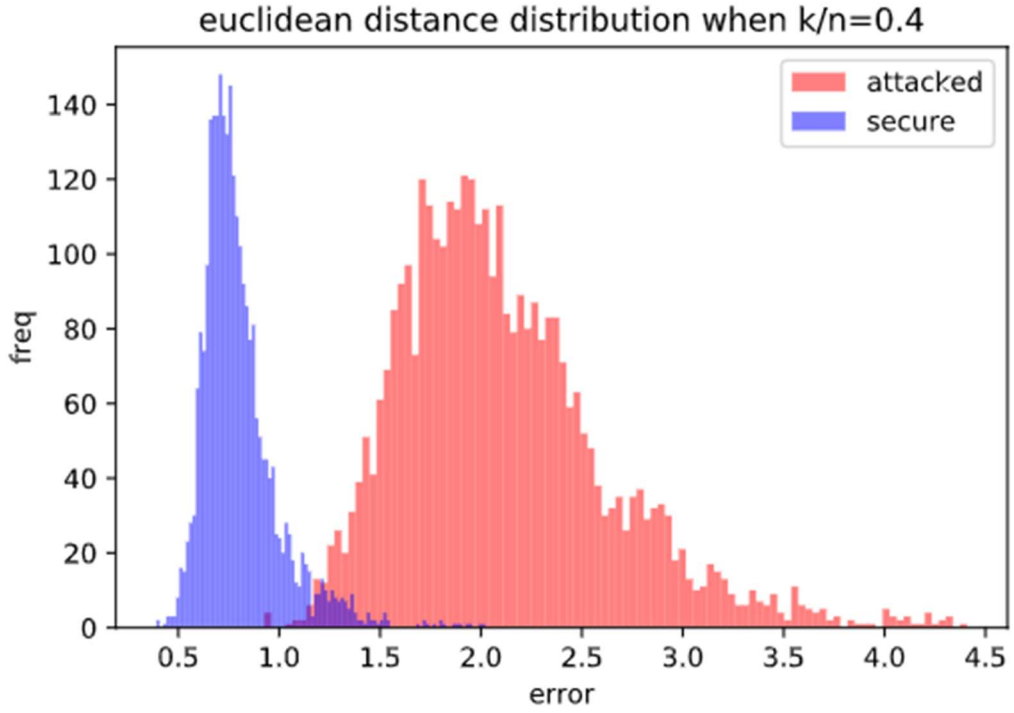

Figure 4-29 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.4

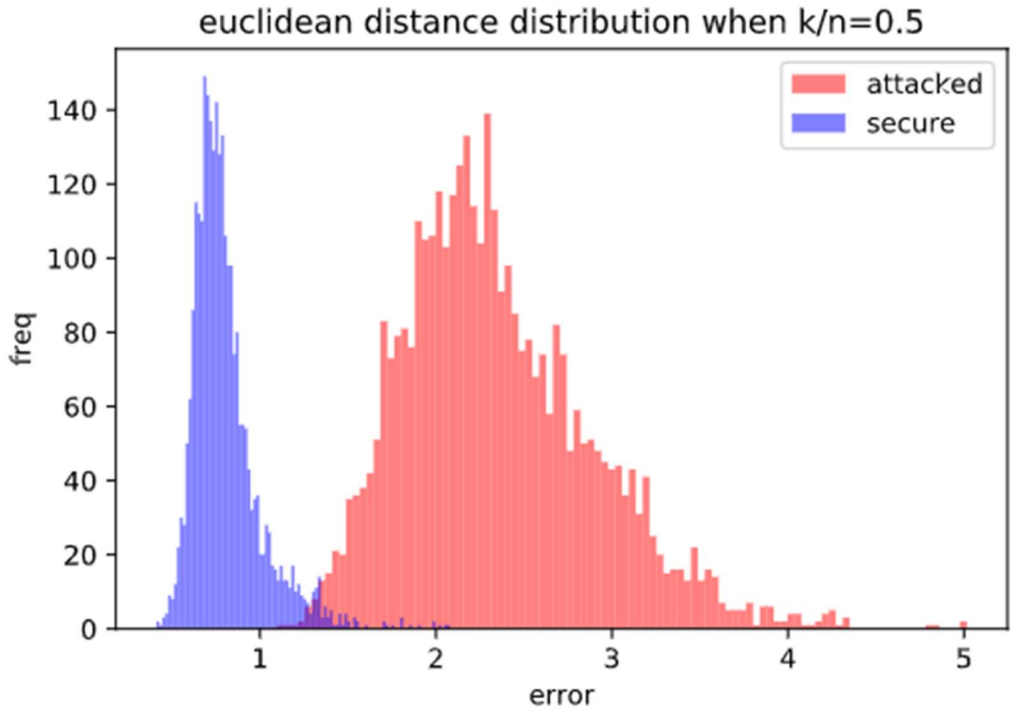

Figure 4-30 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.5

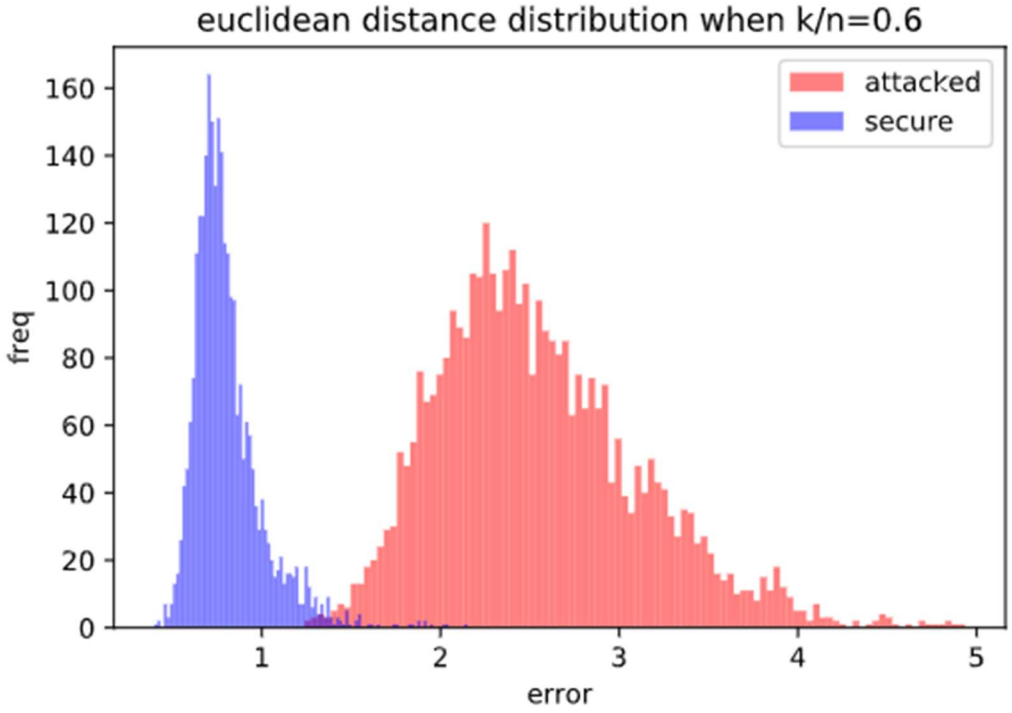

Figure 4-31 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.6

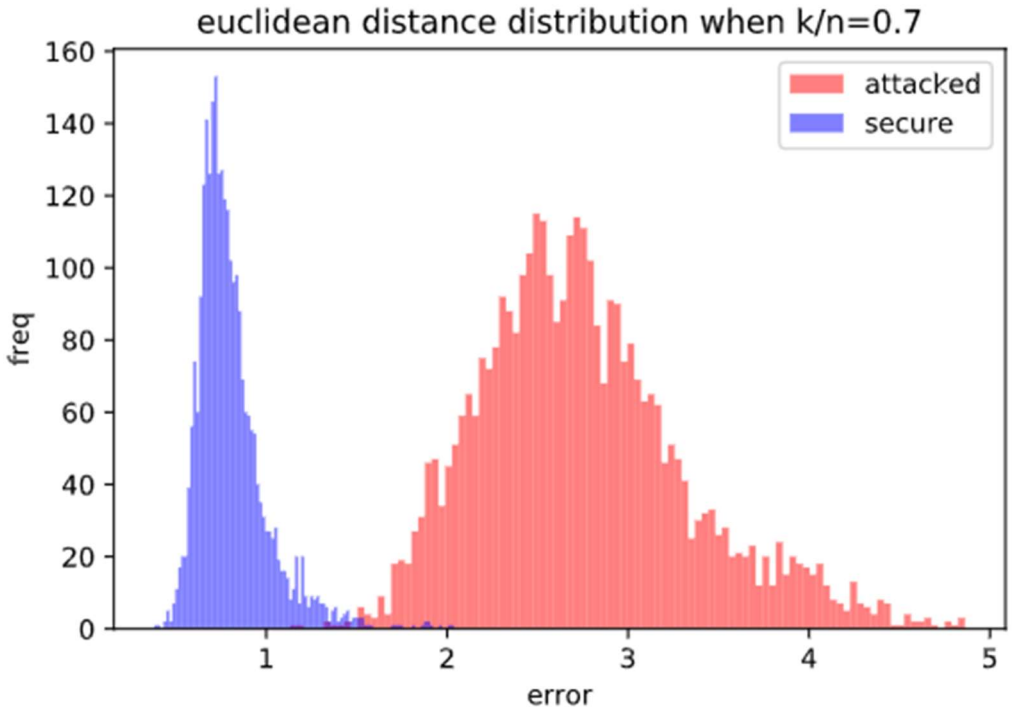

Figure 4-32 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.7

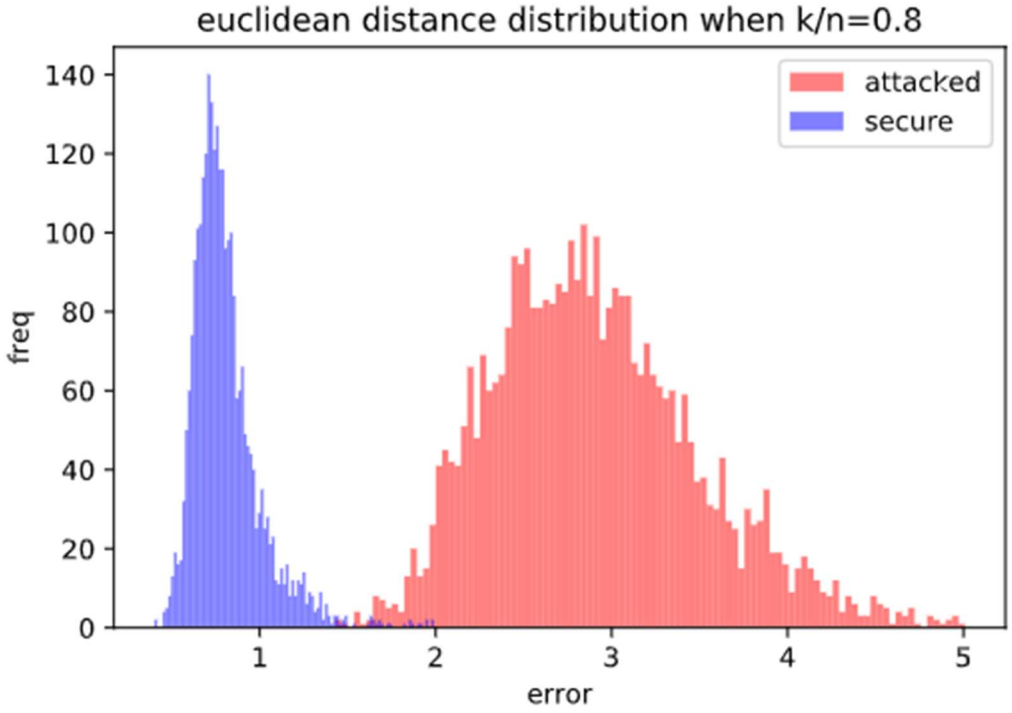

Figure 4-33 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.8

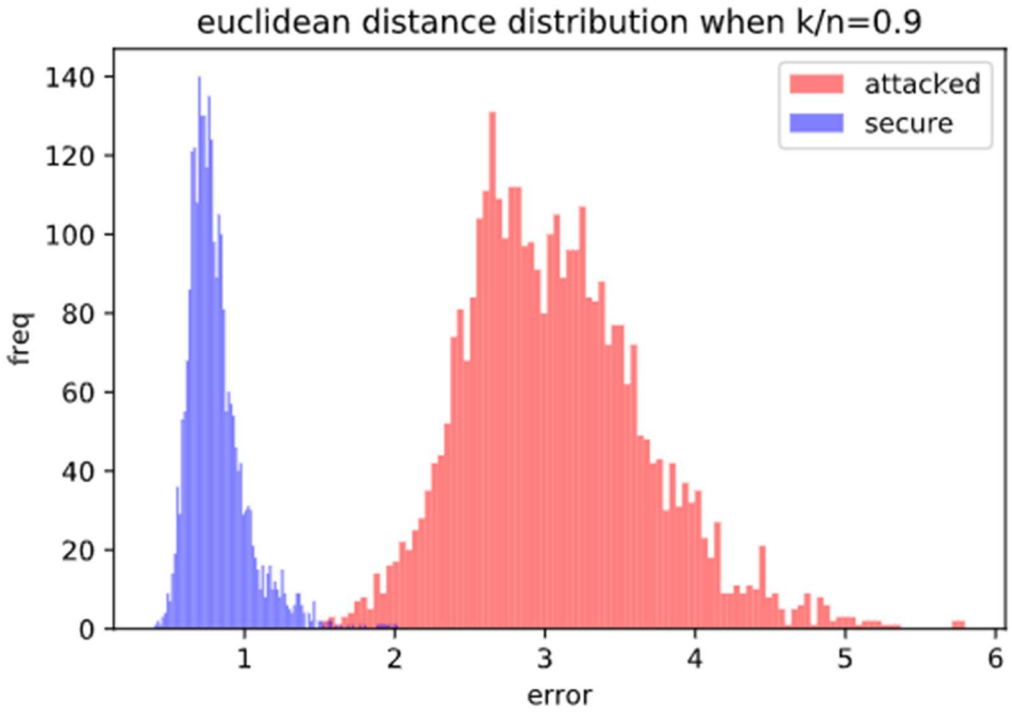

Figure 4-34 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=0.9

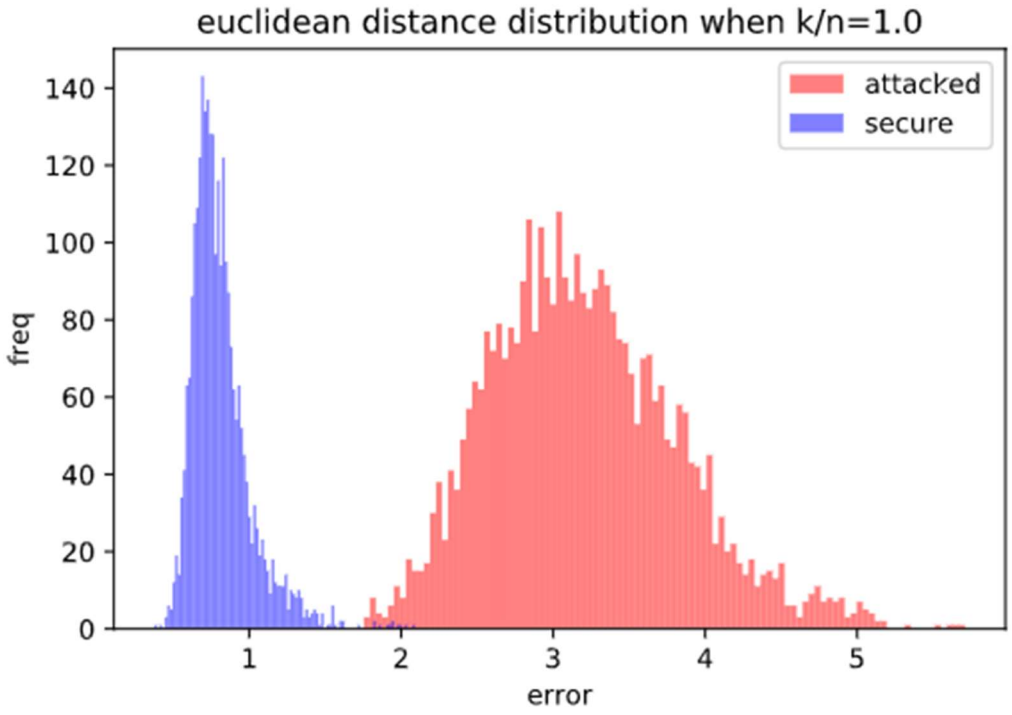

Figure 4-35 Distribution of Euclidean distance between predicted data and actual data under normal targeted attacks with k/n=1.0

4.2.4.2 Playback Targeted Attacks

At the same time, the playback targeted attacks data set was also analyzed. Figure 4-36 to Figure 4-46 show distance distributions. For playback attacks, since the value of the state variable was based on historical data rather than changed randomly, the deviation between the actual value and the predicted value was much larger than that under normal targeted attacks. The deviation almost exceeded 100 when the grid is attacked while mostly not exceeded five when the grid is safe. This phenomenon accounts for why all classifiers performed better on the playback dataset than the normal dataset. Of course, an overlap still existed at each level, resulting in misclassification sometimes.

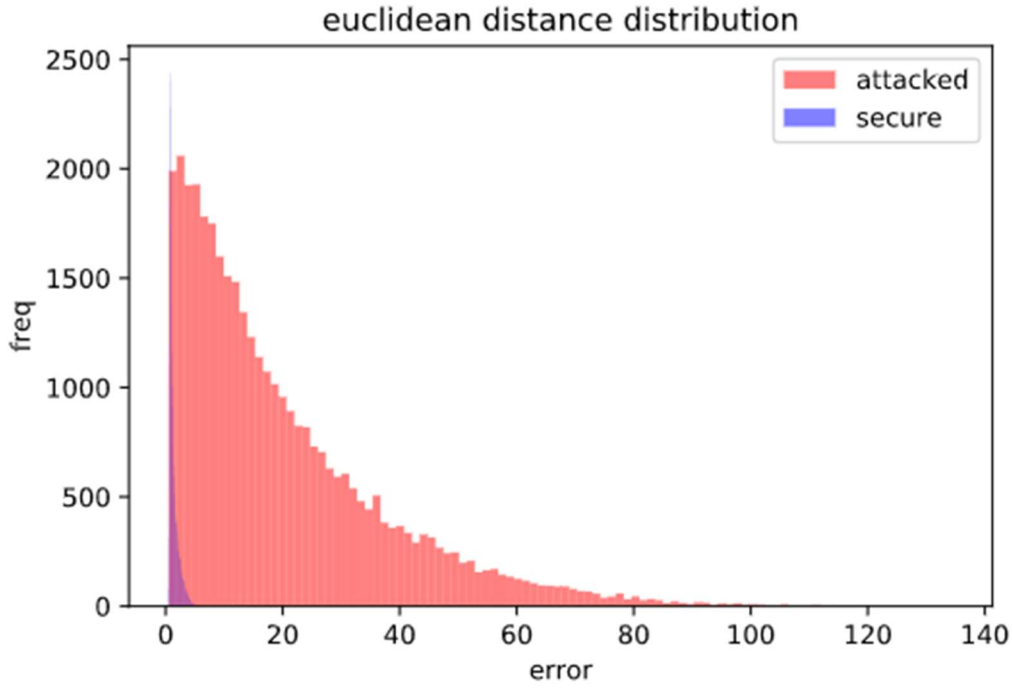

Figure 4-36 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks of mixed levels

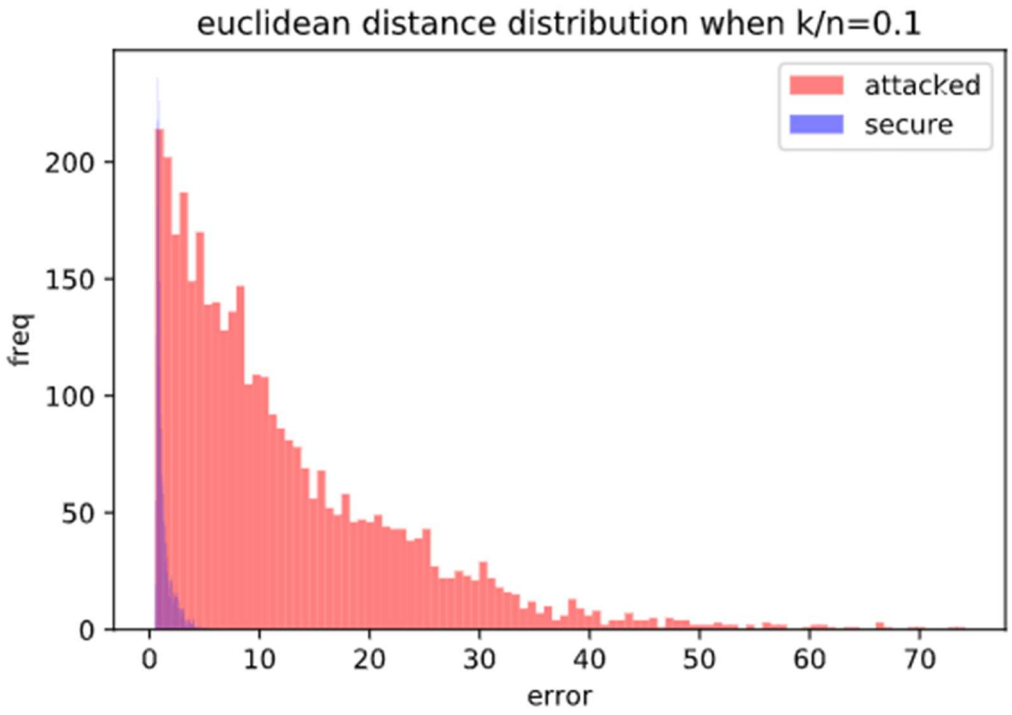

Figure 4-37 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.1

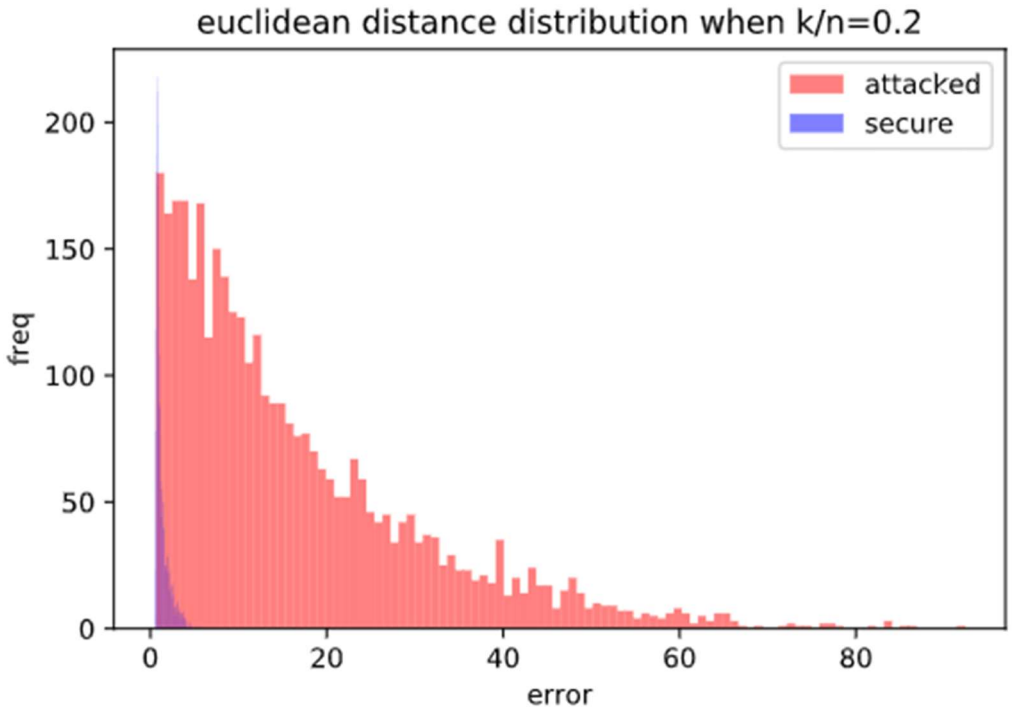

Figure 4-38 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.2

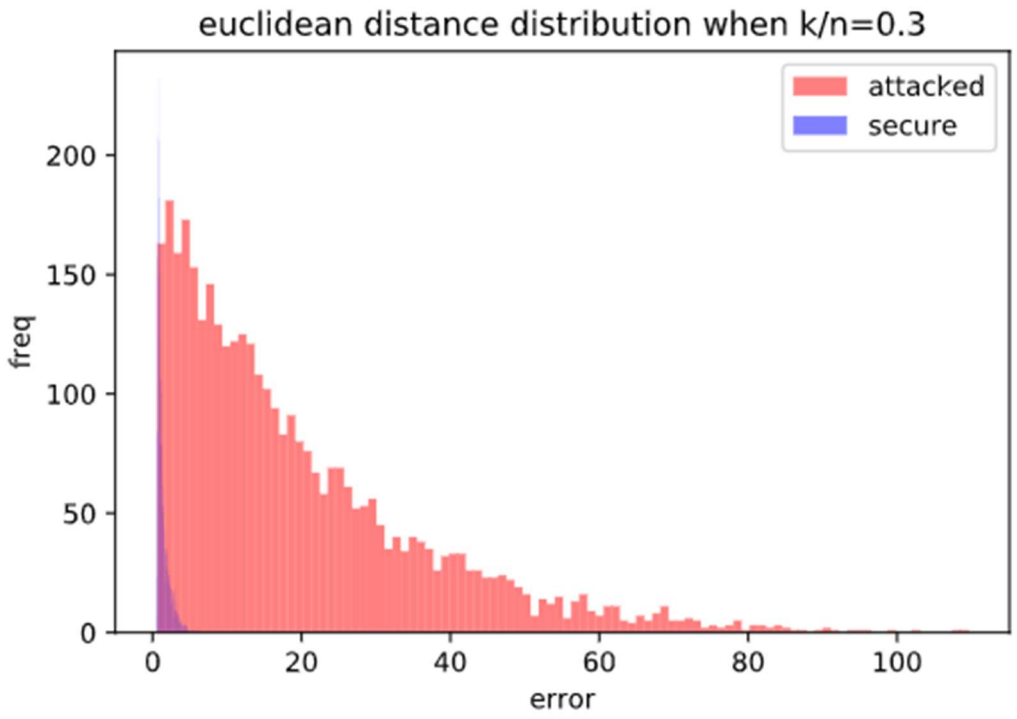

Figure 4-39 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.3

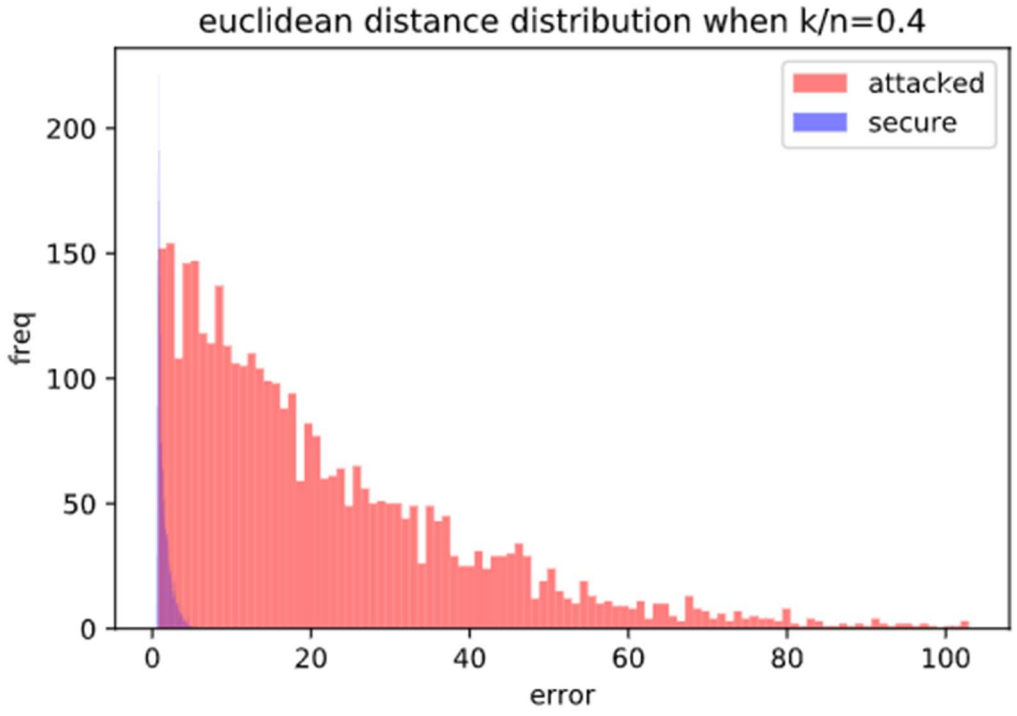

Figure 4-40 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.4

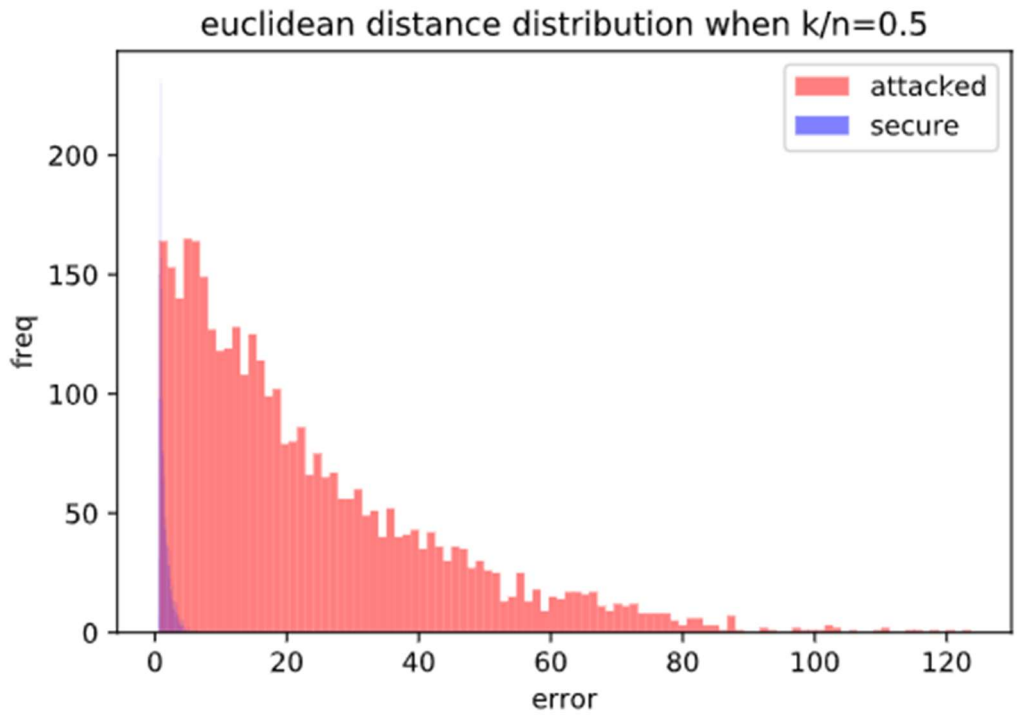

Figure 4-41 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.5

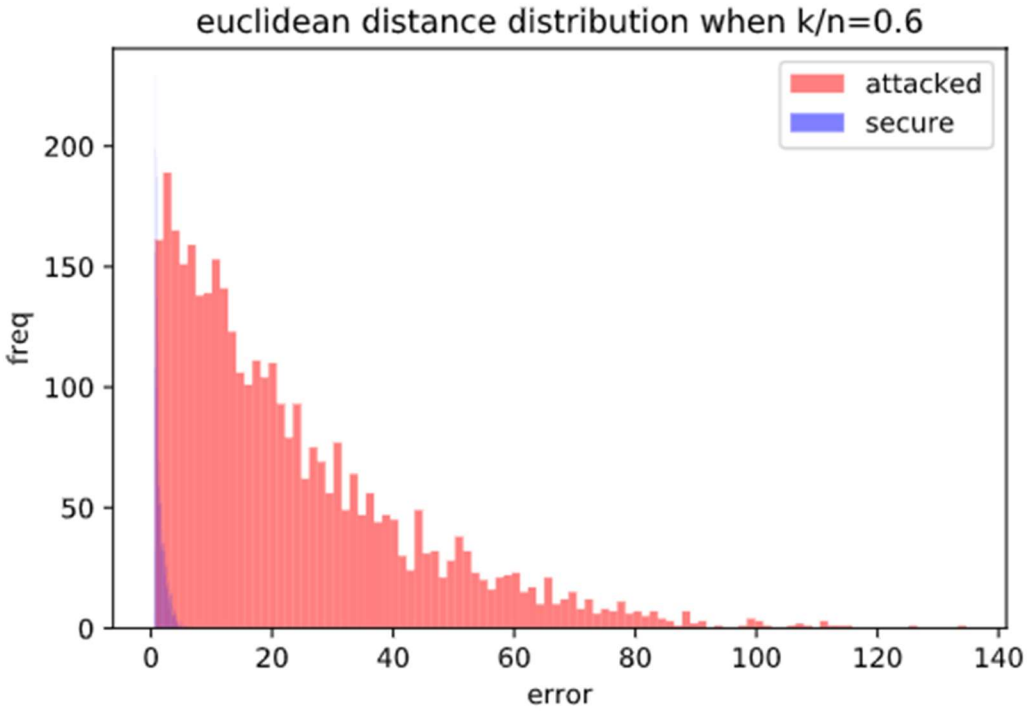

Figure 4-42 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.6

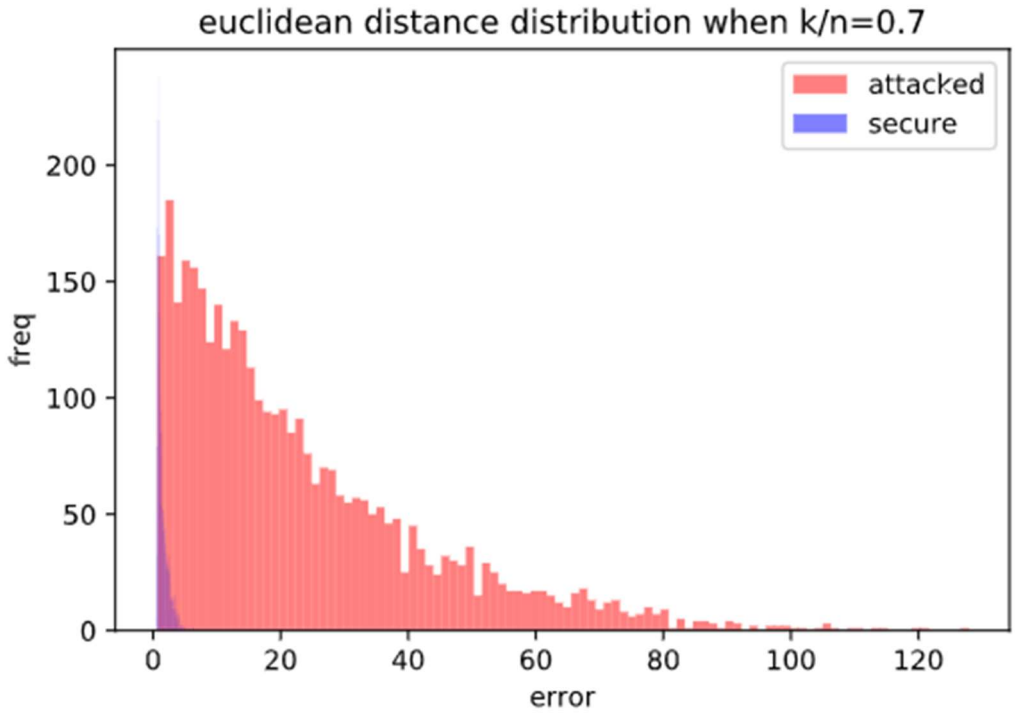

Figure 4-43 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.7

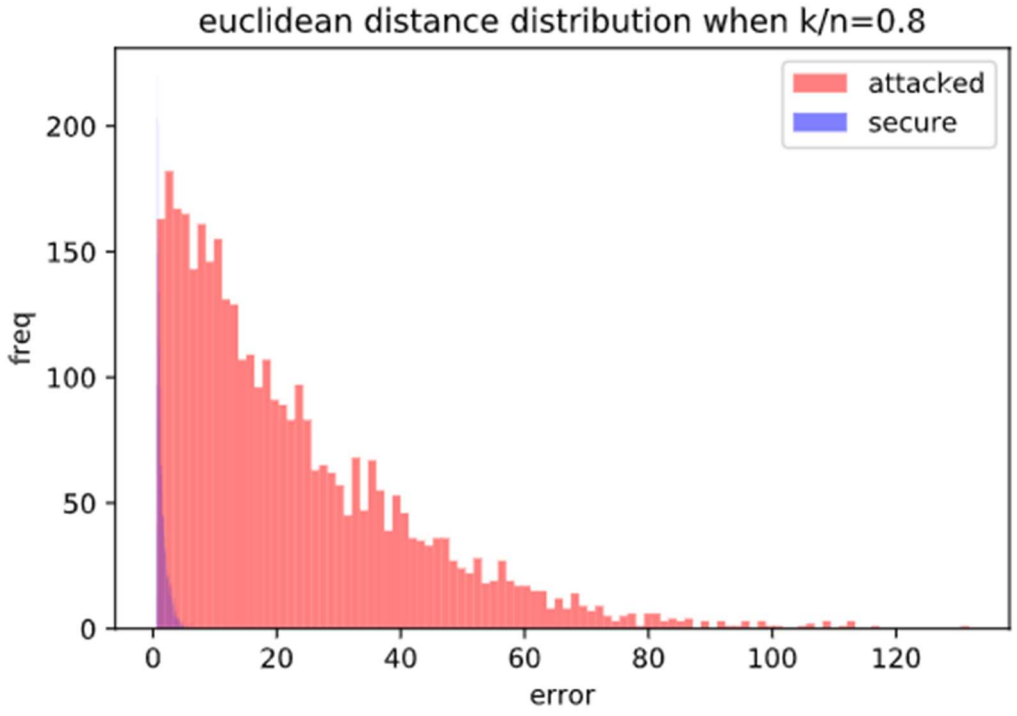

Figure 4-44 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.8

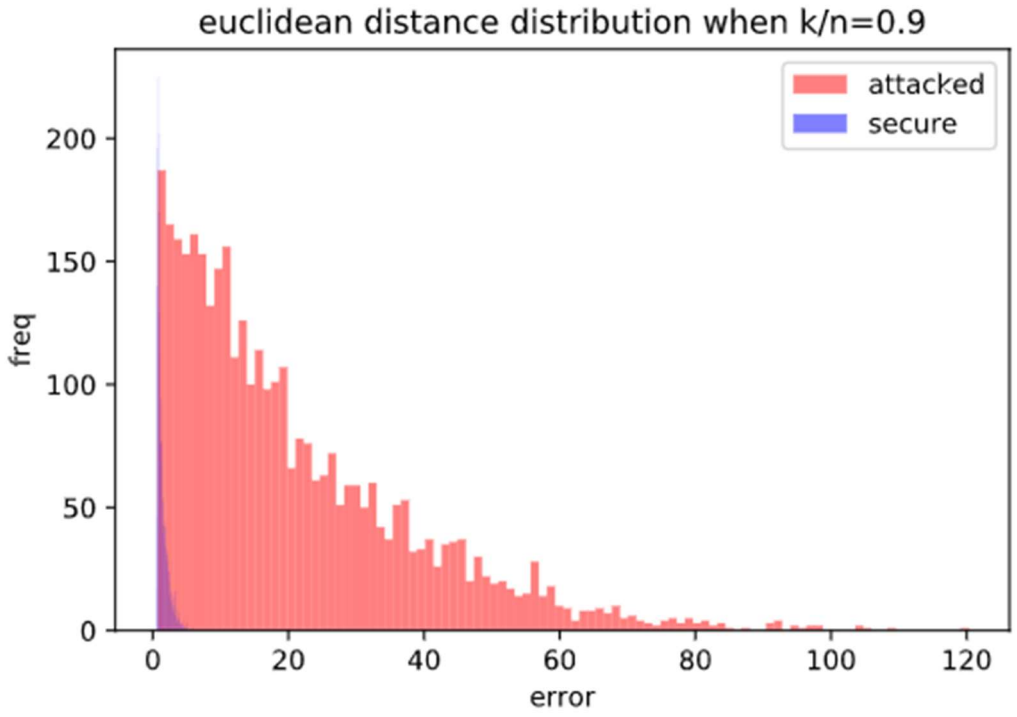

Figure 4-45 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=0.9

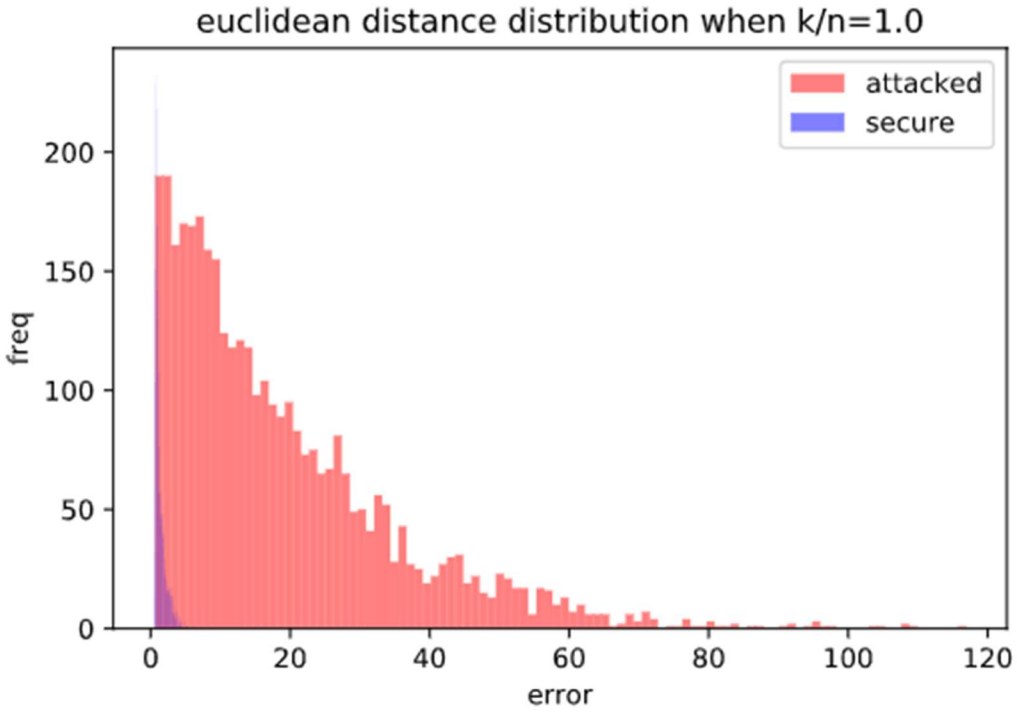

Figure 4-46 Distribution of Euclidean distance between predicted data and actual data under playback targeted attacks with k/n=1.0

4.3 Prediction Based on Measurement Restoration

In the predicting process, once the discriminator sets an alarm, the damaged data is passed to the restorer to repair. Then the repaired data participate in the prediction at the next moment. Figure 4-47 shows the MSE of prediction with the aid of such a recovery mechanism. It is found that the MSE trends to be higher than 0.0071, which is obtained using an utterly attack-free dataset, and MSE gradually increases as k/n increases, but not greater than 0.0080, only increasing by 0.0008 which is negligible to the discriminator and completely acceptable.

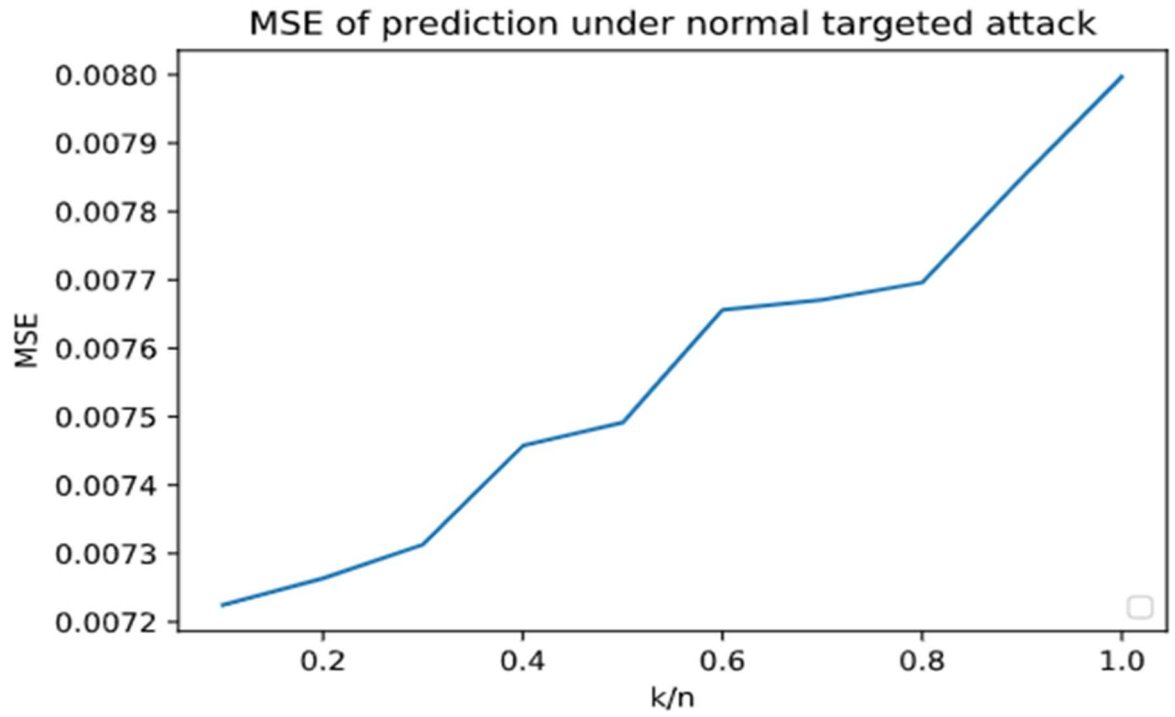

Figure 4-47 MSE of prediction under normal targeted attacks

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this paper, a new prediction-based detector which aims to detect stealthy FDIAs against SE in smart grids was proposed using deep learning techniques. It leverages both convolutional neural networks, which excel in spatial feature extraction, and recurrent neural networks, which excel in exploring temporal correlation. It features a separable architecture, i.e., predictor and discriminator. The predictor learns behaviors from normal historical data. The discriminator learns the deviation between prediction and actual measurements into which may be injected a specific type of attack, for example, targeted attacks in this paper, to predict the probability of being attacked. Additionally, three types of discriminators varying in feature extraction, such as convolution, concatenation, and squared-error vector, were designed and evaluated in comparison with the conventional Euclidean-based method and three common non-ANN machine learning classifiers such as SVM, Random Forest, and KNN.

At last, a measurement restoration mechanism was proposed. Through replacing items with high deviation by prediction, the actual damaged measurements can be repaired to reduce the impact on the predictor, even though the discriminator can accommodate a slight prediction error.

Experiments were carried on IEEE 39-bus power system with load profiles from the real world to assess the performance of the proposed detection mechanism. With the consideration that

targeted FDIAs generated randomly may cover random FDIAs, targeted FDIAs and its playback form were conducted respectively. No matter in normal or playback targeted attacks, the proposed detector with any of three discriminators can achieve outstanding attack detection performance, particularly better in playback form than in normal form due to the former one lacks in randomness. It has also been found that the more devices are compromised, the easier attacks are to be distinguished. Also, the results show that ANN-based discriminators outperform non-ANN machine learning algorithms if time is ample. Particularly, the squared-error-vector-based discriminator was the best under normal targeted attacks, and convolution-based discriminator was the best under playback targeted attacks.

5.2 Future Work

1. Apart from linear stealthy FDIAs having been used in this paper, nonlinear stealthy FDIAs can be applied to test the proposed detector.

2. More data can be collected to be investigated.

3. Fault events can be involved to make the problem more complicated.

4. Larger power systems, such as the 118-bus power system, can be studied to verify the performance of the proposed model.

5. The slow injecting process can be considered.

REFERENCES

[1] A. S. Musleh, G. Chen, and Z. Y. Dong, "A Survey on the Detection Algorithms for False Data Injection Attacks in Smart Grids," *IEEE Trans. Smart Grid*, p. 1, 2019.

[2] *Smart Grid Research: Power - IEEE Grid Vision 2050*.

[3] M. Faheem *et al.,* "Smart grid communication and information technologies in the perspective of Industry 4.0: Opportunities and challenges," *Computer Science Review*, vol. 30, pp. 1–30, 2018.

[4] Wikipedia, *December 2015 Ukraine power grid cyberattack.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=December_2015_Ukraine_power_grid_cyberattack&oldid=946593004 (accessed: Apr. 14 2020).

[5] T. L. Hardy, *Software and system safety: Accidents, incidents, and lessons learned*. Bloomington, IN: Authorhouse, 2012.

[6] POWER, "What You Need to Know (and Don't) About the AURORA Vulnerability," *POWER Magazine*, 31 Aug., 2013. https://www.powermag.com/what-you-need-to-know-and-dont-about-the-aurora-vulnerability/ (accessed: Apr. 14 2020).

[7] Z. Zhang, S. Gong, A. D. Dimitrovski, and H. Li, "Time Synchronization Attack in Smart Grid: Impact and Analysis," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 87–98, 2013.

[8] L. R. Phillips *et al.,* "Analysis of operations and cyber security policies for a system of cooperating Flexible Alternating Current Transmission System (FACTS) devices," 2005.

[9] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A Review of False Data Injection Attacks Against Modern Power Systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1630–1638, 2017.

[10] Yang Weng, Rohit Negi, Christos Faloutsos, and Marija D. Ilic, "Robust Data-Driven State Estimation for Smart Grid," (in English (US)), *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1956–1967, 2017.

[11] P. Eder-Neuhauser, T. Zseby, and J. Fabini, "Resilience and Security: A Qualitative Survey of Urban Smart Grid Architectures," *IEEE Access*, vol. 4, pp. 839–848, 2016.

[12] R. Khorshidi and F. Shabaninia, "A new method for detection of fake data in measurements at smart grids state estimation," *IET Science, Measurement & Technology*, vol. 9, no. 6, pp. 765–773, 2015.

[13] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, p. 13, 2011.

[14] X. Liu and Z. Li, "Local Topology Attacks in Smart Grids," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2617–2626, 2017.

[15] Wikipedia, *Deep learning.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=949467054 (accessed: Apr. 14 2020).

[16] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Sparse Attack Construction and State Estimation in the Smart Grid: Centralized and Distributed Models," *IEEE J. Select. Areas Commun.*, vol. 31, no. 7, pp. 1306–1318, 2013.

[17] L. Liu, M. Esmalifalak, Q. Ding, V. A. Emesih, and Z. Han, "Detecting False Data Injection Attacks on Power Grid by Sparse Optimization," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 612–621, 2014.

[18] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor, "Machine Learning Methods

for Attack Detection in the Smart Grid," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 8, pp. 1773–1786, 2016.

[19] Markus Goldstein and Seiichi Uchida, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data," *PLOS ONE*, vol. 11, no. 4, e0152173, 2016.

[20] S. A. Foroutan and F. R. Salmasi, "Detection of false data injection attacks against state estimation in smart grids based on a mixture Gaussian distribution learning method," *IET Cyber-Physical Systems: Theory & Applications*, vol. 2, no. 4, pp. 161–171, 2017.

[21] Murray Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.

[22] S. McLaughlin, B. Holbert, A. Fawaz, R. Berthier, and S. Zonouz, "A Multi-Sensor Energy Theft Detection Framework for Advanced Metering Infrastructures," *IEEE J. Select. Areas Commun.*, vol. 31, no. 7, pp. 1319–1330, 2013.

[23] ARNAV KUNDU, "DEEP LEARNING TECHNIQUES FOR DETECTION OF FALSE DATA INJECTION ATTACKS ON ELECTRIC POWER GRID," MASTER OF SCIENCE, August/2019.

[24] A. S. Musleh, M. Debouza, H. M. Khalid, and A. Al-Durra, "Detection of False Data Injection Attacks in Smart Grids: A Real-Time Principle Component Analysis," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society: Convention Center, Lisbon, Portugal, 14 - 17 October, 2019*, Lisbon, Portugal, 2019, pp. 2958–2963.

[25] S. Ahmed, Y. Lee, S.-H. Hyun, and I. Koo, "Unsupervised Machine Learning-Based Detection of Covert Data Integrity Assault in Smart Grid Networks Utilizing Isolation Forest," *IEEE Trans.Inform.Forensic Secur.*, vol. 14, no. 10, pp. 2765–2777, 2019.

[26] Y. Song, Z. Yu, X. Liu, J. Tian, and M. Chen, "Isolation Forest based Detection for False Data Attacks in Power Systems," in *2019 IEEE PES International Conference on Innovative Smart Grid Technologies Asia (ISGT 2019): May 21-24, 2019, Chengdu, China*, Chengdu, China, 2019, pp. 4170–4174.

[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[29] J. Li *et al.,* "Jasper: An End-to-End Convolutional Neural Acoustic Model," 2019.

[30] R. Chalapathy and S. Chawla, "Deep Learning for Anomaly Detection: A Survey," 2019.

[31] R. Chalapathy, E. Z. Borzeshi, and M. Piccardi, "An Investigation of Recurrent Neural Architectures for Drug Name Recognition," 2016.

[32] D. Wulsin, J. Blanco, R. Mani, and B. Litt, "Semi-Supervised Anomaly Detection for EEG Waveforms Using Deep Belief Nets," in *2010 International Conference on Machine Learning and Applications*, Washington, DC, USA, 2010, pp. 436–441.

[33] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams," 2017.

[34] Y. He, G. J. Mendis, and J. Wei, "Real-Time Detection of False Data Injection Attacks in Smart Grid: A Deep Learning-Based Intelligent Mechanism," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2505–2516, 2017.

[35] J. Wei and G. J. Mendis, "A deep learning-based cyber-physical strategy to mitigate false data injection attack in smart grids," in *IEEE proceedings of the 2016 Joint Workshop on Cyber-Physical Security and Resilience in Smart Grids (CPSR-SG): 11th April 2016, Vienna, Austria*, Vienna, 2016,

pp. 1–6.
[36] H. Wang *et al.,* “Deep Learning-Based Interval State Estimation of AC Smart Grids Against Sparse Cyber Attacks,” *IEEE Trans. Ind. Inf.*, vol. 14, no. 11, pp. 4766–4778, 2018.
[37] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery,” 2017.
[38] Jinwon An and Sungzoon Cho, “Variational Autoencoder based Anomaly Detection using Reconstruction Probability,” 2015.
[39] A. Ayad, H. E. Z. Farag, A. Youssef, and E. F. El-Saadany, “Detection of false data injection attacks in smart grids using Recurrent Neural Networks,” in *2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT): 19-22 Feb. 2018*, Washington, DC, 2018, pp. 1–5.
[40] S. Binna, S. R. Kuppannagari, D. Engel, and V. K. Prasanna, “Subset Level Detection of False Data Injection Attacks in Smart Grids,” in *2018 IEEE Conference on Technologies for Sustainability (SusTech)*, Long Beach, CA, USA, 2018, pp. 1–7.
[41] A. Kundu, “DEEP LEARNING TECHNIQUES FOR DETECTION OF FALSE DATA INJECTION ATTACKS ON ELECTRIC POWER GRID,”
[42] J. J. Q. Yu, Y. Hou, and V. O. K. Li, “Online False Data Injection Attack Detection With Wavelet Transform and Deep Neural Networks,” *IEEE Trans. Ind. Inf.*, vol. 14, no. 7, pp. 3271–3280, 2018.
[43] Q. Deng and J. Sun, “False Data Injection Attack Detection in a Power Grid Using RNN,” in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society: Omni Shoreham Hotel, Washington DC, United States of America, 20-23 October, 2018 : proceedings*, Washington, DC, 2018, pp. 5983–5988.
[44] X. Niu, J. Li, J. Sun, and K. Tomsovic, “Dynamic Detection of False Data Injection Attack in Smart Grid using Deep Learning,” in *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, Washington, DC, USA, 2019, pp. 1–6.
[45] Y. Wang, W. Shi, Q. Jin, and J. Ma, “An Accurate False Data Detection in Smart Grid Based on Residual Recurrent Neural Network and Adaptive threshold,” in *2019 IEEE International Conference on Energy Internet (ICEI)*, Nanjing, China, May. 2019 - May. 2019, pp. 499–504.
[46] Wikipedia, *Convolutional neural network.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=950807179 (accessed: Apr. 13 2020).
[47] K. Fukushima, “Neocognitron,” *Scholarpedia*, vol. 2, no. 1, p. 1717, 2007.
[48] *Blurring an Image | Apple Developer Documentation.* [Online]. Available: https://developer.apple.com/documentation/accelerate/blurring_an_image (accessed: Apr. 13 2020).
[49] *Max-pooling / Pooling - Computer Science Wiki.* [Online]. Available: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling (accessed: Apr. 14 2020).
[50] Daniel Gibert, “Convolutional Neural Networks for Malware Classification,” Master's thesis, Polytechnic University of Catalonia; Rovira i Virgili University; University of Barcelona, Barcelona, Spain, October 20/2016.
[51] *Convolutional Neural Networks (LeNet) — DeepLearning 0.1 documentation.* [Online]. Available: http://deeplearning.net/tutorial/lenet.html (accessed: Apr. 14 2020).
[52] Wikipedia, *Recurrent neural network.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Recurrent_neural_network&oldid=950614617 (accessed: Apr. 13 2020).
[53] FAVPNG.com, *Recurrent Neural Network Artificial Neural Network Long Short-term Memory Feedforward Neural Network Recursive Neural Network - PNG - Download Free* (accessed: Apr. 13 2020).

[54] Ronald J. Williams and David Zipser, “Gradient-Based Learning Algorithms for Recurrent Connectionist Networks,” April Dec. 1990.
[55] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” (in En;en), *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[56] R. J. Williams and D. Zipser, “A Learning Algorithm for Continually Running Fully Recurrent Neural Networks,” *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.
[57] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks,” 2017.
[58] I. Sutskever, *Training Recurrent Neural Networks*. Ottawa: Library and Archives Canada = Bibliothèque et Archives Canada, 2014.
[59] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
[60] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[61] Q. V. Le, N. Jaitly, and G. E. Hinton, “A Simple Way to Initialize Recurrent Networks of Rectified Linear Units,” 2015.
[62] Guillaume Chevalier, *The LSTM cell.* [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=71836793 (accessed: Apr. 13 2020).
[63] Wikipedia, *Long short-term memory.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=950477748 (accessed: Apr. 12 2020).
[64] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: continual prediction with LSTM,” *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
[65] David Martin Ward Powers, “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation,” *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation*, vol. 2, no. 1, pp. 37–63, 2011.
[66] A. Abur and A. G. Expósito, *Power System State Estimation : Theory and Implementation*: CRC Press, 2004.
[67] S. H. WALKER and D. B. DUNCAN, “Estimation of the probability of an event as a function of several independent variables,” *Biometrika*, vol. 54, 1-2, pp. 167–179, 1967.
[68] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” 2014.
[69] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” vol. 15, pp. 1929–1958, 2014.
[70] Wikipedia, *Receiver operating characteristic.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=948137802 (accessed: Apr. 10 2020).
[71] Wikipedia, *Sensitivity and specificity.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sensitivity_and_specificity&oldid=948670532 (accessed: Apr. 10 2020).
[72] Wikipedia, *Youden's J statistic.* [Online]. Available: https://bioinfopublication.org/files/articles/2_1_1_JMLT.pdf (accessed: Apr. 10 2020).
[73] *Load Data - NYISO.* [Online]. Available: https://www.nyiso.com/load-data (accessed: Apr. 22 2020).
[74] T. Athay, R. Podmore, and S. Virmani, “A Practical Method for the Direct Analysis of Transient Stability,” *IEEE Trans. on Power Apparatus and Syst.*, PAS-98, no. 2, pp. 573–584, 1979.
[75] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, “MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education,” *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.