Chapter 5

A neural-network based identification and location detection of false data injection attacks

5.1 Introduction

The current trend in the detection technique of FDIAs remains localized in their presence detection within the measurement set whereas their exact locations of intrusions remain unknown. To palliate this issue, this work showcases an effective implementation of advanced deep learning models to determine the exact intrusion points in real-time. Such novel deep learning models along with the conventional BDD can effectively identify the presence of any unstructured and structured false data injections within the measurement set, hence providing a cost-effective strategy. These deep learning models can effectively capture the inconsistencies introduced by the attack vectors within the measurement set, thus providing a multilabel classification approach. Moreover, such a class of deep learning architectures being model free endows the grid operators with real-time detection without any preliminary statistical assumptions of the grid. An extensive analysis on the standard IEEE test bench showcases the efficiency of such a class of detection strategy under varying attack and noise margins.

This chapter undertakes an effective FDIA formulation using the full column space against the linear state estimation algorithm as shown in Chapter 2. Moreover, to efficiently identify such a class of attack within the acquired measurements, this chapter

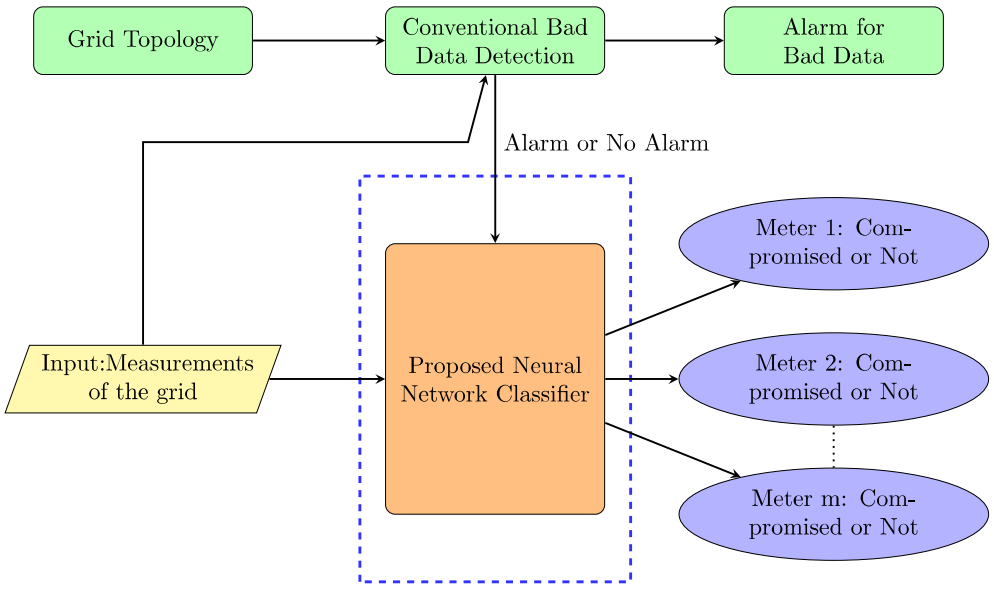

Figure 5.1: FDIA identification strategy

presents a novel data-driven false data detection strategy as depicted in figure 5.1, which is competent enough to detect the intrusion points of the attack vectors in real-time. It can be inferred from Fig. 5.1 that the measurements coming from the respective meters are filtered through the BDD for an effective determination of attack vectors. Unstructured attack vectors can be detected using such an approach. Moreover, it is further classified by the trained multilabel classifiers for effective identification of structured FDIAs developed on the basis of the full column space of the topology matrix as shown in Chapter 2.

It can be seen from Fig. 5.1 that the meters can be labelled as infected or not based on the proposed neural network classifiers. The meters undertaken for this study are conventional line flow and injection meters. Based on the measurements acquired from these respective meters, the state estimation algorithm is invoked. The training data for these respective classifiers can be defined using the labelled data consisting of the state estimates under normal operating conditions and the estimates derived following an attack on the full column space.

5.2 Location identification of FDIA

With the recent advancement in data-driven techniques, FDIA identification strategies have become simpler as they do not require prior statistical assumptions of the grid. Such identification schemes capture the inconsistencies injected due to the attack vector within the measurements. This chapter has performed a comparison between three deep learning models along with a decision tree classifier and an MLP model to showcase the efficacy of the proposed structure. A detailed overview of the proposed models following their implementation has been also presented in this chapter.

5.2.1 Identification of intrusion Points

Identifying the presence of FDIA in the measurement set can be defined as classifying the available measurements into two definite classes like absence or presence of FDIA. This leads to a single-label classification problem. The problem of identifying the locations of attack vector intrusions can be seen as classifying each element of the measurement set into two corresponding labels. Deep learning models have recently demonstrated higher classification accuracies for single-label classification problems, hence this chapter adopts deep learning techniques for effective multilabel classification. In the case of multilabel classification, the feature space is generally highly imbalanced, hence down sampling strategies to balance such feature space is a critical task. To mitigate this problem, effective modeling of the neural network architectures has been performed. These neural network models can easily extract and describe the information from the measurement data, thus giving satisfactory performance.

5.2.2 Proposed FDIA detection policy

Firstly, a CNN model is proposed followed by CNN-LSTM and CNN-BiLSTM models for effective multilabel classification. All such classifiers are fed with the set of available measurements at the control center at discrete time instances and while training such models, only the measurement sets followed by their corresponding truth labels are required. The meters which are injected with attack vectors are marked as the attacked ones and the others are represented as normal ones. This comprise the truth labels used for model training. Such a strategy does not involve any prior statistical information

about the grid and topology matrix H. All the available measurements are firstly given as input to the standard bad data detection algorithm followed by the proposed neural network models. This bad data detection is essential as it is used to determine the quality of the measurements by determining the $\mathcal{L}_2$ norm of the measurement residuals with a pre-determined threshold τ. It is useful in discarding any falsified measurements due to noise in communication channels, meter failures, etc. along with any unstructured FDIA. If the attacker formulates a stealthy attack vector that is capable of bypassing the traditional bad data detection technique, the proposed neural network classifiers can inherently identify the presence and locations of intrusions of attack. The CNN structures followed by its modifications have been implemented inherently in all the proposed deep learning classifiers as it is capable of determining the high dimensional features of attack.

Data used for Training

To train the classifiers, measurements (z^t) coming after bad data detection at discrete instances of time (t) are fed to the models, whereas the output for training can be defined as the truth labels (y^t) of the corresponding meters. The respective output of the models can be defined as $\hat{y}^t$. Truth labels used for training can be defined as:

$$y_i^t = \begin{cases} 1 & \text{Meter i is having an intrusion at time t} \\ 0 & \text{No intrusion} \end{cases} \tag{5.1}$$

A discrimination threshold of 0.5 has been imposed to determine the predicted class of classifier output. Such discrimination threshold plays an important role in defining the accuracy of the trained classifiers as seen in the results section. It can be seen from equation 5.1 that the neural networks are trained based on the defined truth labels. During training a thresholding operation is undertaken for all the neural networks in the last layer as it is fed with the sigmoid activation function. A thresholding of 0.5 has been undertaken for this study as it defines a balanced parity between the true positive and the false positive rates. This leads to defining the balanced boundary between the true positive and false positive classes. A lower thresholding would lead to a higher true positive rate with a lower false positive one and vice versa.

CNN model

The proposed CNN model to determine the locations of FDIA can be seen in figure 5.2. The proposed architecture is developed with an input layer accompanied by several 1D convolution layers followed by a flatten layer. A dense layer comprising the output layer is present after the flatten layer. The input layer is fed with m set of measurements at every discrete time step. Filters have been applied for every 1D convolution layer to the input set of measurements to develop features. To prevent model over-fitting, batch normalization has been adopted. Relu activation function has been adopted for every convolution layer [247]. The developed features ($F_{1,j}$) after the first convolution layer operation over the measurement set can be defined as

$$F_{1,j} = Relu(z * k_{1,j} + b_{1,j}) \tag{5.2}$$

where, $k_{1,j}$ denotes the j^{th} kernel for this convolution layer and $b_{1,j}$ denotes the scalar bias for the first convolution layer. Such scalar biases have been adopted as a common strategy for most of the recent neural network topologies [247]. $(*)$ represents the convolution operation.

Feature maps developed after $(I-1)^{th}$ convolution layer operation are fed to the I^{th} convolution layer. Output from the I^{th} convolution layer can be defined as

$$F_{I,j} = Relu(F_{I-1,j} * k_{I,j} + b_{I,j}) \tag{5.3}$$

where, $F_{I,j}$ can be seen as the j^{th} features developed after the I^{th} convolution operation. Hyper-parameters for such kinds of architectures can be seen as the filters at every layer followed by the number of convolution layers. The features ($F_{I,j}$) as developed from the I^{th} convolution layer are fed to a single flatten layer to develop a single vector. The flatten layer is sequentially connected to the fully connected dense layer. Relu activation function has been adopted for this flatten layer as follows.

$$F_F = Relu(w_f \times F_I + b_f) \tag{5.4}$$

where F_F, w_f and b_f denote the extracted features, weights and bias for the flatten layer respectively. The dense layer after the flatten layer is connected to the m outputs of the output layer. A sigmoid activation function is adopted at the output layer to establish the corresponding labels of the associated measurements. For any measurement j at time

t, the label for the corresponding meters can be developed as the final output of the multilabel classifier which can be defined as:

$$\hat{y}_j^t = Sigmoid(w_d \times F_F + b_d) \tag{5.5}$$

where w_d and b_d denote the weights and the bias respectively at the dense layer. A detailed

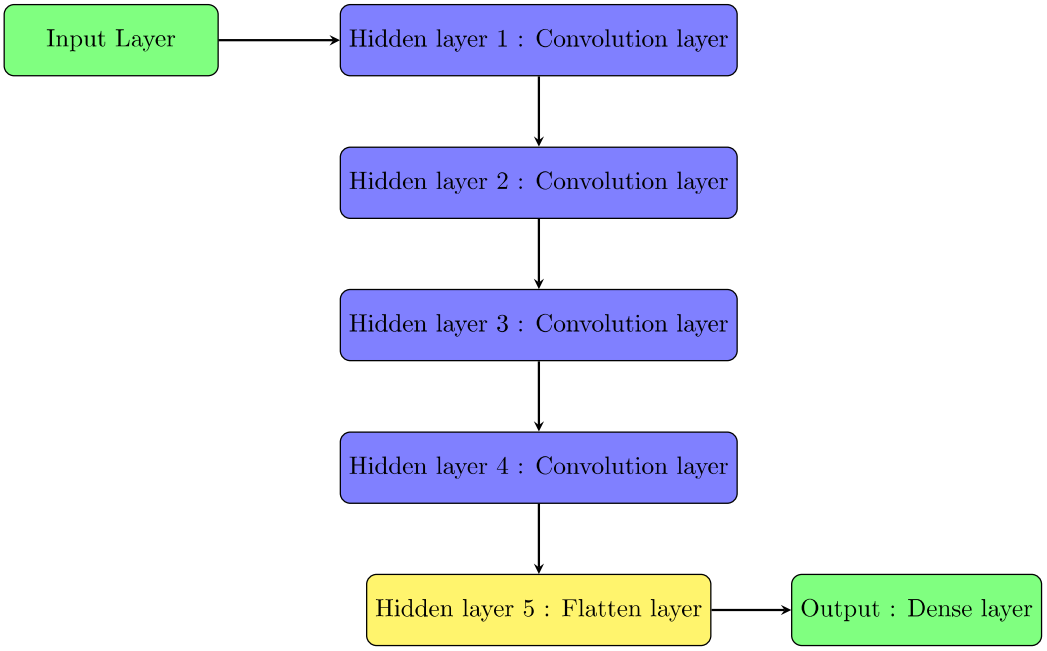

Figure 5.2: Proposed CNN model

overview of the underlying model parameters for the proposed convolution neural network can be seen in Fig. 5.3. It shows the number of trainable parameters (weights) at each layer of the developed CNN multilabel classifier.

CNN-LSTM model

The developed deep learning model can be defined as an input layer accompanied by several convolution layers having the Relu activation function. The features developed after convolution operation are given as input to a single hidden dense layer which is accompanied by several LSTM module layers. The outputs received from the LSTM modules are given as input to a single flatten layer accompanied by several dense layers with an output layer. m discrete sets of measurements are provided at the input layer. Filters are applied at each convolution phase to develop features. To prevent model overfitting, batch normalization has been adopted. The features generated after convolution

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 176, 128)          768
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 174, 256)          98560
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 172, 128)          98432
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 170, 128)          49280
_________________________________________________________________
flatten_1 (Flatten)          (None, 21760)             0
_________________________________________________________________
dense_1 (Dense)              (None, 180)               3916980
=================================================================
Total params: 4,164,020
Trainable params: 4,164,020
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.3: CNN network parameters

operation as per equation (5.3) are given as input to a single hidden layer whose output can be given as:

$$y_{d1} = Relu(w_{d1} \times F_I + b_{d1}) \tag{5.6}$$

Nonlinear activation function Relu has also been adopted at this single hidden layer. The output of this single hidden layer is fed into LSTM modules. w_{d1}, b_{d1} represent the weights and the biases at the single hidden layer respectively. The output generated from the LSTM modules can be defined as per (4.10). A single flatten layer follows the LSTM modules, whose output can be seen as per (5.4). This develops a single vector.

$$F_{F1} = Relu(w_{f1} \times h_1'(t) + b_{f1}) \tag{5.7}$$

where w_{f1}, F_{F1} and b_{f1} respectively represent the weights, extracted features and the biases at the flatten layer. The output from the flatten layer is given over a series of hidden layers having the Relu activation function. The final dense layer is connected to the m nodes of the output layer having a sigmoid activation function as per equation

(5.5). It is used to determine the truth labels of the corresponding measurements. Such a model having LSTM layers over traditional CNN architecture leads to a better multilabel classification strategy than the MLP model. The proposed CNN-LSTM model can be

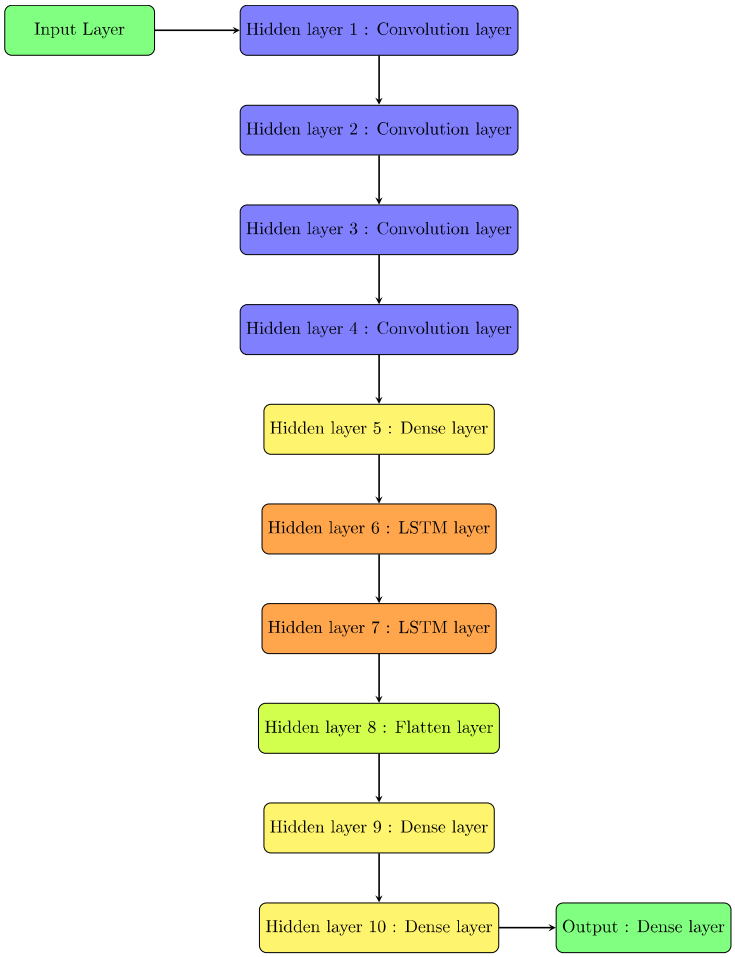

Figure 5.4: Proposed CNN-LSTM model

seen as per Fig. 5.4 while a single LSTM module is shown as per Fig. 4.2. An overview of the model parameters of the proposed CNN-LSTM deep learning framework can be seen in Fig. 5.5. It shows the number of trainable parameters (weights) at each layer of the developed CNN-LSTM multilabel classifier along with the output shape of each hidden layers.

CNN-BiLSTM model

It can be easily seen that basic LSTM structures can easily access the previous information at every specific time instance. However, for the FDIA detection problem, it can be seen

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_5 (Conv1D)            (None, 176, 128)          768
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 174, 256)          98560
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 172, 128)          98432
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 170, 128)          49280
_________________________________________________________________
dense_5 (Dense)              (None, 170, 180)          23220
_________________________________________________________________
lstm_3 (LSTM)                (None, 170, 100)          112400
_________________________________________________________________
lstm_4 (LSTM)                (None, 170, 100)          80400
_________________________________________________________________
flatten_2 (Flatten)          (None, 17000)             0
_________________________________________________________________
dense_6 (Dense)              (None, 25)                425025
_________________________________________________________________
dense_7 (Dense)              (None, 100)               2600
_________________________________________________________________
dense_8 (Dense)              (None, 180)               18180
=================================================================
Total params: 908,865
Trainable params: 908,865
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.5: CNN-LSTM network parameters

that the meter measurements are highly correlated with strong temporal dependency. Future context can be useful to deal with such measurement sets. BiLSTMs are capable of accessing and processing data including forward and backward sweeps using two separate hidden layers followed by a feed forward to a fixed output layer. The following equations show the working strategy of Bidirectional LSTMs with the fixed hidden layer while the $\longrightarrow, \longleftarrow$ respectively denote the forward and backward sweeps. An elaborate explanation of working of BiLSTMs can be seen in [248].

For Forward sweep:

$$\overrightarrow{f'_1(t)} = \sigma(\overrightarrow{w_{1f}}[\overrightarrow{h'_1(t-1)}, \overrightarrow{x'_1(t)}] + \overrightarrow{b_{1f}}) \tag{5.8}$$

$$\overrightarrow{c'_1(t)} = \overrightarrow{f'_1(t)} \odot \overrightarrow{c'_1(t-1)} + \overrightarrow{i'_1(t)} \odot \overrightarrow{g'_1(t)} \tag{5.9}$$

$$\overrightarrow{i'_1(t)} = \sigma(\overrightarrow{w_{1i}}[\overrightarrow{h'_1(t-1)}, \overrightarrow{x'_1(t)}] + \overrightarrow{b_{1i}}) \tag{5.10}$$

$$\overrightarrow{g'_1(t)} = tanh(\overrightarrow{w_{1g}}[\overrightarrow{h'_1(t-1)}, \overrightarrow{x'_1(t)}] + \overrightarrow{b_{1g}}) \tag{5.11}$$

$$\overrightarrow{o'_1(t)} = \sigma(\overrightarrow{w_{1o}}[\overrightarrow{h'_1(t-1)}, \overrightarrow{x'_1(t)}] + \overrightarrow{b_{1o}}) \tag{5.12}$$

$$\overrightarrow{h'_1(t)} = \overrightarrow{o'_1(t)} \odot tanh(\overrightarrow{c'_1(t)}) \tag{5.13}$$

For backward sweep:

$$\overleftarrow{f'_1(t)} = \sigma(\overleftarrow{w_{1f}}[\overleftarrow{h'_1(t-1)}, \overleftarrow{x'_1(t)}] + \overleftarrow{b_{1f}}) \tag{5.14}$$

$$\overleftarrow{c'_1(t)} = \overleftarrow{f'_1(t)} \odot \overleftarrow{c'_1(t-1)} + \overleftarrow{i'_1(t)} \odot \overleftarrow{g'_1(t)} \tag{5.15}$$

$$\overleftarrow{i'_1(t)} = \sigma(\overleftarrow{w_{1i}}[\overleftarrow{h'_1(t-1)}, \overleftarrow{x'_1(t)}] + \overleftarrow{b_{1i}}) \tag{5.16}$$

$$\overleftarrow{g'_1(t)} = tanh(\overleftarrow{w_{1g}}[\overleftarrow{h'_1(t-1)}, \overleftarrow{x'_1(t)}] + \overleftarrow{b_{1g}}) \tag{5.17}$$

$$\overleftarrow{o'_1(t)} = \sigma(\overleftarrow{w_{1o}}[\overleftarrow{h'_1(t-1)}, \overleftarrow{x'_1(t)}] + \overleftarrow{b_{1o}}) \tag{5.18}$$

$$\overleftarrow{h'_1(t)} = \overleftarrow{o'_1(t)} \odot tanh(\overleftarrow{c'_1(t)}) \tag{5.19}$$

The final output ($h'_{1b}(t)$) from the fixed output layer of the BiLSTM module can be defined as the concatenated vector defined from the forward ($\overrightarrow{h'_1(t)}$) and backward ($\overleftarrow{h'_1(t)}$) sweeps which can be defined as follows:

$$h'_{1b}(t) = tanh[\overrightarrow{h'_1(t)}, \overleftarrow{h'_1(t)}] \tag{5.20}$$

The proposed CNN-BiLSTM model is quite analogous to the developed CNN-LSTM model and is obtained by only replacing the LSTM layers with the BiLSTM ones. This is done as BiLSTMs are capable of efficiently encoding temporal information within the developed feature space. Relu activation function is accepted for all the layers. The output layer has the sigmoid activation function. Measurements at discrete time instances are given as input to the input layer, while the output layer showcases the truth labels of the multilabel classifier. The output of the BiLSTM layer which is given as input to the single flatten layer can be defined as:

$$F_{F2} = Relu(w_{f2} \times h'_{1b}(t) + b_{f2}) \tag{5.21}$$

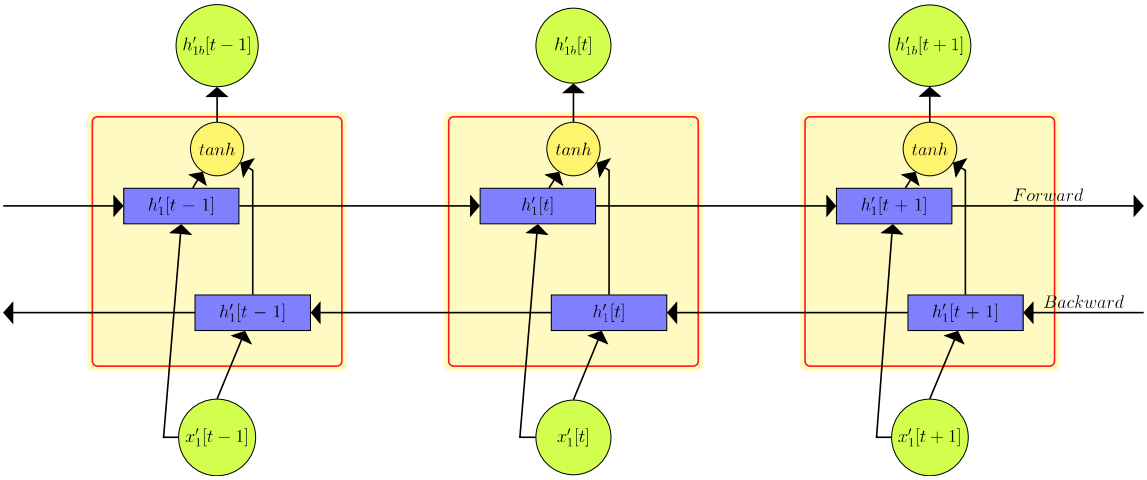

Figure 5.6: Bi-Directional LSTM modules

$h'_{1b}(t)$ can be seen as the output of the BiLSTM layers as shown in equation (5.20). F_{F2}, w_{f2}, b_{f2} respectively represent the developed features, weights and the bias at the flatten layer. Bi-Directional LSTM modules can be shown as per Fig. 5.6. The proposed CNN-BiLSTM model can be shown as per Fig. 5.7 with an overview of the model parameters as depicted in Fig. 5.8.

For conventional convolution neural networks, pooling and dropout layers are crucial. However, the aforementioned layers are not incorporated in the proposed models as pooling layers are mostly deployed for the downsampling of high-dimensional features, hence providing a quite effective approach for two or three-dimensional convolution operation. On the other hand, all the undertaken models incorporate convolution operations that are one dimensional, hence they prove to be efficient and effective under normal GPU programming. Moreover, pooling layers are also sometimes used to develop nonlinear mapping within the CNN architectures. This chapter adopts a nonlinear activation function (Relu) for all the proposed structures which establishes inherent nonlinearity within the models. Hence rendering up of pooling layers prove to be quite useful and provides better performance as it may lead to loss of useful data [249]. Dropout regularisation is generally adopted to control model over-fitting. It is to be noted that all the undertaken models encompass batch normalization to prevent model over-fitting which inherently introduces ample noise for each update of the gradient. Pooling and dropout layers may be introduced for such one-dimensional convolution operation, but it barely shows any enhancement of the performance indices of the proposed structures. Hence to achieve a

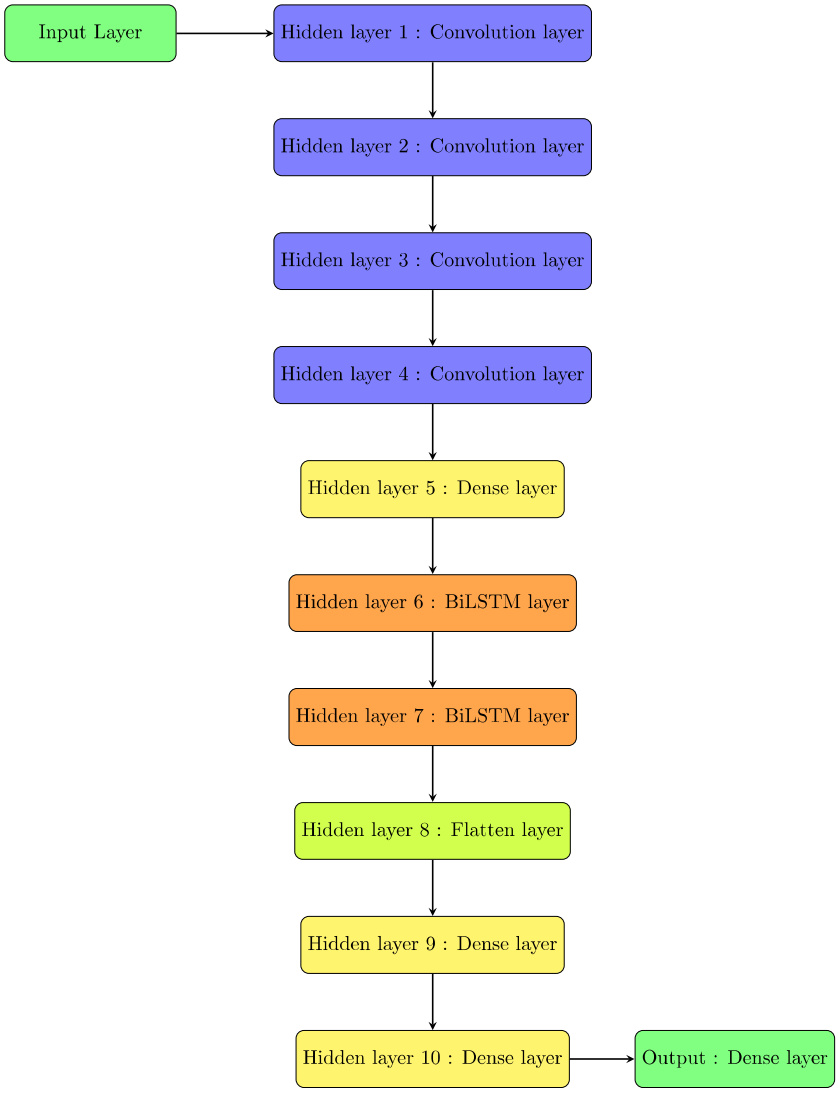

Figure 5.7: Proposed CNN-BiLSTM model

minimal computational burden without loss in model performance, pooling and dropout layers are not incorporated in the proposed structures. As a common approach of incorporating fully connected dense layers following the recurrent neural network structures like GRU which has been reported in the literature [195], this chapter has also adopted them following the flatten layer for the CNN-LSTM and CNN-BiLSTM structures. Such fully connected layers are responsible for interpreting the temporal dependency as developed by the LSTM and BiLSTM structures into indexing representing whether the attack is prevalent within the current set of acquired measurements for the current time step. A similar analogy holds for the CNN model where the output dense layer is responsible for indexing the acquired meter measurements.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 176, 128)          768
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 174, 256)          98560
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 172, 128)          98432
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 170, 128)          49280
_________________________________________________________________
dense_1 (Dense)              (None, 170, 180)          23220
_________________________________________________________________
bidirectional_1 (Bidirection (None, 170, 200)          224800
_________________________________________________________________
bidirectional_2 (Bidirection (None, 170, 200)          240800
_________________________________________________________________
flatten_1 (Flatten)          (None, 34000)             0
_________________________________________________________________
dense_2 (Dense)              (None, 25)                850025
_________________________________________________________________
dense_3 (Dense)              (None, 100)               2600
_________________________________________________________________
dense_4 (Dense)              (None, 180)               18180
=================================================================
Total params: 1,606,665
Trainable params: 1,606,665
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.8: CNN-BiLSTM network parameters

As most of the prevalent FDIA detection approaches incorporate linear deep learning models showcasing a minimal computational burden [197,198,200,250,251], hence the developed models demonstrate a linear structure. Linear models are simpler and are seen to be quite effective with a minimal computational burden during training in comparison to nonlinear structures. Such models also provide satisfactory performance in the detection of the presence of attack within the set of acquired measurements. A nonlinear multilayered perceptron model is also developed for a critical comparison with the inherent linear CNN models. It is seen from the obtained results that such nonlinear structures provide an effective presence detection accuracy along with performance poorer than the proposed CNN model in the detection of locations of intrusions of attack. It is seen that all the

undertaken models incorporate the Relu activation function at each layer with no bias or kernel regularizers. An elaborate overview of bias and kernel regularizers can be seen as per [247]. The deep learning multilabel classifiers incorporate inherent convolution layers where the features are developed using a stride length of one and with no padding on the input features. Furthermore, such models do not incorporate any bias or kernel constraints. The developed models are initialized with a set of bias vectors followed by filters developed by Glorot uniform [247]. With minimization of the demonstrated loss function as shown in equation (5.22), an optimal set of the aforementioned set of parameters was achieved. In the case of the CNN-LSTM and CNN-BiLSTM structures, the developed LSTM and BiLSTM layers over the convolution layers also do not incorporate any bias and recurrent regularizers along with their respective constraints.

Conventional machine learning classifier with traditional MLP model

To effectively portray the efficacy of the furnished CNN architecture, this work has also incorporated a comparative analysis of the proposed deep learning classifier with the conventional machine learning model like a decision tree (DT). As DT classifiers have already demonstrated their accuracy for single-label multiclass and multilabel classification problems, hence this work has incorporated it. Furthermore, an MLP model as shown in Fig. 5.9 has also been adopted as a potential FDIA detector. The nonlinear deep learning model can be seen to have six hidden layers with the presence of sub-hidden layers with one common layer of concatenation. The model is trained with adam optimizer for 500 epochs. The MLP classifier is fed with the Relu activation function at each layer with a dropout regularisation to prevent model over-fitting. The output dense layer has a sigmoid activation function and the model is trained using cross-entropy loss as the loss function as shown in equation (5.22).

5.2.3 Training strategy of the models

To effectively identify FDIA from the measurements given to the classifiers after filtering through the BDDs, optimal tuning of parameters like weights, biases, as well as filters for each layer, has been done carefully. Such an optimal set of parameters leads to an effective nonlinear mapping between the input set of measurements with their corresponding truth labels. Conventional state estimators at the control center are fed with measurements

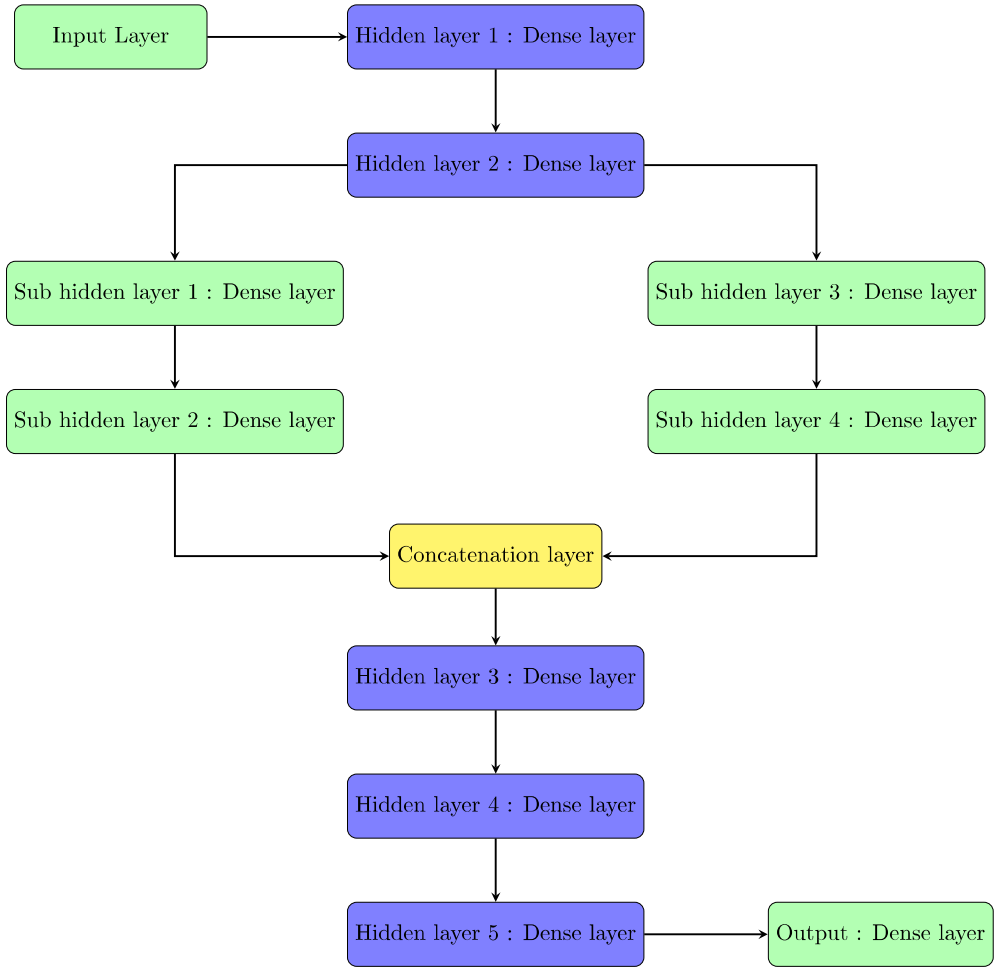

Figure 5.9: Multilayered perceptron classifier

like line power flow, power injections at buses, etc. for an effective determination of the operating states. Operators at the control center adopt the bad data detection technique to determine the quality of the acquired set of measurements at SCADA. Such BDDs can effectively determine the presence of any randomized or unstructured FDIA within the measurements. If the attacker is capable of formulating any structured attacks as shown in chapter 2 with its limited resources, the proposed deep learning classifiers can effectively determine its presence followed by its locations of intrusions. The acquired set of measurements at SCADA which is retrieved from the BDDs are given as input to the multilabel classifiers for accurate detection and identification of an attack.

Mini Batch with Cross Validation

All the proposed structures have inherent convolution layers which use a mini-batch approach for effective training. For each mini-batch approach [247], 300 samples of data are taken randomly from the training set at every iteration. As a common strategy adopted for training the neural network models, the measurement data is split in the ratio of 7:3 where 70% of the measurement data is randomly chosen as the training set and the remaining 30% of the measurement data is kept for model validation. To enhance the model convergence rate, adam [247] optimizer has been adopted for this study followed by a learning rate of 0.0001.

Loss Function

Determination of an optimal set of parameters is a critical task for the neural network models, hence to define a set of optimal parameters, a loss function is defined based on the errors between the actual output from the classifiers with respect to their actual truth labels for each mini-batch. Cross-entropy loss has been adopted as the loss function for the multilabel classification output. The loss function for a mini-batch $\mathcal{T} = [t_1, \ldots, t_{300}]$ is defined as:

$$\mathcal{T} = \sum_{t \in \mathcal{T}} -\frac{1}{m} \sum_{k=1}^{m} (\hat{y}_k^t log(y_k^t) + (1 - \hat{y}_k^t log(1 - y_k^t))) \tag{5.22}$$

where, $\hat{y}^t$ represents the respective output of the models whereas y^t denotes the actual truth labels of the respective meters. Such a clear notion of loss function [247] for the multilabel classifiers leads to an optimal set of parameters.

5.3 Results

This section primarily deals with the generation of training and testing data followed by its implementation within the classifiers for effective multilabel classification. A detailed implementation of stealthy FDIA along with its effective identification policy encompassing the inconsistencies and co-occurrence dependencies of the neighboring measurements have also been presented. The proposed multilabel classifier can effectively identify the locations of intrusions of FDIA followed by their presence detection.

5.3.1 Data set preparation

The primary objective of this section is to showcase an effective location detection of FDIA for the IEEE 118-bus system. Real-time measurements followed by the current grid topology are acquired from the real-time digital simulator. The raw measurement sets showcase a strong correlation for the adjacent lines as well as buses. Raw measurement sets acquired from the real-time digital simulator along with their corresponding truth labels at discrete instances of time are fed as input to the developed multilabel deep learning classifiers for effective training. All the developed deep learning structures have inherent convolution layers with 1-D convolution operation which incorporates a sliding window approach over the input set of measurements as retrieved from the BDDs to develop features. Such sliding window technique is adopted by the filters at each layer.

As all the proposed models have inherent CNN architectures, features can be developed from the raw measurements of adjacent indices. All the meters are labeled based on the present grid topology. The indexing algorithm can be shown as per algorithm 5. All the injection meters are labeled after labeling of the line meters which is done on the

Algorithm 5: Indexing algorithm for meters with their corresponding measurements

1 **for** $a = 1 \longrightarrow 118$ **do**
2 Start labelling of the measurements along with the line flow meters for bus $a = 1$
3 Label all the lines connected with bus a accordingly.
4 **Set** $a = a + 1$

basis of the bus index in ascending order. A fully indexed IEEE 118-bus system as per algorithm 5 can be seen in Fig. 5.10. To train the models, initially, a group of uncorrupted measurements is generated accompanied by a set of corrupted measurements. As the corrupted set of measurements can be generated with the help of structured as well as unstructured FDIA, this chapter encompasses a formulation of structured FDIA in the measurement set as unstructured FDIAs are more susceptive to BDDs [230].

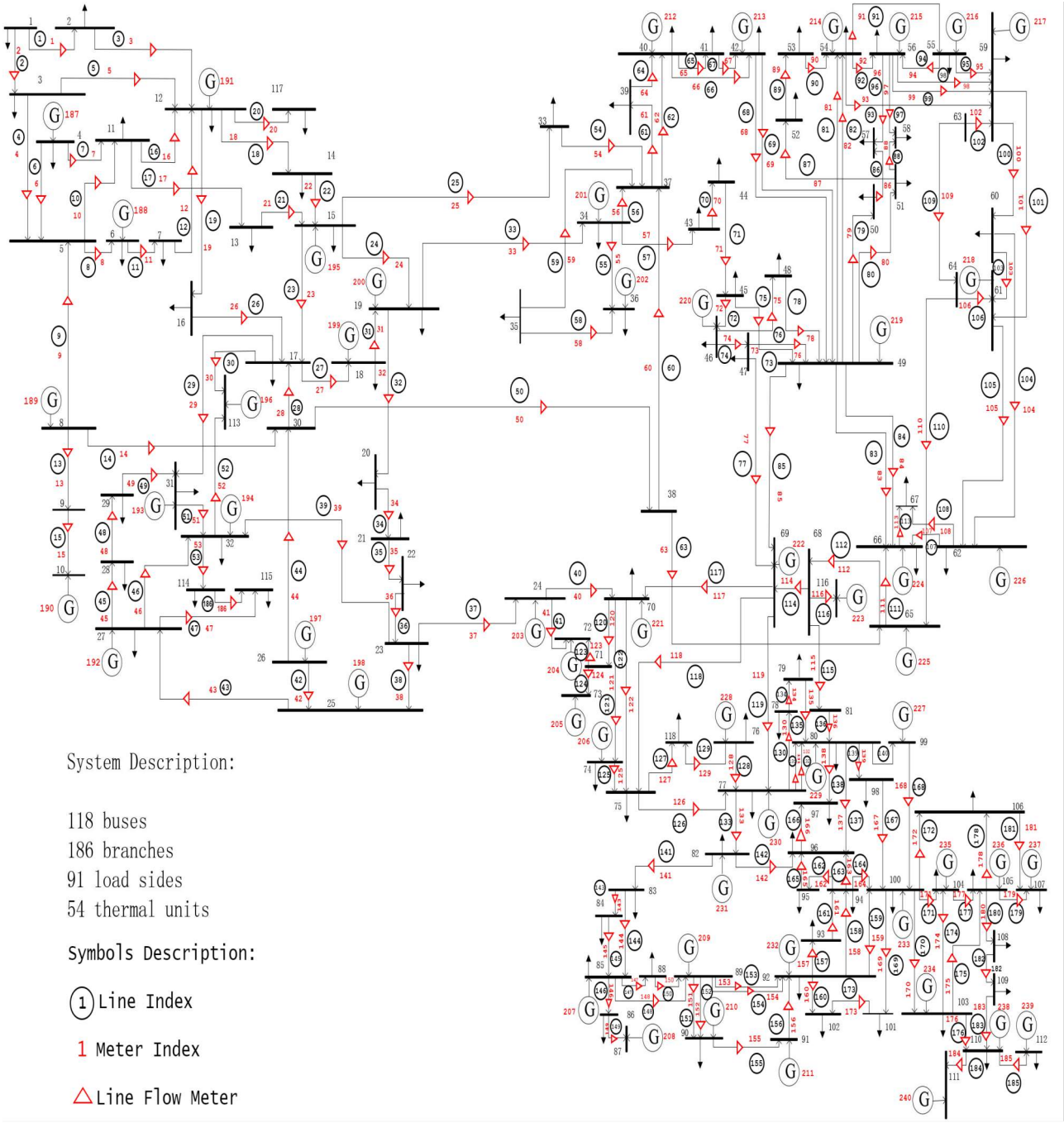

Figure 5.10: An indexed IEEE 118-bus model

5.3.2 Structured FDIA Implementation

It is a well accepted assumption that though the attacker can possess information pertaining to the grid like line admittance, susceptance, etc., still the attacker can rarely gain access to all meters simultaneously and define a perfect attack scenario [42]. A stealthy FDIA has been formulated with the attack vector lying in the column space of the topology matrix H as shown in chapter 2. The attacker can launch an optimal attack vector using limited information about the grid if the attacker can possess it with minimal cost. The following parameters were developed during the formulation of a structured FDIA.

- The state variables ($\hat{x} \in \mathcal{R}^n$) for the IEEE 118-bus system can be seen to develop a discrete uniform distribution. To generate a stealthy attack vector, meters with their corresponding measurements which lead to the development of vulnerable conditions of the grid are targeted [252].

- $\mathcal{L}_2$ norm of the injected attack vector is varied from 1 to 10.

5.3.3 Noise in the measurements

The measurements acquired in the control center have inherent noise in them. Though meters are highly reliable, dynamic noise is still persistent followed by noise incorporated due to communication channels. Hence, a Gaussian noise is incorporated initially within the measurements with a standard deviation of 0.1 and a mean of 0. All the proposed neural network models are hence tested under the following three scenarios like measurements under ideal conditions with no noise followed by measurements under 5% and 10% noise respectively. It can be seen from the results that the proposed CNN architecture can effectively showcase a better performance than the other undertaken models under both ideal as well as under noise conditions.

5.3.4 Training and testing Data

To effectively train the models under various stealthy attack vectors and noise scenarios, 1,00,000 instances of uncorrupted measurements are generated followed by 10,000 instances of corrupted measurements. This leads to the formulation of the training dataset. The testing dataset is comprised of 500 uncorrupted and 500 corrupted instances. For

effective training of the models, initially, the stealthy attack vectors so formulated has an $\mathcal{L}_2$ norm varying within the unit ball. For effective cross-validation of the models, 10 different testing datasets having the same probabilities of injection have been developed. All the results portrayed in the following sections are averaged over all the corresponding testing datasets. The number of hidden layers for the respective models may be defined as one of the hyper-parameters. An optimal set of such hyper-parameters may be defined by assigning the number of hidden layers to a model which showcases the highest F_1 score to the validation set. It must be noted that to showcase the robust performance of the models, this chapter has also undertaken testing datasets with varying noise margins and $\mathcal{L}_2$ norm of attack.

Dataset Visualization using t-SNE

To provide a better explanation of the dataset used for model training, t-SNE (t-distributed stochastic neighbor embedding) has been adopted in this chapter. As seen from [104], for a better feature visualization of the data used for training the models for accurate detection of FDIA, t-SNE-based low dimensional features are developed as can be seen in Fig. 5.11. The undertaken approach (t-SNE) belongs to a nonlinear dimensionality reduction technique that is primarily undertaken to transform high dimensional data into a low dimensional space. As it imbibes probability distribution functions to develop the low dimensional feature space, it can be seen to be having a higher computational burden than principal component analysis (PCA) which belongs to a linear dimensionality reduction technique. Fig. 5.11 furnishes a subset of the training data used for model training. It is seen that t-SNE has a computational burden of nearly 120.223 seconds for the undertaken subset of training data. An overview of t-SNE based low-dimensional visualization of data can be seen in [253].

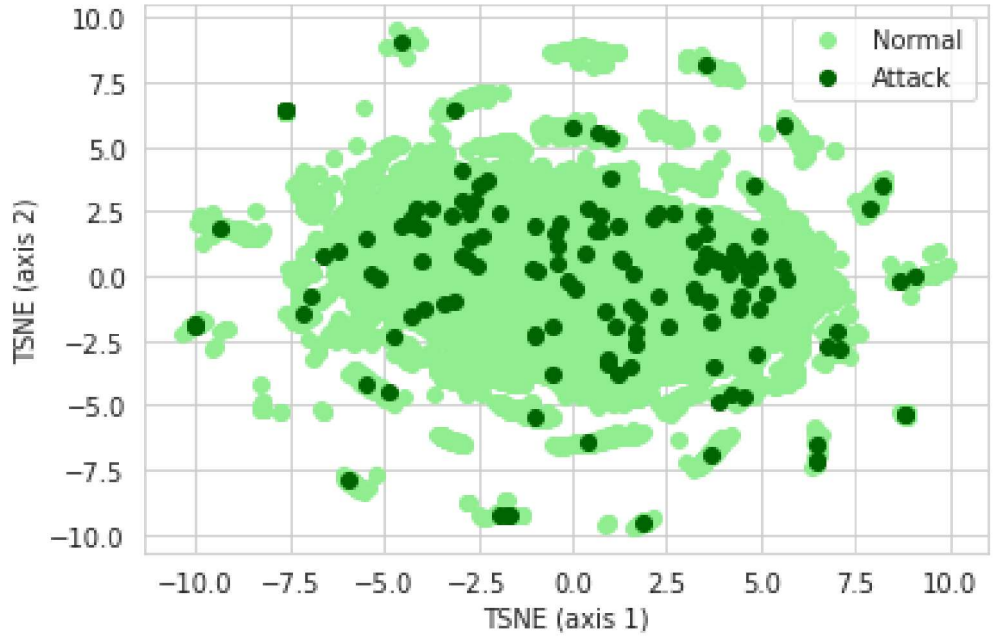

Figure 5.11: t-SNE of a subset of data taken from the training dataset

5.3.5 Performance metrics for detection of intrusions of FDIA

Precision and Recall are undertaken as the primary metrics for all the undertaken models. These metrics can be explained as follows:

$$Precision = \frac{True\ positive\ rate}{True\ positive\ rate + False\ positive\ rate} \tag{5.23}$$

$$Recall = \frac{True\ positive\ rate}{True\ positive\ rate + False\ negative\ rate} \tag{5.24}$$

The true positive rate can be portrayed as those instances where the multilabel classifier can accurately define the compromised locations as compromised. False positive rates can be portrayed as those instances where the multilabel classifier misinterprets an uncompromised location as a compromised one. Moreover, false negative rates may be defined as those instances where the classifier defines a compromised location as uncompromised. F_1 score has also been adopted as one of the key performance metrics in this chapter which is undertaken for a critical comparison between the proposed models. It can be explained as the geometric mean between recall and precision.

$$F_1 score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{5.25}$$

This chapter has also undertaken another performance index for evaluation of model performance namely row accuracy (RACC). RACC may be defined as the probability of

the output of the multilabel classifier where it can accurately label all the compromised locations as compromised followed by labeling all the uncompromised locations as uncompromised. The number of hidden layers was varied for all the undertaken models to define an optimal number of hidden layers.

5.3.6 Results for the IEEE 118-bus system

For the undertaken IEEE 118-bus test bench, a comparative analysis between the proposed multilabel classifiers with an increasing number of hidden layers can be showcased as per table 5.1. The optimal tuning of hidden layers for the MLP classifier can be depicted

Neural Network model	Hidden Layers	Precision (%)	Recall (%)	F_1 score (%)	RACC (%)
CNN	2	98.25	99.14	98.69	87.8
CNN	3	98.52	99.19	98.85	89.1
CNN	4	99.43	99.32	99.37	92.8
CNN	**5**	**99.81**	**99.63**	**99.71**	**93.4**
CNN	6	99.41	99.51	99.45	92.1
CNN-LSTM	6	79.86	80.11	79.98	73.02
CNN-LSTM	7	83.68	83.24	83.45	75.36
CNN-LSTM	8	85.57	86.18	85.87	76.75
CNN-LSTM	9	86.15	86.57	86.35	77.10
CNN-LSTM	**10**	**87.35**	**87.10**	**87.22**	**78.34**
CNN-LSTM	11	86.78	86.67	86.72	77.55
CNN-BiLSTM	6	79.84	80.09	79.96	73.01
CNN-BiLSTM	7	83.70	83.27	83.48	75.45
CNN-BiLSTM	8	85.55	86.14	85.84	76.73
CNN-BiLSTM	9	86.18	86.60	86.38	77.12
CNN-BiLSTM	**10**	**87.35**	**87.16**	**87.25**	**78.39**
CNN-BiLSTM	11	86.81	86.71	86.75	77.59

Table 5.1: Comparison of the models for the IEEE 118-bus system

as per table 5.2. It can be inferred from table 5.1 that the conventional CNN classifier can outclass the other undertaken models based on RACC and F_1 score, hence portraying the efficacy of the developed methodology. It can be deduced from table 5.1 that with

Hidden & (sub hidden layers)	Precision (%)	Recall (%)	F_1 score (%)	RACC (%)
3 (1)	73.47	75.01	74.23	46.12
4 (1)	75.68	77.13	76.39	48.98
4 (2)	77.88	78.34	78.10	55.78
5 (1)	78.98	79.34	79.15	58.12
5 (2)	79.18	80.29	79.73	59.61
6 (2)	**80.48**	**81.69**	**81.08**	**60.01**
7 (2)	80.12	81.23	80.67	59.74

Table 5.2: Optimal set of hyper-parameters of the MLP classifier

an increasing number of hidden layers ($2 \longrightarrow 4$), the performance metrics of the CNN model tend to increase. With more number of hidden layers being added ($4 \longrightarrow 6$), the performance metrics do not show any significant improvement. Moreover, as more number of hidden layers are added, performance metrics decrease slightly. This issue can be coined as a degradation problem. It shows a saturation of accuracy as the depth of the network increases. Moreover, model accuracy decreases rapidly after a definite threshold of hidden layers [249]. It can be clearly inferred from table 5.1 that the performance index of the proposed CNN model increases from layers ($2 \longrightarrow 5$). It gets saturated as more hidden layers are added ($5 \longrightarrow 6$). It can be seen that the proposed CNN model can achieve a very high RACC and F_1 score. As the performance metrics get saturated after layer 5 with an increase in computational burden for the additional layers, the proposed CNN architecture is developed with 5 hidden layers as an optimal hyper-parameter. It leads to a great parity between computational complexity and location accuracy. In the case of CNN-BiLSTM and CNN-LSTM structures, it is well observed that the degradation problem kicks in after 10 hidden layers. Hence an optimal number of hidden layers being equal to 10 has been adopted for these models. The primary selection of the number of hidden layers for the developed multilabel classifiers as shown in table 5.1 is done so as to minimize the cross-entropy loss as the objective function with the least computational burden, hence maximizing the validation accuracy of the respective models. The number of hidden layers with its set of parameters was increased sequentially for the developed classifiers so as to minimize the undertaken loss function. An optimal set of such hyper-parameters are achieved by undertaking the minimal number of hidden layers with its

trainable parameters which demonstrate a satisfactory attack detection accuracy during model training with the least training time. The hyper-parameter tuning strategy of the undertaken multilabel classifiers can be depicted as per [254].

It can be seen that as the number of hidden layers for CNN increases, there is saturation in model performance. The number of hidden layers with their trainable parameters was also optimized for the undertaken CNN-LSTM and the CNN-BiLSTM models. The obtained set of hyper-parameters which minimises the loss function in real-time are depicted as per table 5.1. The model Accuracy of the multilabel classifiers like CNN, CNN-LSTM, and CNN-BiLSTM models under ideal conditions during training can be shown as per fig. 5.12. The validation and training loss of the respective models under ideal conditions can be shown as per fig. 5.13.

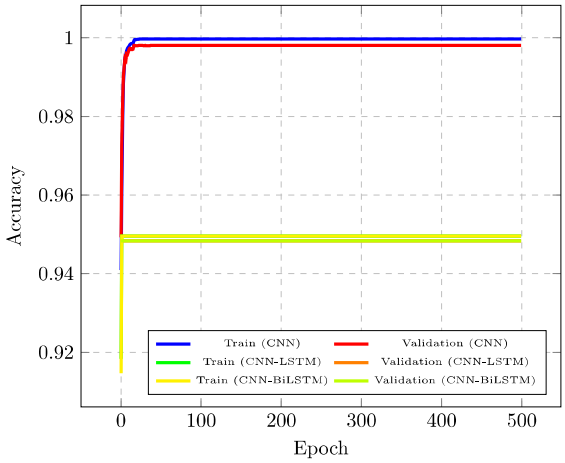

Figure 5.12: Accuracy of the undertaken classifiers under ideal condition

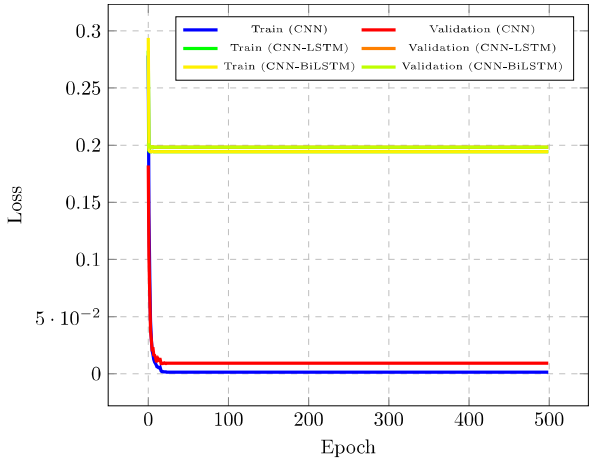

Figure 5.13: Value of loss function of the undertaken classifiers under ideal condition

It can be clearly deduced from the results that the proposed CNN model can better capture the inconsistencies of the measurements due to FDIA, hence showcasing higher accuracy levels. It can be seen that the developed CNN architecture is capable of showing satisfactory performance under 5% and 10% measurement noise also. The accuracy of the multilabel classifiers like CNN, CNN-LSTM and CNN-BiLSTM structures under 5% and 10% noise can be seen as per figs. 5.14 and 5.16 respectively while their training and validation loss can be seen as per figs. 5.15 and 5.17 respectively. Moreover, the

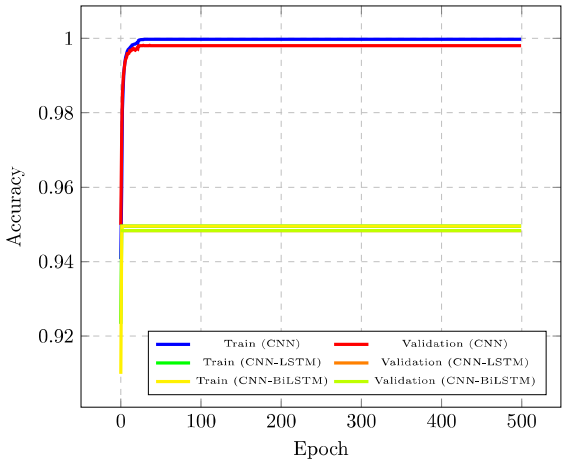

Figure 5.14: Accuracy of the undertaken classifiers under 5% noise condition

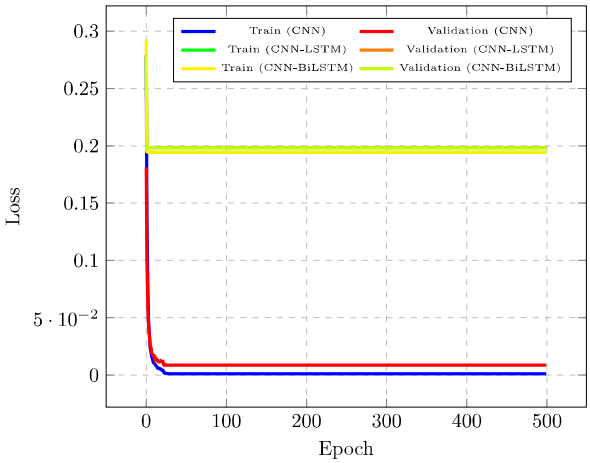

Figure 5.15: Value of loss function of the undertaken classifiers under 5% noise condition

proposed neural network architecture (CNN) shows a better FDIA detection accuracy for measurements with high correlation. During this study, the following locations of intrusions have been selected and an attack vector lying within the unit ball of the $\mathcal{L}_2$ norm has been defined which is shown as follows:

- 3^{rd} measurement is injected with an attack vector while the 1^{st} one is not.
- 1^{st} measurement is injected with an attack vector keeping the 3^{rd} one uncorrupted.
- Both of these locations are injected with an attack vector.
- None of the locations are compromised.

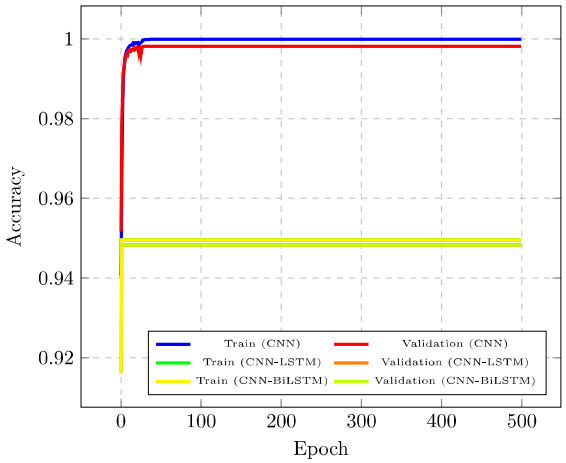

Figure 5.16: Accuracy of the undertaken classifiers under 10% noise condition

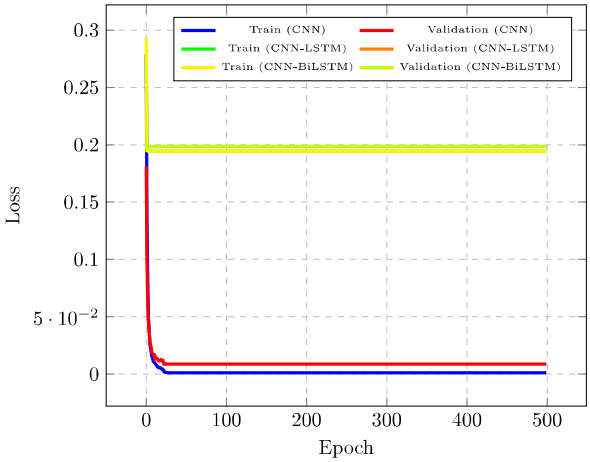

Figure 5.17: Loss of the undertaken classifiers under 10% noise condition

It can be observed from table 5.3 that the proposed architecture can effectively identify the presence of FDIA on 1^{st} and 3^{rd} measurements as they are highly correlated. To showcase the performance of the proposed structure on measurements with the least correlation, this chapter has also defined an attack vector on the 11^{th} measurement as well. It can be observed from table 5.3 that the detection accuracy of FDIA is greater

when the 1^{st}, 3^{rd} measurements are attacked than that of the 3^{rd}, 11^{th} measurement. This is due to the fact that the first set of measurements have high correlation than the second set, while any attack vector introduces inconsistencies within them. This provides a better classification accuracy.

Locations of Intrusion	Accuracy
3^{rd}	0.9960
1^{st}	0.9960
1^{st} and 3^{rd}	0.9989
None	0.9970
3^{rd}	0.9960
11^{th}	0.9956
11^{th} and 3^{rd}	0.9962
None	0.9971

Table 5.3: FDIA identification results on correlated measurements

With advancements in machine learning techniques, an accurate FDIA detection has been achieved [197, 255, 256]. Such machine learning classifiers are highly accurate as they incorporate a binary classification approach [110, 250, 257–259]. The developed CNN model demonstrates a better presence detection accuracy than classifiers trained on such binary labeled data [110]. It is seen that most of the prevalent FDIA detection strategies using deep learning techniques furnish the presence of an attack within the measurements [108–110, 129]. With the deployment of robust deep learning structures, an FDIA identification accuracy of 90% can be achieved. Detection of the presence of attack using a Long-short term memory (LSTM) based denoising autoencoder has recently shown promising results [260].

Although such robust deep learning frameworks furnish an accurate FDIA detection, still they showcase an inherent shortcoming of presence detection only. It can be seen that the developed CNN model demonstrates a superior presence detection accuracy than [108, 109, 129]. In comparison to some recently demonstrated location attack detection approaches using machine learning techniques like SVMs [235], decision trees [235], autoencoder-based generative adversarial networks (AE-GAN) [261], etc. the developed CNN model has demonstrated a superior location detection accuracy with a minimal

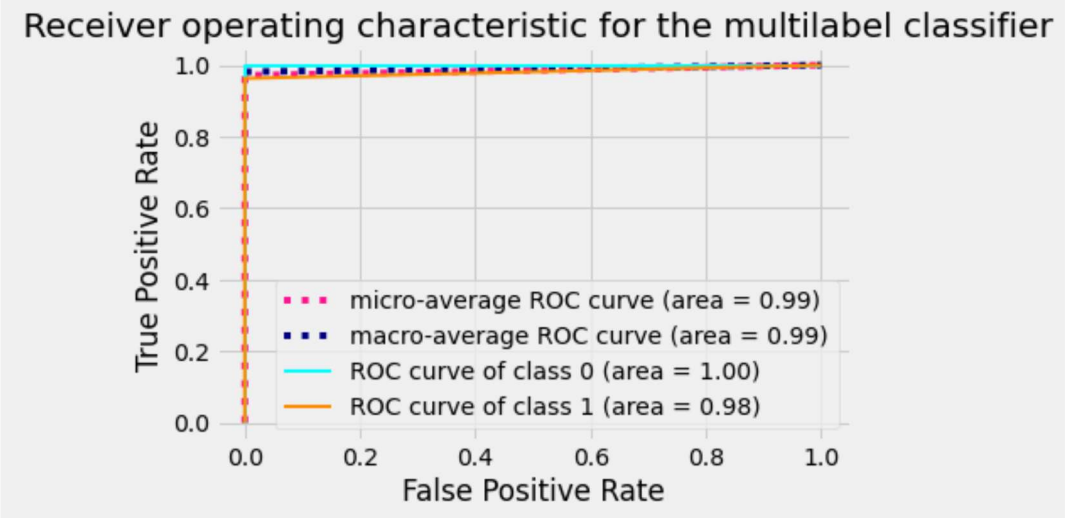

Figure 5.18: ROC curve of the proposed CNN classifier under ideal conditions

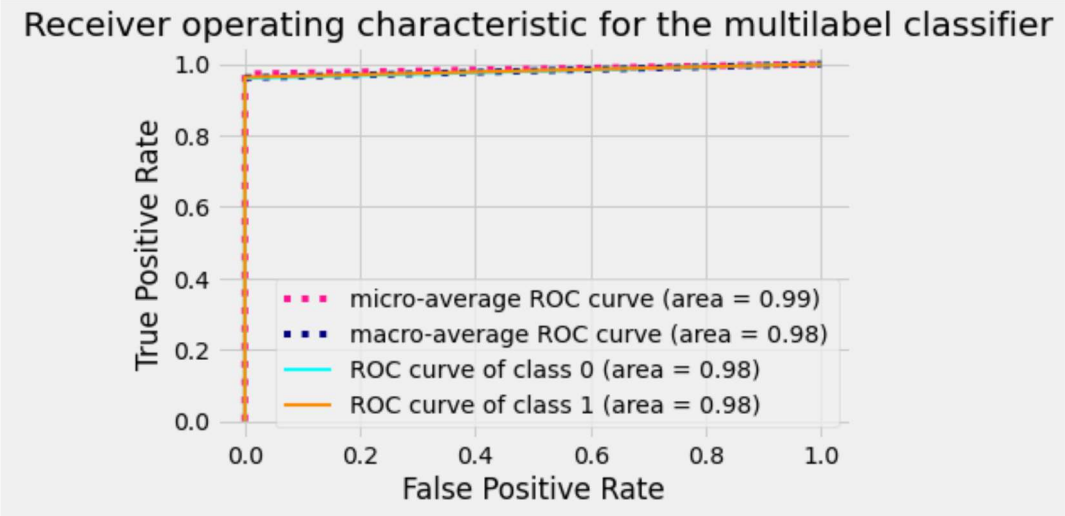

Figure 5.19: ROC curve of the proposed CNN classifier under 5% noise conditions

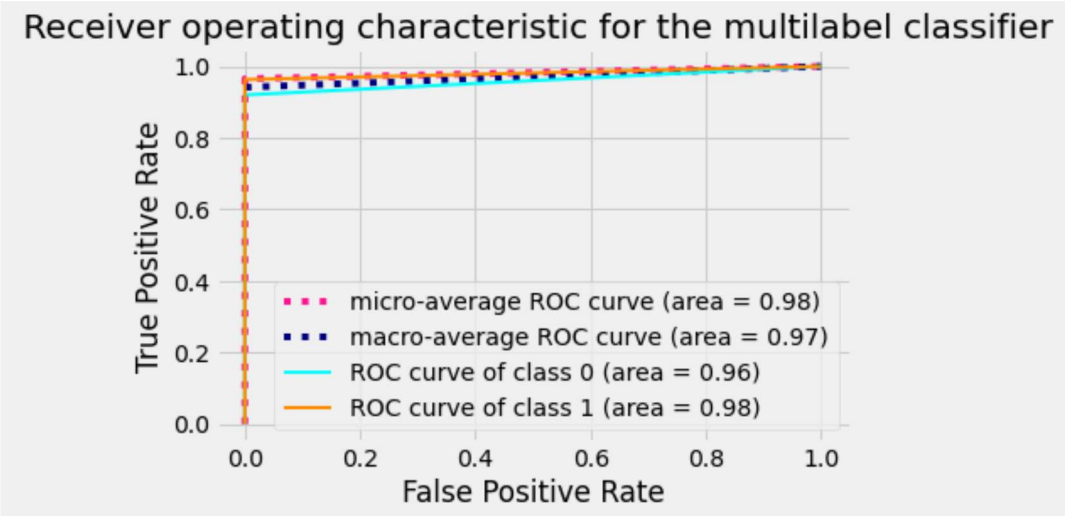

Figure 5.20: ROC curve of the proposed CNN classifier under 10% noise conditions

computational burden [235, 261]. The AE-GAN model showcases an average accuracy of 80% for localization of attacks followed by a presence detection accuracy of 95%, whereas the proposed CNN model demonstrates a location identification accuracy of nearly 99%. The AE-GAN also demonstrates poorer performance metrics like Recall, Precision, and F_1 score than the proposed CNN architecture, hence furnishing an inferior determination of locations of intrusions of attack vectors.

Furthermore, the proposed CNN model also demonstrates a minimal computational burden in comparison to the state-of-the-art studies in this domain like [110,257,261,262]. The proposed approach demonstrates a minimal training time in comparison to location attack detectors like AE-GAN [261]. Although [108,109,129,155,261] demonstrate a state forecasting driven FDIA presence detection model, still they impose a higher computational burden in comparison to the proposed approach. All the undertaken indexes have shown the effectiveness of the proposed CNN model. All such indexes when computed under noise conditions have also demonstrated the superiority of the proposed CNN approach. It can be seen from Fig. 5.18 that the proposed model achieves an operating point close to one under ideal conditions. The proposed CNN multilabel classifier also demonstrates a robust performance with minimal deviation of its operating point while undergoing 5% and 10% noise conditions as shown in Figs. 5.19 and 5.20 respectively. The output of the proposed model is determined following a predefined threshold of 0.5. It leads to the identification of the boundary between the true positive and false positive rates. A selection of subsequently smaller thresholds leads to a higher true positive rate with a lesser false positive rate.

It can be observed from the results that all the undertaken models show a slight decrease in the F_1 score as the noise levels increase. The primary reason for such a scenario is that with an increasing noise level, patterns for normal and corrupted data become less distinguishable. The proposed model achieves a very high F_1 score nearly to 100% as the added Gaussian noise has a standard deviation lying between 0.1-0.5 while the attack vector lying within the unit ball of the $\mathcal{L}_2$ norm. This demonstrates the effectiveness of the proposed approach against noise due to communication channels, meters, etc. It should be noted here that the proposed robust deep learning model (CNN) takes approximately 200μs for training, while SCADA refreshes at the rate of 100 Hz [108, 109, 129]. This promotes the implementation of the proposed robust CNN model for real-time online

FDIA detection scheme showing very less computational burden. It is seen that the CNN-LSTM and CNN-BiLSTM models require a training time of nearly 800 μs and 1200 μs respectively for the undertaken IEEE 118-bus system. The undertaken models have their respective testing times in the order of μs as well, hence leading to a real-time online FDIA detection by the operator in the control center.

5.3.7 Presence identification performance

All the proposed neural networks are also capable of determining the presence of FDIA in the raw measurement set. The grid is claimed to be working under normal operating conditions if $\hat{y}_i^t = 0, \forall\ i = 1, 2, \ldots m$. If the above condition fails for any $\hat{y}_i^t$ *i.e.* $\hat{y}_i^t \neq 0$, this shows the presence of attack vectors in the measurement set. It can be seen from the results that all the models achieve their highest accuracy under ideal conditions.

The accuracy of all the models tends to decrease as the noise margin increases. It can be inferred from Fig. 5.21 that the proposed structure showcases the least variation in the F_1 score with increasing noise margins in the measurements. With increasing strength of the attack vector ($\mathcal{L}_2$ norm) within the measurements, it can be seen that the proposed CNN model showcases a satisfactory performance. All the undertaken models showcase an enhanced identification accuracy with an increasing norm of the attack vector as patterns become distinct for normal measurement data from the measurements under attack. Variation of the F_1 score of the undertaken models with varying $\mathcal{L}_2$ norm of the attack vector can be shown as per Fig. 5.22. Similar nature of variation in F_1 score of the undertaken multilabel classifiers can also be seen if the standard deviation of the injected noise is also varied as can be seen in Fig. 5.23. As the acquired measurements at the control center are subjected to large-scale noises and outliers owing to failures in communication media, and meter failures, this work has also incorporated them. It can be seen from Fig. 5.24 that performance of the undertaken multilabel classifiers degrade with the incorporation of outliers within measurements. The proposed deep CNN model furnishes a minimal variation in the performance metrics like Precision, Recall, and F_1 score in presence of outliers. Such a scenario is prevalent as with the incorporation of outliers within the measurements, patterns for attack and normal operating conditions become less separable. With the incorporation of large-scale noises and outliers, the measurement residuals increase. The BDDs at the control center based on the statistical chi-square

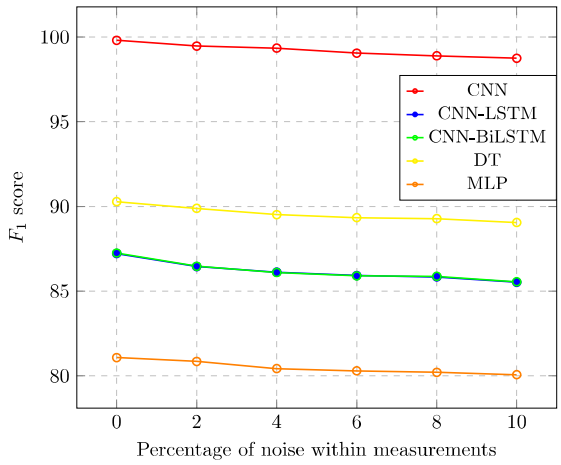

Figure 5.21: Variation of F_1 score for the undertaken models with incorporation of noise within measurements

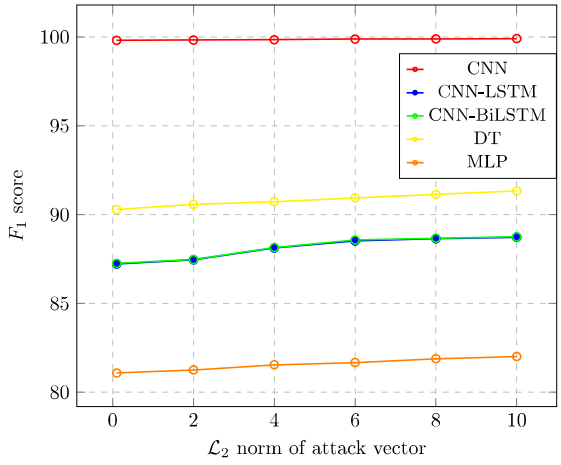

Figure 5.22: Variation of F_1 score with $\mathcal{L}_2$ norm of injected Attack vector

test can successfully detect the presence of large-scale noises and outliers within the set of acquired raw measurements by comparing it with the predefined threshold τ. Measurements with higher residuals due to the incorporation of large-scale noises and outliers are effectively discarded as potential bad data. Due to the presence of sufficient redundancy within the overdetermined system, the state estimation algorithm is undertaken at the control center with the acquired set of measurements after removing the potential bad data.

Furthermore, the proposed model demonstrates a robust performance under varying attack and noise conditions than the nonlinear deep learning models as demonstrated in [108,109,129]. The developed multilabel classifier also demonstrates a minimal variation

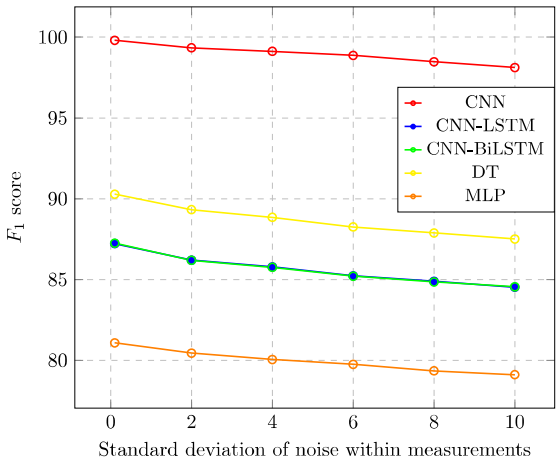

Figure 5.23: Variation of F_1 score for the undertaken models with standard deviation of noise within measurements

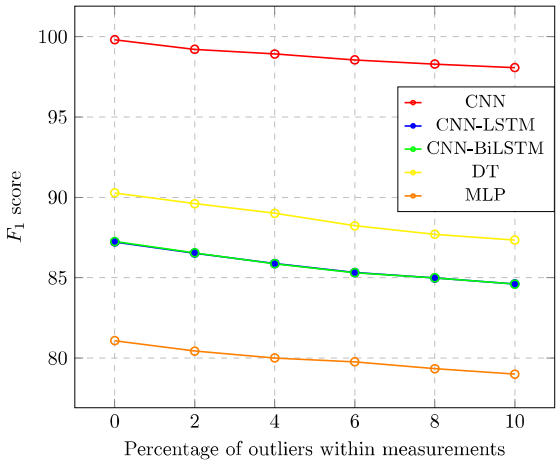

Figure 5.24: Variation of F_1 score for the undertaken models with the incorporation of outliers within measurements

in its performance metrics like Recall, Precision, and F_1 score when outliers are present within the measurements. This inherently demonstrates a better attack detection than state forecasting-driven attack detection schemes as demonstrated in [108, 109, 129].

Thus it can be inferred from Figs. 5.21 - 5.24 that the proposed CNN multilabel classifier is robust enough having the highest detection accuracy with a minimal variation of the performance metrics under all the aforementioned scenarios. Hence it can be concluded that the proposed model can not only identify the intrusion points of FDIA but also can effectively determine their presence within the measurement set in real-time.

5.3.8 Presence and location identification using conventional machine learning classifier and traditional MLP model

Fig. 5.25 demonstrates the reciever operating characteristic (ROC) for the DT classifier acting as a potential FDIA detector for the IEEE 118-bus system under ideal conditions. It can be seen that the DT classifier demonstrates a poorer performance in comparison to the proposed CNN architecture for the IEEE 118-bus system operating under ideal conditions with a Precision, Recall, F_1 score and RACC as $89.88\%, 90.69\%, 90.28\%, 82.11\%$ respectively.

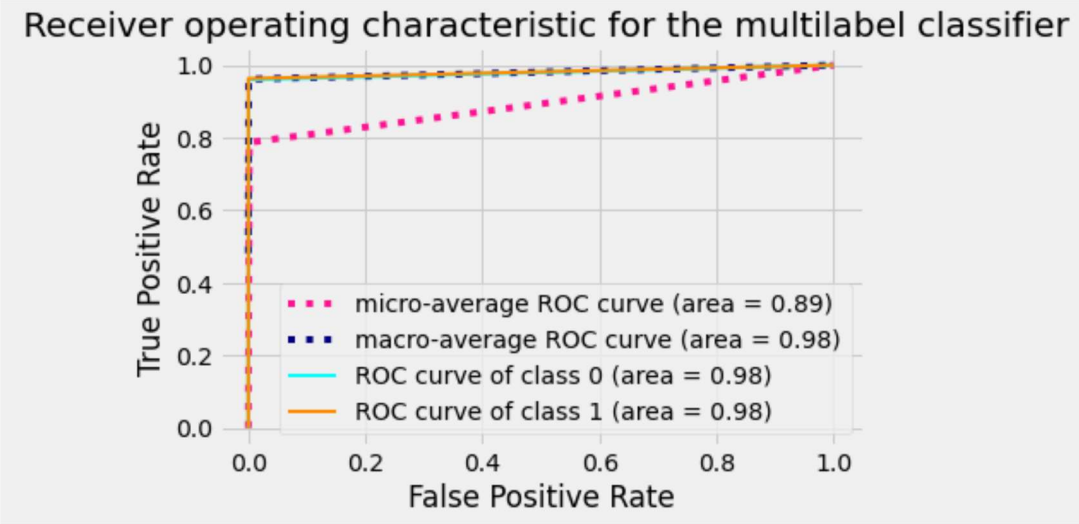

Figure 5.25: ROC curve for the DT classifier

It is seen that the nonlinear MLP model shows a poorer performance than the DT multilabel classifier under ideal conditions for the IEEE 118-bus system with a Precision, Recall, F_1 score and RACC as $80.48\%, 81.69\%, 81.08\%, 60.01\%$ respectively. The operating curve for the MLP classifier can be seen in Fig. 5.26. It can be seen that although the MLP classifier demonstrates a satisfactory attack detection performance as can be seen from the performance metrics like Precision, Recall, and F_1 score, still it shows a poorer performance than the proposed CNN multilabel classifier. It must also be noted that the MLP model also shows a poorer RACC than the other undertaken models which shows that such a classifier is capable of accurately identifying the presence of attacks within the measurements but not its locations for large-scale systems.

It is seen that the undertaken machine learning classifier (DT) has a computa-

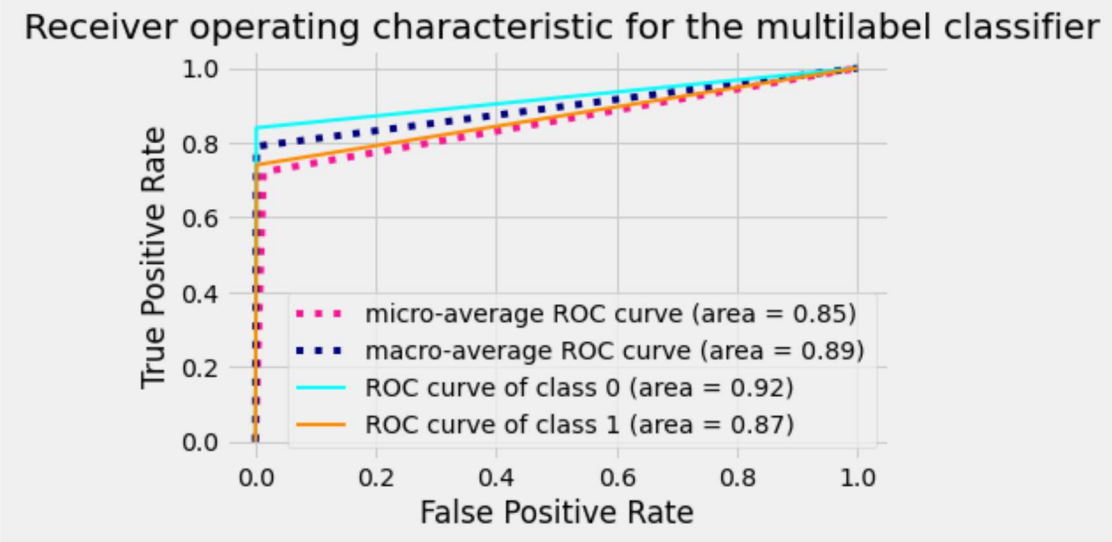

Figure 5.26: ROC curve for the MLP classifier

tional burden of nearly 2000 μseconds while the nonlinear MLP classifier needs nearly 270 μseconds for an effective model training. Although the undertaken MLP classifier demonstrates a computational burden comparable to the proposed CNN structure, still it showcases a poorer FDIA detection accuracy. Furthermore, it is seen from Figs. 5.21 - 5.24 that the performance of all the models (including MLP and DT classifier) decreases with the incorporation of noise and outliers.

5.4 Summary

Hence, it can be concluded that an accurate presence, as well as location detection of FDIA, is possible using the proposed scheme. The neural network models demonstrate an efficient identification of traditional attack vector formulation schemes against the linear state estimation technique under varying noise and attack scenarios, hence developing robust FDIA detectors. Moreover, such detectors can be implemented for large-scale systems as they demonstrate a minimal computational burden with real-time performance.