Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9933111, 19 pages
https://doi.org/10.1155/2021/9933111

Research Article

A Hybrid Deep Neural Network for Electricity Theft Detection Using Intelligent Antenna-Based Smart Meters

Ashraf Ullah,[1] Nadeem Javaid,[1] Adamu Sani Yahaya,[1] Tanzeela Sultana,[1] Fahad Ahmad Al-Zahrani,[2] and Fawad Zaman[3]

[1]*Department of Computer Science, COMSATS University Islamabad, Islamabad 44000, Pakistan*
[2]*Computer Engineering Department, Umm AlQura University, Mecca 24381, Saudi Arabia*
[3]*Department of Electrical and Computer Engineering, COMSATS University Islamabad, Islamabad 44000, Pakistan*

Correspondence should be addressed to Nadeem Javaid; nadeemjavaidqau@gmail.com

Received 12 March 2021; Revised 16 June 2021; Accepted 10 August 2021; Published 26 August 2021

Academic Editor: Daniele Pinchera

This paper presents a hybrid model, named as hybrid deep neural network, which combines convolutional neural network, particle swarm optimization, and gated recurrent unit, termed as convolutional neural network-particle swarm optimization-gated recurrent unit model. The major aims of the model are to perform accurate electricity theft detection and to overcome the issues in the existing models. The issues include overfitting and inability of the models to handle imbalanced data. For this purpose, the electricity consumption data of smart meters is taken from state grid corporation of China. An electric utility company gathers the data from the intelligent antenna-based smart meters installed at the consumers' end. The dataset contains real-time data with missing values and outliers. Therefore, it is first preprocessed to get the refined data followed by feature engineering for selection and extraction of the finest features from the dataset using convolutional neural network. The classification of electricity consumers is performed by dividing them into honest and fraudulent classes using the proposed particle swarm optimization-gated recurrent unit model. The proposed model is evaluated by performing simulations in terms of several performance measures that include accuracy, area under the curve, $F1$-score, recall, and precision. The comparison between the proposed hybrid deep neural network and benchmark models is also performed. The benchmark models include gated recurrent unit, long short term memory, logistic regression, support vector machine, and genetic algorithm-based gated recurrent unit. The results indicate that the proposed hybrid deep neural network model is more efficient in handling class imbalanced issues and performing electricity theft detection. The robustness, accuracy, and generalization of the model are also analyzed in the proposed work.

1. Introduction

The rapid growth of energy consumers has increased the energy demand, which requires efficient generation and distribution of energy at the grid level. In this regard, smart grid [1], with the incorporation of advanced metering infrastructure (AMI), monitors the energy consumption patterns of consumers. AMI establishes a bidirectional communication between consumers and grid to balance supply and demand of energy [2]. Despite of demand management challenge, smart grid faces two types of losses during energy transmission. The first type is technical losses (TLs); whereas the second type is nontechnical losses (NTLs). The former occurs due to energy loss in power distribution lines and transformers. The latter is also known as commercial loss and it occurs due to unregistered connections [3], unpaid bills [4], tampering the antenna-based smart meters, etc., [5]. Moreover, the major reason for NTLs is electricity theft, caused by fraudulent consumers.

NTLs further lead to revenue loss in the economy of countries [6]. For instance, the electric utilities of Brazil and India face about 4.5 billion dollars loss annually [7]. The utilities of USA face around 6 billion dollars loss annually [8]. It is necessary for the power utilities to overcome the revenue loss by detecting NTLs. Therefore, several strategies are being used by the power utilities. The integration of AMI

in power grids provides many advanced and automatic electricity theft detection (ETD) methods. However, calculating such losses and figuring out their exact locations are considered as the most crucial tasks [9]. The inefficiency and less profitability of power utilities are other major concerns. These issues cause extra burden for honest consumers by adding extra charges in their actual utility bills. The aforementioned losses lead to other issues as well, such as hampering and inflation of the industrial routine and load shedding [10].

Several strategies and methods are presented in literature to handle issues related to ETD. These methods are commonly based on hardware, game theory, and data driven [1]. The hardware-based methods, termed as the state-based methods as well [11], perform their operations with the utilization of physical devices, such as transformers, radio-frequency identification tags, sensors, and other electrical equipment. The state-based methods calculate the difference between energy generation at power utility side and energy consumption at consumers' side. These methods achieve high efficiency in theft detection; however, the maintenance and safety of physical devices are major concerns [12]. On the other hand, in the game theory-based methods [13], a game is created between the participants (utilities and energy consumers). Both of the participants compete with each other in order to increase the utility and to get more benefits [14]. Although the game theory-based methods are more efficient contrary to the state-based methods; however, they are based on assumptions and are not able to perform efficient ETD.

In literature, data-driven-based methods are presented to perform efficient ETD. These methods utilize the machine learning (ML) techniques and models. Some of them are decision tree (DT) [3], artificial neural network (ANN) [15], support vector machine (SVM) [16], etc. These techniques use both labeled and unlabeled data to give optimal ETD results, as they have efficient learning capabilities. However, handling the imbalanced class data is challenging for these methods [17]. The traditional classifiers become biased towards the majority class if the data is imbalanced. Therefore, data balancing is required before classification in order to avoid biasness of a classifier and to achieve optimal ETD results. Data balancing is performed using resampling techniques, which balance the data in different classes present in a dataset. The abbreviations used in this work are presented in Table 1.

1.1. Problem Definition and Statement. The increased electricity demand has led to several issues, not just in underdeveloped countries but also in developed countries. The increase in poverty rate is one of the issues, which forces the people to perform electricity theft. In the energy sector, people are adopting illegal means of using electricity to fulfill their demands, i.e., electricity theft. Therefore, ETD is an important thing and needs immediate attention to avoid ever-increasing electricity theft rate. Keeping this in mind, many data analysis techniques, such as SVM, logistic regression (LR), and gated recurrent unit (GRU), have been presented in the literature. However, efficient results have not been achieved yet because of the limitations in these techniques [15]. Some of these limitations are poor learning rate, limited generalization capability, etc. However, the biggest issue being faced is the class imbalanced issue. There exists a huge difference in the number of instances in both classes, i.e., honest and theft consumers' classes. Electricity theft leads to a revenue loss of billion dollars annually for electric utilities [1] and poses serious threats to the country's economy. In addition, electric utilities face electricity losses, which are further classified into TLs ad NTLs; the latter being the most difficult to tackle. NTLs are caused by meddling either with the smart antennas or the smart meters installed at the consumers' end. Therefore, to deal with the class imbalanced issue and to avoid NTLs, an efficient model is presented in this work, termed as particle swarm optimization GRU (PSO-GRU).

1.2. Contributions. This work presented a new variant of neural networks (NNs), named as new hybrid deep neural network (HDNN), in order to address the class imbalanced and overfitting problems in ETD. This work extends the idea present in [18]. The following are the primary contributions of this work:

(i) A metaheuristic model, known as particle swarm optimization (PSO), is used with conventional GRU and convolutional neural network (CNN) to fine tune the parameters and to improve the learning rate, which makes the proposed PSO-GRU model more generalized in terms of training and testing in order to solve the overfitting issue

(ii) A real-world electricity consumption (EC) dataset provided by State Grid Corporation of China (SGCC) [19] is used

(iii) Data normalization and preprocessing are done using local average method and min-max normalization technique, respectively, and

(iv) A comparison is made between the proposed and existing models, which proves the model's efficiency in terms of ETD

The rest of the paper is organized as follows. Section 2 gives an overview of the related work done for ETD. The proposed HDNN model is described in Section 3. The simulations performed in the proposed work are discussed in Section 4. In the end, Section 5 concludes the paper and presents the future work.

2. Related Work

In literature, many systems and approaches are presented for ETD. Most of them are based on hardware and game theory. However, maintenance and data diversity problems are still faced by these approaches. It is observed that the ML techniques are better than the abovementioned ETD methods due to no maintenance requirements and their ability to handle data diversity. However, various existing machine

Table 1: List of abbreviations.

AMI	Advanced metering infrastructure
ANN	Artificial neural network
AUC	Area under curve
CFSFDP	Clustering technique by fast search and find of density peaks
CNN	Convolutional neural network
DNN	Deep neural network
DT	Decision tree
EBT	Ensemble bagged tree
EC	Electricity consumption
ETD	Electricity theft detection
GA	Genetic algorithm
GMM	Gaussian mixture model
MODWPT	Maximal overlap discrete wavelet packet transform
GRU	Gated recurrent unit
HDNN	Hybrid deep neural network
LR	Logistic regression
LSTM	Long short term memory
MEPCO	Multan electric power company
MIC	Maximum information coefficient
ML	Machine learning
NaN	Not a number
NN	Neural network
NTLs	Non-technical losses
PCA	Principal component analysis
PSO-GRU	Particle swarm optimization GRU
RE	Relative entropy
RUSBoost	Random undersampling boosting
SMOTE	Synthetic minority oversampling technique
SGCC	State Grid Corporation of China
SVM	Support vector machine
TLs	Technical losses

and deep learning techniques proposed in the literature face the overfitting problem [17].

The authors in [3] proposed a classification technique based on ensemble bagged tree (EBT) for detecting NTLs in power grids. The proposed technique handles the electricity loss issue in Multan Electric Power Company (MEPCO), Pakistan. In the proposed work, the technique is validated in terms of various performance metrics and is found more efficient than the existing techniques. In [20], the authors proposed a hybrid model for ETD that is based on long short-term memory (LSTM) and CNN. The model is compared with other models, and the results show that it beats the existing models and achieves high accuracy. The proposed model used in [21] is based on the relative entropy (RE) along with principal component analysis (PCA). This work is aimed at detecting the electricity losses, which occur in the vicinity of AMI, using the reconstructed data. The model is evaluated for sensitivity and specificity, and results indicate the good performance of the model for ETD.

In [22], the authors used fuzzy logic technique for the detection of suspicious electricity consumers. The selected time series data is linked with consumers, and fuzzy sets of suspicion are created. Based on these fuzzy sets, a threshold value is decided, which helps in the detection of suspicious consumers. The proposed technique's performance is examined in terms of curve membership function, and the results show that it performs better than benchmark techniques. Similarly, the authors in [23] presented fuzzy logic technique to detect electricity theft and to increase the reliability of the power grid. The proposed technique is evaluated by presenting sixteen real-world scenarios. Efficiency of the technique is evaluated in terms of classifying honest and fraudulent consumers. However, integrating the renewable sources with power grids is not handled well. Similarly, the authors in [24] extracted the EC behavior patterns of the users and detected the abnormal consumption behavior. The authors in [25] proposed a deep learning model to overcome the issues related to NTLs in smart grids. This model considered an unlabeled data and an adversarial model to mitigate the

overfitting issue. It performs better than the existing models. In [26], a hybrid approach is presented for ETD, which is based upon Gaussian mixture model (GMM) and LSTM. In this work, actual time series data is considered and some improvements are made in the LSTM structure. The simulations are carried out to show the performance of the proposed approach in terms of ETD.

In [27], the authors proposed a maximal overlap discrete wavelet packet transform (MODWPT) based model for feature extraction and random undersampling boosting (RUSBoost) technique to detect NTLs in the power grids. A comparison of the proposed and existing techniques is done, and results indicate the efficient performance of the proposed technique for NTL detection. The model is compared with the benchmark techniques, and the results show that the proposed technique outperforms the existing techniques in terms of NTL detection. The authors in [28] established a relationship between commercial losses and characterization of irregular consumers using black hole algorithm. Two different datasets are used in this work provided by Brazilian electric utility, and theft categorization is performed.

The authors in [29] presented a clustering-based approach for the detection of electricity thefts. The approach is based on maximum information coefficient (MIC) and clustering technique by fast search and find of density peaks (CFSFDP). Irish smart meter dataset is used in this work for carrying out the simulations. It is observed that the proposed model performs efficient theft detection. Similarly, a clustering approach is used by the authors in [30] to divide the consumers into clusters on the basis of load consumption and perform efficient short-term load forecasting. The authors in [31] adopted the LSTM method for forecasting EC of the consumers based on the recent past consumption profiles. The continuous monitoring of the profiles helps in efficient ETD. The simulation results prove the model's efficiency. The authors in [32] used big data analytics for forecasting the EC along with the corresponding price. The simulations are performed to prove the model's efficiency in terms of price forecasting. The authors in [33] proposed a fault-tolerant model to preserve the privacy of users and perform data aggregation at the smart grids. Table 2 summarizes the related work in a tabular form for better understanding.

3. Proposed System Model

In this section, the proposed system model is described along with the dataset used in this work. Furthermore, different techniques used in this work are discussed. The proposed system model is shown in Figure 1.

3.1. Description of the Proposed Model. The proposed HDNN-based model consists of several steps, discussed as follows. Initially, the data is gathered from the intelligent antenna-based smart meters installed at the consumers' end, shown in the lower box of the proposed system model. Each smart home has a smart meter and an intelligent antenna, which helps in recording the EC data. The data is saved and is made publicly available by SGCC. Afterward, the dataset is preprocessed in order to normalize the data and to remove redundant and irrelevant data. Also, the outliers are removed to get more refined data. The most important features are obtained by performing feature engineering process. In this process, feature selection and extraction are done. Then, the classification of normal and fraudulent consumers is done. Figure 2 shows the flow of data acquired from the smart homes having intelligent antenna-based smart meters. In Table 3, the identified limitations are mapped with their respective solutions and validations.

(1) *Description of the Dataset*. In this work, real-time EC data of the users is used, provided by SGCC [19]. The data is gathered from the consumers' side using intelligent antenna-based smart meters. The dataset consists of 1,035 features. A subset containing data of 3000 consumers is selected from the whole dataset, in which 2480 are normal consumers; while the remaining 520 consumers are fraudulent. It can be clearly observed that the dataset is imbalanced, due to which ETD is highly affected. In this work, the data is balanced using SMOTE [20], which balanced the number of fraudulent and normal consumers. In addition, the dataset is divided into 75% and 25%, respectively, for training and testing purposes. Table 4 gives a detailed description of the dataset used in the proposed work

(2) *Synthetic Minority Oversampling Technique*. It is considered as one of the oversampling techniques, which increases the data points in the minority class (fraudsters) in order to handle imbalanced data problem. In SMOTE, the synthesized data points are generated in the minority class. In this work, the highly imbalanced data is balanced by using SMOTE. In SMOTE, if (x_1, x_2) depicts a sample of a minority class, then (x_1', x_2') is selected as its nearest neighbors. The synthesized or fake data points are generated by the following equation.

$$(X_1, X_2) = (x_1, x_2) + \text{random}(0, 1) * \Delta, \tag{1}$$

where random $(0, 1)$ presents a number that is chosen between 0 and 1 and Δ denotes the Euclidean distance between the minority class and its neighboring class sample. It is calculated in

$$\Delta = \left(x_1' - x_1\right), \left(x_2' - x_2\right). \tag{2}$$

3.2. Data Preprocessing. The major aim of preprocessing step is to get the most refined data from the whole dataset. In this step, the missing or Not a Number (NaN) values are recovered by local average method, which is given in the following equation [7].

Table 2: Summary of related work.

Proposed solutions	Performance metrics	Limitations
EBT [3]	Sensitivity, specificity, false-positive rate, and $F1$-score	Increased computational time
CNN and LSTM [20]	$F1$-score, area under curve (AUC), precision, and recall	No normalization and parameter tuning
PCA and RE [21]	ROC, specificity, and sensitivity	PCA only works for linear data
Fuzzy logic [22]	Generalized bell curve membership function	Increased computational time
Fuzzy logic [23]	Accuracy, $F1$-score, AUC	Issues related to renewable sources are not handled
Semisupervised deep neural network (DNN) [25]	Precision, true and false-positive rates, recall, and $F1$-score	High false-positive rate
LSTM and GMM [26]	AUC, MCC, recall, and accuracy	Data imbalance is not handled
MODWPT and RUSBoost [27]	$F1$-score, AUC, and precision	Oversampling issue is not tackled
Blackhole algorithm [28]	Average execution time and convergence	High false-positive rate
MIC and CFSFDP [29]	$F1$-score, precision, and recall	Low precision and recall
LSTM [31]	Accuracy, sensitivity, and specificity	Overfitting is not handled well
LSTM and regression [32]	$F1$-score, recall, and precision	Low $F1$-score

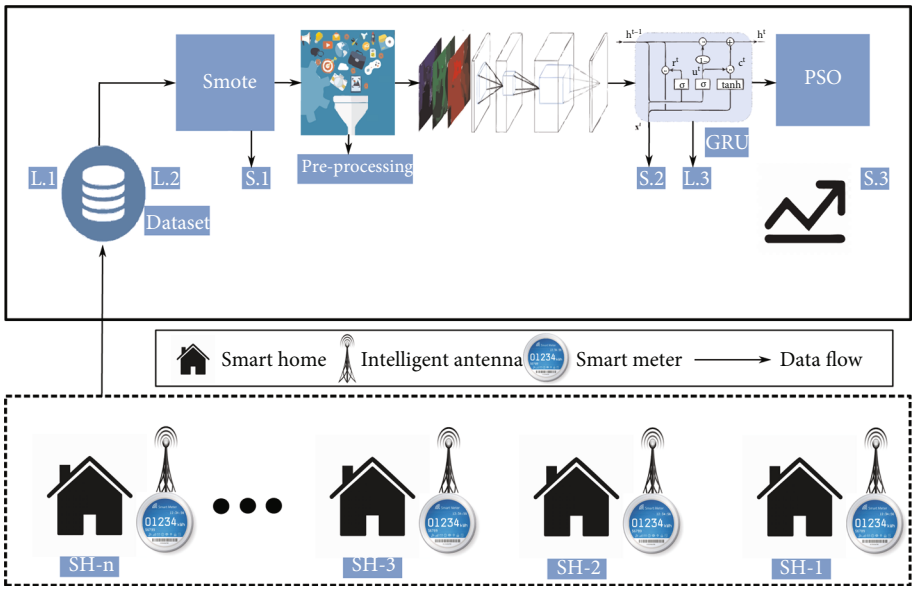

Figure 1: The proposed system model to process data coming from intelligent antenna-based smart meters for users' classification.

$$f(x_i) = \begin{cases} \sum_{k=i-1}^{i+5} P_k \mu_k * \text{Average}_{\text{local}} & \text{if } x_i = \text{NAN}, \\ x_i & \text{if } x_i \notin \text{NAN}, \end{cases} \quad (3)$$

when the value of NAN is continuous, then 0.10 will be the value of P_k. x_i represents the EC of a user at specific time interval. μ_k has a binary value, i.e., either 0 or 1, which is based on threshold k and is calculated using the following equation.

$$\mu_k = \begin{cases} 1 & \text{if } x_k \geq \text{Average}_{\text{local}}, \\ 0 & \text{if } x_k < \text{Average}_{\text{local}}, \end{cases} \quad (4)$$

where $\text{Average}_{\text{local}}$ is computed in the following equation

$$\text{Average}_{\text{local}} = \frac{1}{10} * \sum_{i-5}^{i+5} f(x_i). \quad (5)$$

The min-max normalization technique is used to transform and normalize the dataset in the range of [0, 1]. The minimum value is transformed into 0; whereas the maximum value is transformed into 1 and other values are transformed between 0 and 1. The min-max technique is calculated by the equation given below,

$$B = \frac{A - \min(A)}{\max(A) - \min(A)}. \quad (6)$$

A refers to the actual value of the features and B indicates the value after normalization. While max (A) is the

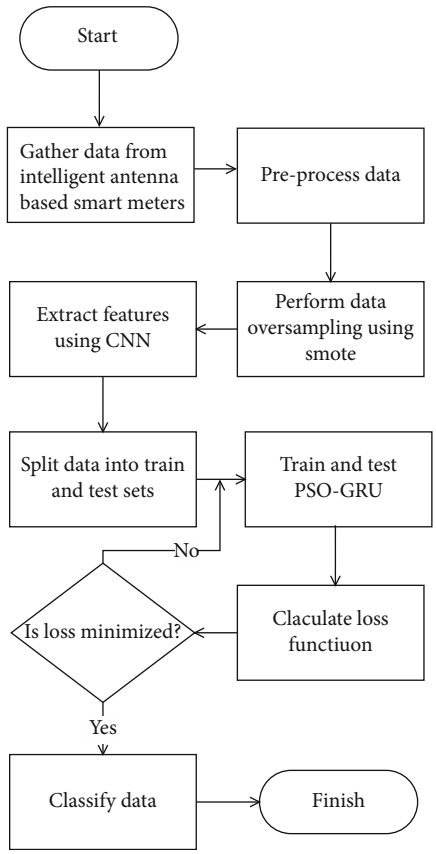

Figure 2: The flow of data gathered from intelligent antenna-based smart meters.

maximum value of A, and min (A) is the minimum value of A. The equations used above are adopted from [20].

3.3. Feature Engineering. Once the data is preprocessed and normalized, the feature engineering process is performed. This process includes two steps: one is feature selection and another is feature extraction. The former selects the most relevant data features from the whole dataset to reduce both overfitting and training time and to improve accuracy; whereas the latter extracts the selected features for data dimensionality reduction and removal of data redundancy. With the feature engineering process, the performance of the model is enhanced.

In this work, feature engineering is done using a DNN, termed as CNN. The idea of CNN was primarily presented in [20]. The typical CNN architecture has various layers, which include convolution, pooling, and fully connected layers. The first convolution layer contains many convolution filters, which are termed as kernels and they perform mapping operation. The convolution layer is mathematically given in the following equation [34]

$$y_{\text{conv}}\left(X_t^{\text{ft}}\right) = \sigma\left(\sum_{\text{ft}=1}^{\text{ft}} W_t^{\text{ft}} * X_t^{\text{ft}} + b_t^{\text{ft}}\right), \tag{7}$$

where σ refers to the activation function and $*$ represents the convolution operation. W_t^{ft} and b_t^{ft} represent the learnable parameters in the fth feature filter. The next layer in CNN is the pooling layer, which comes after the convolution layer. The main objectives of this layer are to extract the meaningful features and to perform the downsampling of each feature map to achieve dimensionality reduction. It also reduces the execution time of the network. The pooling layer consists of two common functions, which are as follows.

(i) *Average Pooling*. This function calculates the average number of features in the feature map

(ii) *Maximum Pooling*. This function calculates the maximum number of optimal features in the feature map

In CNN, the third fully connected layer performs the final classification. In ETD, the data is classified into honest and fraudulent consumers' classes. The mathematical representation of the fully connected layer is given in Equation (8), taken from [34],

$$Y_{\text{ft}}(X_t) = \sigma(W_t.X_t + b_t), \tag{8}$$

where W represents weight and b represents bias.

Function that is used in CNN to predict the final output is known as the Softmax function. The output is given in binary form, either 0 or 1 [20]. Equation (9) provides a complete mathematical form of CNN, as given in [34]. Figure 3 gives an overview of the architecture of CNN,

$$A(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left|y_{W,b}(x_i) - y_i\right|^2\right). \tag{9}$$

The parameters involved in Equation (9) are initialized with some random number using the normal distribution. m denotes the total number of training example. Initially, the weight W_i, j is assigned randomly, then later it is updated using the gradient descent method. Equations (10) and (11) correspond to $W_{ij}^{(l)}$ and $b_i^{(l)}$,

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b), \tag{10}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b), \tag{11}$$

where α represents the learning rate, ∂ denotes partial derivative, $W_{ij}^{(l)}$ is the connection weight between ith neuron in the lth layer and jth neuron in the $(l+1)$ th layer. b is the bias of the ith neuron in the lth layer. Equations (10) and (11) are repeated until the optimal value of objective function $A(W,b)$ is achieved. The above mathematical representations are motivated from [34]. The hyperparameters of CNN used in this work and their values are given in Table 5.

3.4. Gated Recurrent Unit. It is considered as a variation of LSTM and recurrent neural network (RNN) and is a subclass of DNN. It resolves the vanishing gradient problem of RNN

Table 3: Mapping of limitations with solutions and validations.

Limitations	Proposed solution	Validation
Inability to handle imbalanced data (L.1)	Synthetic minority oversampling technique (SMOTE) (S.1)	SMOTE has balanced the data, shown in Table 4 (V.1).
Overfitting (L.2)	CNN-GRU (S.2)	The validation is given in Figures 7 and 8 (V.2)
High false-positive rate (L.3)	PSO (S.3)	The validation is given in Figures 9 and 10 (V.3)

Table 4: Description of dataset.

Hyperparameters	Values
Original dataset size	42372
No. of normal users	38757
No. of fraudulent users	3615
Selected dataset size	5960
No. of selected normal users	2480
No. of selected fraudulent users before SMOTE	520
No. of selected fraudulent users after SMOTE	2480

by using two gates: update gate and reset gate. These gates determine that how much information is required to pass to the future. In the updated gate, the past or previous information needed to be passed to the future is determined. Equation (12) gives the formula to calculate the output of the updated gate z_t for time series data, taken from [35].

$$z_t = \sigma(W^z x_t + U^z h_{t-1}), \tag{12}$$

where x_t shows an input that is given to the network unit and is multiplied by its weight W^z. The h_{t-1} maintains the previous information and is multiplied by its weight U^z as well. Then, these weights are summed up and the result is squashed between 0 and 1 by applying the sigmoid function. The reset gate decides that how much previous information is required to be neglected. Equation (13) gives the mathematical form of the reset gate, taken from [35].

$$z_t = \sigma(W^r x_t + U^r h_{t-1}), \tag{13}$$

where x_t is multiplied by its weight W^r, and the h_{t-1} is multiplied by its weight U^r. Figure 4 shows the architectural view of GRU. In Table 6, the hyperparameters of GRU used in this work along with their values are presented.

3.5. Particle Swarm Optimization. It is a population-based stochastic technique that handles the local optima issue by covering the search space with global optimum solutions. The traditional ML techniques, such as GRU, LR, and SVM, are mostly stuck into local optima. That is why it is not suitable to utilize such techniques for ETD due to their poor ETD performance. In this work, a hybrid technique is made by integrating PSO with GRU to perform efficient and accurate ETD. The proposed technique overcomes the local optima issue very efficiently. PSO performs the searching operation via swarm particles, which are updated in every next iteration. The best optimal solution is achieved by moving each particle in the direction of previous best $\text{pbest}(i,t)$ and global best $\text{gbest}(t)$ solutions in the swarm [18]. Equations (14) and (15) give the mathematical form of calculating pbest and gbest, respectively.

$$\mathbf{pbest}(\mathbf{i},\mathbf{t}) = \arg\ \min_{k=1,\cdots,t} \min\ [f(P_i(k))], i \in 1,2,\cdots,N_p, \tag{14}$$

$$\mathbf{gbest}(\mathbf{t}) = \arg\ \min_{i=1_{k-1,\cdots,t},\cdots,N_p} [f(P_i(k))], \tag{15}$$

where i indicates the particle index, t gives the current iteration number, N_p gives the total number of the particles, f represents the fitness function, and P tells the position. The velocity V of a particle is updated using the following equation.

$$\begin{aligned} V_i(t+1) = {} & \omega V_i(t) + c_1 r_1(\mathbf{pbest}(\mathbf{i},\mathbf{t}) - p_i(t)) \\ & + c_2 r_2(\mathbf{gbest}(\mathbf{t}) - p_i(t)), \end{aligned} \tag{16}$$

where ω is a weight of inertia that is used to balance both global and local exploitation. Whereas r_1 and r_2 indicate the uniformly distributed random variables that are in the range of [0,1]. While c_1 and c_2 represent the positive constant parameters, which are also known as acceleration coefficients. The hyperparameters of PSO used in this work and their values are given in Table 7. The above given mathematical formulations of PSO are taken from [18]. Algorithm 1 gives the pseudocode of PSO.

The efficiency of the presented hybrid model is optimized by passing three parameters of GRU to PSO. Based on these parameters, the training and testing processes of the model are optimized for given dataset. As a result, the model becomes accurate and more robust. These parameters are discussed below.

(i) *Hidden Layer*. It is considered the most important layer of GRU. It is positioned between input layer and output layer. This layer primarily performs the computational operations. Moreover, the weights are given to input values by this layer. After successfully accessing optimal input sets, the results are passed to the output layer for final predictions

(ii) *Batches*. They determine the number of training samples required to compute training and testing loss. Generally, the loss is calculated by the predefined loss function

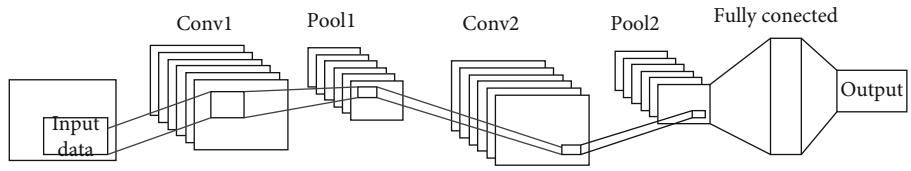

Figure 3: CNN architecture.

Table 5: Hyperparameters selection for CNN through PSO.

Hyperparameters	Selected values	Range of values
Dropout	0.2	0.1, 0.2, 0.3, 0.4
Batch size	25	25, 50, 64, 128
Optimizer	Adam	Adam, SGD, RMSE
Kernel size	3	1, 3, 5
Epochs	6	4, 6, 8, 10
No. of neurons	500	200, 300, 400, 500
No. of hidden layers	5	3, 5, 7, 9, 11
No. of convolution layers	32	4, 8, 16, 32
No. of fully connected layers	1	1, 2, 3, 4

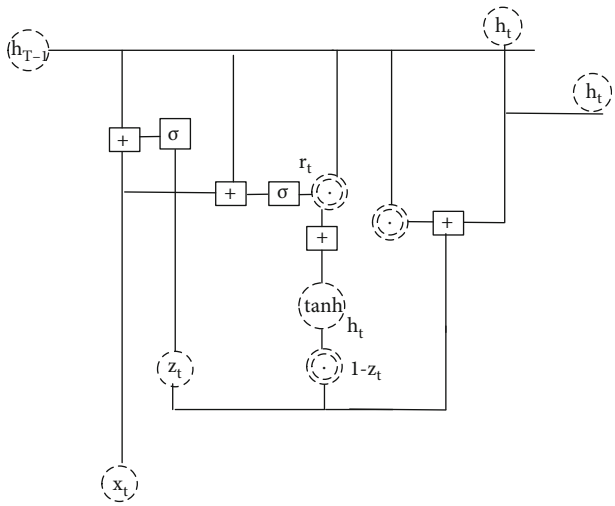

Figure 4: GRU architecture.

Table 6: Hyperparameters selection for GRU through PSO.

Hyperparameters	Selected values	Range of values
Learning rate	0.01	0.1, 0.01, 0.001
Batch size	25	25, 50, 64, 128
Epochs	4	4, 6, 8, 10
Optimizer	Adam	Adam, SGD, RMSE'
No. of neurons	300	100, 200, 300, 400
No. of fully connected layers	1	1, 2, 3, 4
No. of hidden layers	5	3, 5, 7, 9, 11

Table 7: Hyperparameters of PSO.

Hyperparameters	Values
Population size	5960
C1 and C2	2, 2

(iii) *Learning Rate*. It establishes the adjustment of the network weights, which are needed regarding the loss gradient

3.6. Particle Swarm Optimization-Gated Recurrent Unit Binary Classification. A hybrid deep model is presented in this work that is based on PSO-GRU. In the existing models, overfitting and parameter tuning are the main problems. These problems further lead to increase in false-negative rate and false positive rate (which are given in Equations (19) and (20)). Therefore, a hybrid PSO-GRU model is presented to overcome the aforementioned issues. The pseudocode of the model is given in Algorithm 2. The parameters of GRU are tuned using PSO. Afterward, the well-tuned model is used for classification. The main purpose of PSO is to improve the learning of the GRU network and to solve overfitting problem. In this whole process, the data is initially preprocessed. In this phase, local average method is used for recovering missing values; whereas min-max normalization is applied to scale the data. Afterward, data balancing is performed using SMOTE oversampling technique, which balances the provided data in different classes by generating synthetic samples of the minority class. If the classifier is trained on imbalanced data, then it is biased towards the majority class; therefore, the data is balanced using SMOTE technique. Afterward, useful features are extracted from the dataset by CNN. The extracted features are then passed to GRU for training. The parameters of GRU are tuned using PSO. Finally, the fine-tuned model is used to perform classification.

4. Simulation Results and Discussion

Several simulations are conducted to assess the performance of the proposed hybrid model. The simulation results, performance metrics, and benchmark models are discussed in this section.

4.1. Performance Metrics. The performance of the proposed model is examined by considering several performance measures, which include AUC, accuracy, *F*1-score, recall, and precision. The training and testing loss and accuracy are also calculated to assess the performance of the proposed and the

1: Initialization
2: **for** each particle i =1,…,Np, **do.**
3: (a) Initialize the particle's position using uniform distribution as $p_i(0)$ and *U(LB, UB)* where UB and LB represent the upper bound and lower bound of the search space, respectively.
4:
(b) Initialize **pbest** to its initial position:**pbest**(i,0) = $p_i(0)$
5: (c) Initialize **gbest** to the minimal value of the swarm.
:gbest(0) = $p_i(0)$.
6: (d) Initialize Velocity: Vi U(—UB-LB—),—UB-LB—
7: **end for.**
8: Repeat until the Termination Criteria Is Met
9: **for** each particle i =1,…,NP, **do.**
10: (a) Pick random numbers: r1,r2 U(0,1).
11: (b) Update the particle's velocity. See Equation (16).
12: **end for.**
13: **if** f[p_i(t)] < f[**pbest(i,t)**], **then.**
14: Update the best Known Position of Particle I: **Pbest(I,T)** = p_i(T)
15: **if** f[p_i(t)] < f[**gbest(t)**], **then.**
16: Update the swarm's best Known Position: **Gbest(T)** = p_i(T)
17: **end if.**
18: **end if.**
19: Output **gbest(t)** that holds the best found solution.
20: End.

Algorithm 1. Pseudocode of PSO.

1: Initialization
2: Load the dataset D_s from the smart meter S_m
3: Perform pre-processing steps
4: Split D_s into D_{st} (train dataset), D_{ste} (test dataset)
5: Adjust the parameter of the classifier such that
$b \in B, h \in H, n \in N$ and $e \in Ep$. Where b is the number of the batches, h is the number of hidden layers, n is the number of neurons in the hidden layer and e is the number of epoch set for training and testing phases 6: Pass the *h, b and lr* parameters to the PSO 7: Steps in training phase:
8: Build $g_ruPSO \in PSO - GRU$ with the network parameters *b, h* and *lr* for the D_s
9: Repeat
10: At the n^{th} epoch do
11: Train the g_ruPSO to fetch b from D_s
12: Proposed model's performance is measured in terms of AUC, accuracy, precision, recall and F1-score
13: Update the performance of the *queue* ∈ Q. where $Q \in$
P$Push_{opt}$or$Q \in$ PPop_{opt}
14: End
15: The loss function is calculated until loss function $ls \leq Cn$. Where Cn is the convergence threshold
16: If early stopping is performed, then
17: Proceed
18: Else, go to step 8
19: Steps in testing phase:
20: Fetch D_{ste} from the D_s
21: Test is performed by g_ruPSO on the D_{ste} to generate the final results through plots
22: Compare the proposed and benchmark models on the basis of D_{ste} in terms of performance measures for final prediction
23: End

Algorithm 2. Pseudocode of the proposed HDNN model.

benchmark models. A detailed description of the performance metrics is given below.

(1) *Area Under Curve*. This performance metric is used for the validation of model by considering an AUC between two integrals. Moreover, it provides the accumulative performance of the binary classes. Generally, the value of AUC is either 0 or 1. Where 0 value means that the performance of the model is poor; whereas 1 indicates the best performance.

Equation (17) is used to compute the value of AUC [20], as given below

$$\text{AUC} = \frac{\sum \text{Rank}_{i \epsilon \text{positiveclass}} - M(1+M)/2}{M * N}, \quad (17)$$

where the rank values of samples are indicated by Rank_i. M represents the total positive samples, and N represents the negative ones.

(2) *F1 -Score*. It is referred as *F*-measure as well. It calculates the testing accuracy of the model and assesses the testing score using recall and precision. The *F*1 -score is calculated using the following equation [20]

$$F1 - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (18)$$

(3) *Recall and Precision*. The recall and precision are expressed using the following equations

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}, \quad (19)$$

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}, \quad (20)$$

where recall calculates the number of true positives in all of the classifier's results. True positive means correctly classified energy thieves. False-positive means honest electricity consumers misclassified as thieves. False-negative means energy thieves misclassified as honest consumers and true negative means correctly classified honest consumers. Whereas precision is the measure of relevance of the classifier's results. If precision is high, it means that classifier's relevant results are more than the irrelevant results.

4.1.1. Case Study A. In this case study, the SGCC dataset is considered, which is publically available on internet. The brief description about dataset is given in Section III-A1 [Page 3].

4.2. Description of Existing Models with their Performance. This section presents the existing models along with their performance based on the abovementioned performance metrics.

(1) *Support Vector Machine Model.* SVM is widely used in ETD for binary classification [36]. The performance metrics used for SVM, and their results are given in Table 8. It performs better than LR model; however, its performance is worse than LSTM, GRU, and PSO. It means that SVM is less accurate in handling the imbalanced data.

(2) *Logistic Regression Model.* LR is a popular classifier that is widely used for both classification and regression. In literature, it is also used for ETD [37]. In this work, the performance of LR is examined in terms of aforementioned performance metrics. The results of LR are given in Table 9, which show that it performs worse than all of the benchmark techniques. The reasons behind this are overfitting issue and inability of LR to handle imbalanced ETD data

(3) *Long Short Term Memory Model.* LSTM is a DNN model and is widely used for feature extraction and classification in ETD [38, 39]. The performance of LSTM is checked in terms of abovementioned metrics. The results are presented in Table 10, which show that LSTM performs better than SVM and LR and worse than the proposed model. The proposed model performs better than LSTM because it does not face overfitting problem, which degrades the performance of LSTM

(4) *Gated Recurrent Unit Model.* GRU is also a DNN model [40]. It is an advanced version of the LSTM. Its results are shown in Table 11, which are better as compared to the benchmarks. It means that GRU is capable of handling imbalanced data while avoiding overfitting

(5) *Genetic Algorithm Model.* Genetic algorithm (GA) is a metaheuristic technique. Its performance is assessed based on various performance metrics, as given in Table 12. The results show that GA performs better than the benchmark models: SVM, LR, LSTM, and GRU. GA is also found to be more accurate and robust than the benchmarks because of its better learning capability

4.3. Results. The simulations are performed to evaluate the proposed and benchmark models by considering the aforementioned performance measures along with loss and accuracy. Moreover, the models are integrated for the performance evaluation. The SVM and LR obtain 0.68% and 0.63% of ACU score, respectively. SVM has higher AUC score as compared to LR because of using kernel trick to cope with the nonlinear data. In contrast, LR has lowest AUC score of 0.63% because it has only one hidden layer, which did not handle high dimensional data effectively and stuck in local minima. The performance results of SVM and LR are given in Tables 8 and 9, respectively.

Figures 5 and 6 show the accuracy and loss values of combined CNN-LSTM model. The CNN is utilized to extract abstract and latent features from EC data with the help of convolutional and pooling layers. Whereas LSTM extracts temporal patterns and classifies consumers' records into normal and abnormal data patterns. From Figure 5, the training accuracy is observed as 81%; whereas the testing

Table 8: Results of SVM.

Performance evaluation metrics	Results
AUC	68%
*F*1-score	75.89%
Accuracy	76.67%
Precision	77.43%
Recall	78.82%

Table 9: Results of LR.

Performance evaluation metrics	Results
AUC	63%
*F*1-score	67.93%
Accuracy	72.09%
Precision	69.02%
Recall	73.63%

Table 10: Results of LSTM.

Performance evaluation metrics	Results
AUC	81.7%
*F*1-score	79.2%
Accuracy	80.11%
Precision	81.20%
Recall	81.22%

Table 11: Results of GRU.

Performance evaluation metrics	Results
AUC	83.7%
*F*1-score	82.98%
Accuracy	83%
Precision	83.98%
Recall	84.23%

Table 12: Results of CNN-GA-GRU.

Performance evaluation metrics	Results
AUC	87%
*F*1-score	87.22%
Accuracy	87%
Precision	88.1%
Recall	88.72%

accuracy of the model is 75.5%. Both accuracies increase for an increasing number of epochs, which shows that model gives good results on a large number of epochs. However, the model is trained only for four epochs due to limited resources. There is a 6% difference between training and testing accuracies curves, which indicates that the model is stuck into an overfitting problem. On the other hand, Figure 6 shows that the training and testing losses of the

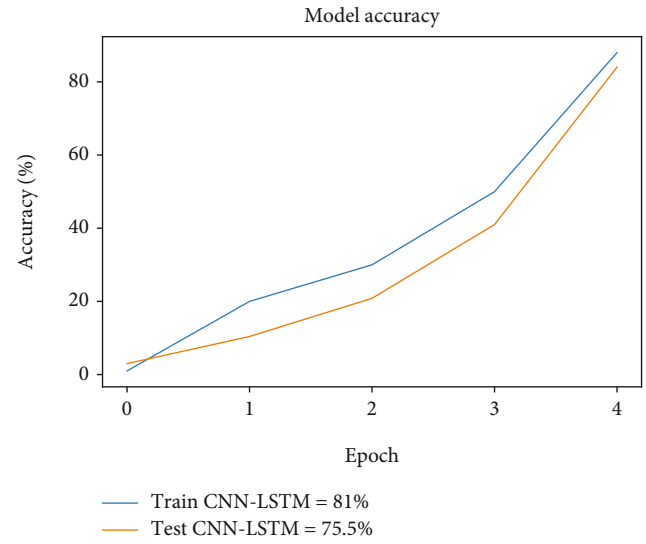

Figure 5: CNN-LSTM accuracy.

model decrease as number of epochs increase. After the 4th epoch, the training and testing loses are 19% and 24.5%, respectively.

This implies that the epoch is the main controlling parameter that decides the optimal point where model achieves higher performance. However, the CNN-LSTM obtains 75.5% test accuracy, which is neither satisfactory nor acceptable in ETD. This is happened due to the inappropriate selection of hyperparameters. For deep learning models, the suitable selection of hyperparameters has great influence on the performance results.

Figures 7 and 8 show accuracy and loss values of combined CNNGRU model on training and testing datasets. The CNN is used to extract optimal features while classification task is performed through GRU model. The GRU model has reset and update gates that extract more relevant information from extracted high variance features through CNN and remove the noisy and redundant features. This process makes the performance of CNN-GRU better than CNN-LSTM. There is a 4% difference between accuracy curves and a 6.97% difference between loss curves on training and testing datasets, which indicate that the CNN-GRU model is stuck into an overfitting problem. The inappropriate tuning of hyperparameters leads to an overfitting problem where the model gives good results on seen data as compared to unseen data.

In literature, there are different techniques to tune the hyperparameters of ML and deep learning models like random search, grid search, gradient-based optimization, and evolutionary algorithms. Each one of these methods has its pros and cons. In this study, we utilize evolutionary algorithms PSO and GA to find optimal hyperparameters of the CNN-GRU model. These algorithms make a search space of hyperparameters and try to find the optimal combination where the model gives high values of performance indicators.

The PSO is merged with CNN-GRU for hyperparameters tuning to enhance the performance of the proposed model. The results are shown in Figures 9 and 10. The

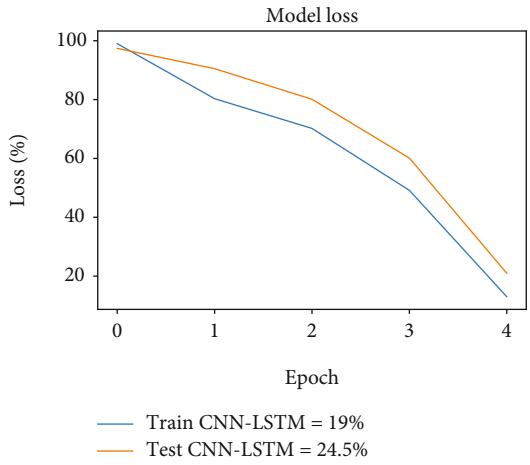

Figure 6: CNN-LSTM loss.

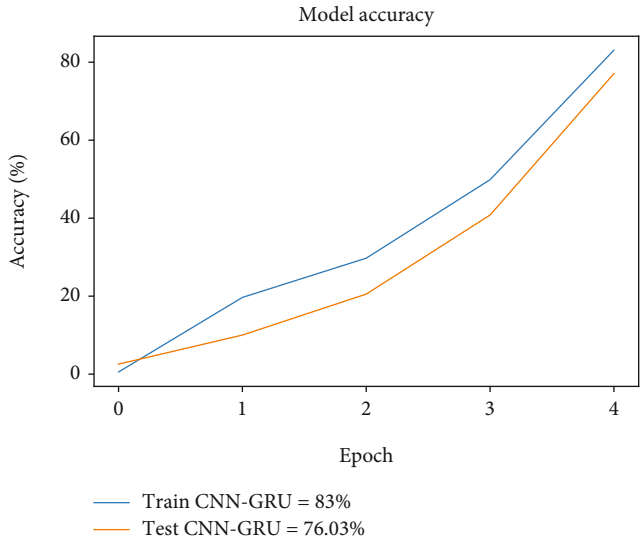

Figure 7: CNN-GRU accuracy.

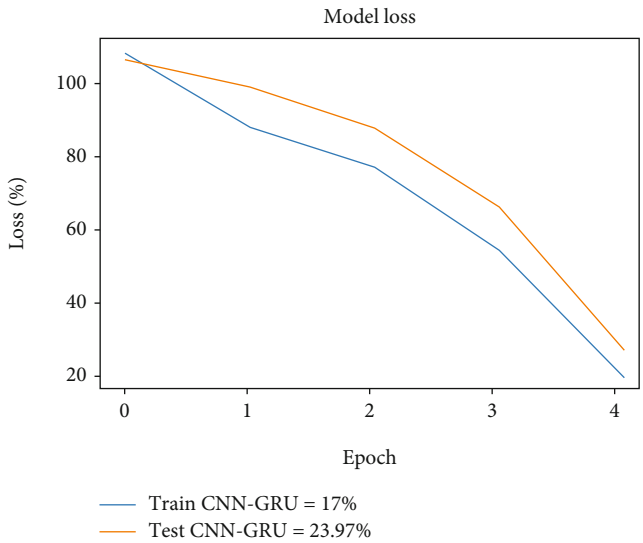

Figure 8: CNN-GRU-loss.

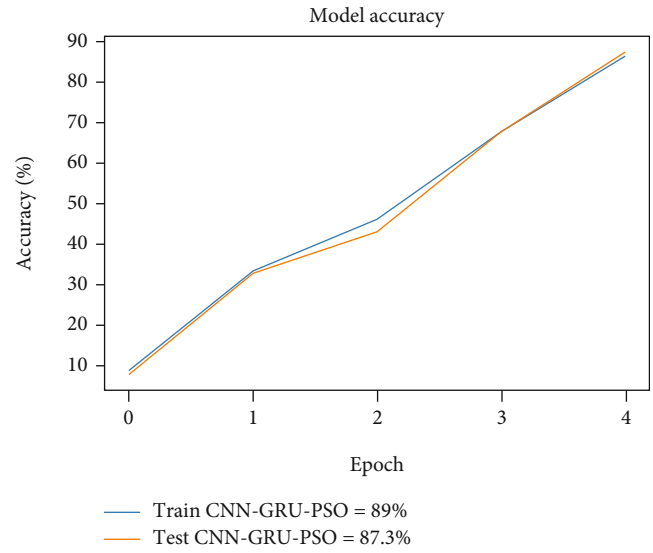

Figure 9: CNN-GRU-PSO accuracy.

former shows the training and testing accuracy of CNN-GRU, which is increased with the increase in the number of epochs. Whereas the training and testing loss are presented in the latter figure and the results show that the loss decreases with the increase in the number of epochs. Furthermore, CNN-GRU is integrated with GA to find optimal combination of hyperparameters. The performance of combined CNN-GRU-GA is presented in Figures 11 and 12. The training and testing accuracies of CNN-GRU-GA are shown in Figure 11. The figure shows that both training and testing accuracy are approximately 87% and 86.3%, respectively. Figure 12 presents the training and testing loss of CNNGRU-GA, which keep decreasing with the increasing number of epochs. As shown in the figure, the training loss and the testing loss are 13% and 13.7%, respectively, which are approximately the same. Next, training and testing accuracy and loss of hybrid CNNGRU-PSO model are evaluated in Figures 9 and 10, respectively. Here, PSO and GA are used for tuning the hyperparameters of the CNNGRU model. The results exhibit that the PSO obtains optimal set of parameters as compared to GA because the PSO require less number of parameters and less execution time. In PSO, each solution has its own local best, which leads it towards global best after each iteration. Whereas GA has crossover and mutation steps that create diversity in newly generated offsprings and prevent the model from falling into local optima problem. However, in this case, PSO performs better as compared to GA and gives optimal combination hyperparameters where CNN-GRU gives good results. The former shows that both training and testing accuracy of CNN-GRU-PSO increase with the increasing number of epochs. The training accuracy is 89%; while testing accuracy is 87.3%. Whereas in latter, the training and testing loss are given, which are 11% and 12.7%, respectively. Moreover, CNN is combined with GRU and the performance of CNN-GRU is evaluated in terms of training and testing accuracy and loss. In Figures 5–10, the performance of the combined models is depicted.

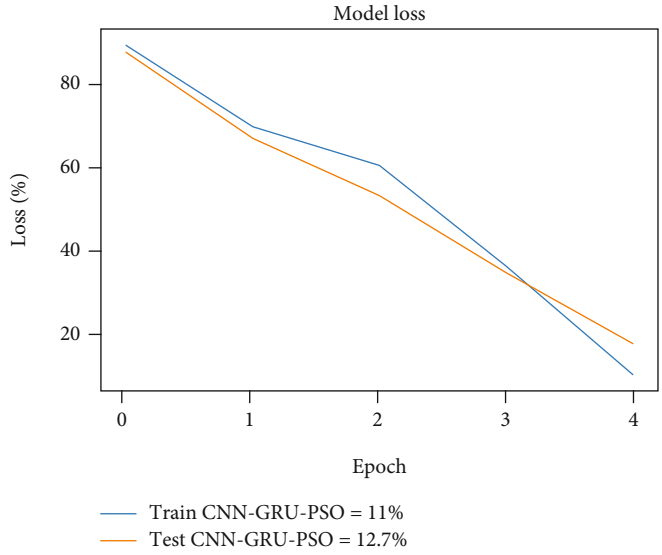

Figure 10: CNN-GRU-PSO loss.

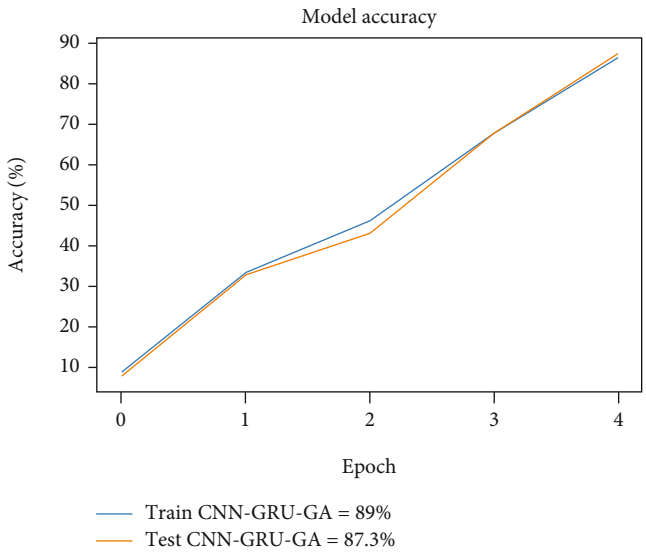

Figure 11: CNN-GRU-GA accuracy.

It is observed that the CNN-GRU-PSO has the maximum accuracy as compared to the other models. Similarly, the CNN-GRU-PSO model has a minimum loss, which shows the model's generalization. Figures 13 and 14 show the combined performance of the used and existing models for AUC. In former, AUC is calculated using both the false positive rate and true positive rate. The results indicate that the proposed CNN-GRU-PSO achieves high AUC score. Whereas the other models have a low AUC score. Moreover, it is shown that the proposed model beats the existing ones regarding AUC in the presence of imbalanced dataset. The GRU module in proposed model has strong ability to learn temporal correlation from long-term electricity load profile of consumers. It also maintains the context of previous EC information, which helps out to handle any nonmalicious (weather condition, family structure, etc.) change in EC profile. Moreover, the integration of PSO for parameters tuning further enhance the performance of the proposed model towards efficient ETD. Furthermore, the proposed model is compared with the benchmark models in terms of mentioned performance metrics, and the result is shown in Figure 15. The result indicates that the hybrid CNN-GRU-PSO model is more robust, accurate, efficient, and more generalized than the benchmarks, as given in Table 13.

Table 14 describes the running time of the proposed and baseline models. The SVM model takes 220 s during the training phase, which is higher than all other schemes. The selected SGCC dataset is high dimensional and not linearly separable. SVM draws $n-1$ hyperplanes and then picks an optimal hyperplane of high margin for distinguishing two classes (n represents the number of dimensions).

So, that is why SVM takes higher time as compared to other models.

The LR takes lowest execution time because of its simplex layering structure. It has only one hidden layer of neural network (NN). So, it needs less weights to learn and consumes less time as compared to other deep learning models. The CNN-GRU takes 56 seconds running time in training phase, which is 10 seconds less than CNNLSTM because of less gated configuration as compared to LSTM. The proposed CNN-GRU-PSO has higher execution time as compared to other models because of using PSO for tuning the hyperparameters of both CNN and GRU models concurrently.

4.3.1. Case Study B. In this case study, the PRECON dataset is used, which is publically available on internet. The dataset is collected by Pakistan Residential EC company. This dataset contains the EC history of 42 residential houses for 365 days. In dataset, the EC of each user is recorded after one minute time period. However, in this work, the data granularity is reduced into half hour for ease. EC of 30 minutes is aggregated into single value for all dataset. All the consumers

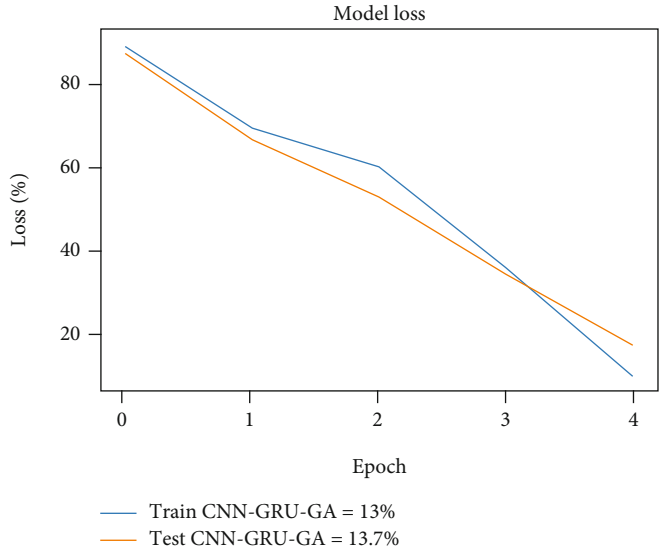

Figure 12: CNN-GRU-GA loss.

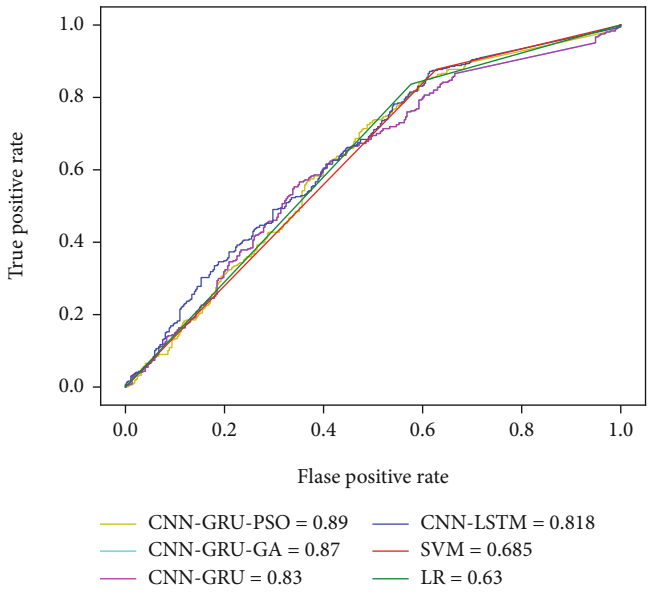

Figure 13: Area under the curve.

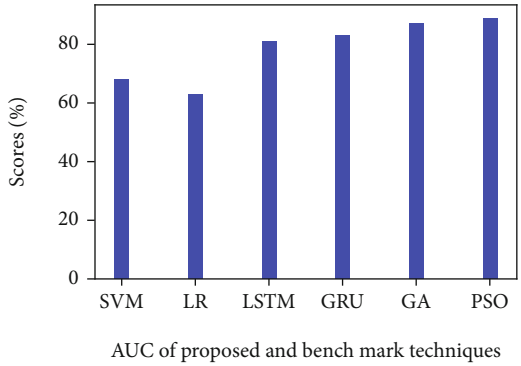

Figure 14: Area under the curve.

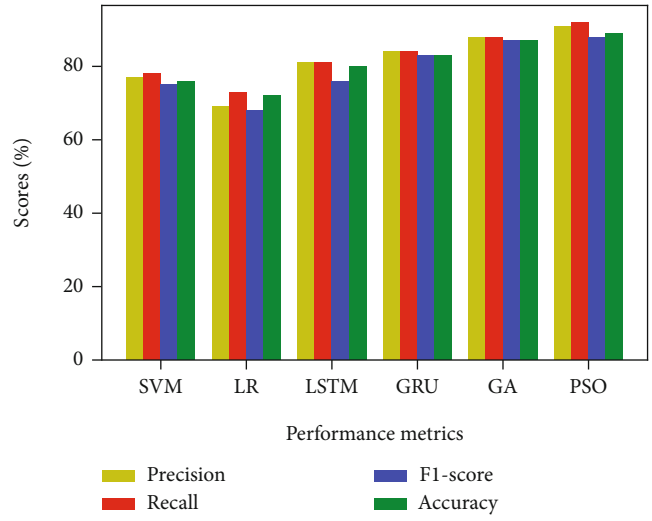

Figure 15: Combined performance evaluation.

Table 13: Results of CNN-PSO-GRU.

Performance evaluation matrix	Results
AUC	89%
*F*1-score	88.54%
Accuracy	89%
Precision	91.23%
Recall	92.72%

which are participated in the experiment are assumed as honest consumers. So, the dataset contains the EC of only normal consumers. For ETD, malicious samples need to be generated as well. For this, six theft attacks are applied on benign samples in order to generate attack samples. These attacks are introduced by Jokar et al. in [1]. After applying theft attacks, the generated theft samples are drastically increased than benign class samples. Now, the dataset

Table 14: Execution time of the proposed and baseline models.

Models	Execution time (seconds)
SVM	220
LR	35
CNN-LSTM	65
CNN-GRU	56
CNN-GRU-PSO	89

becomes imbalance. For data balancing, SMOTE is used to synthesize the minority benign class.

4.4. Results and Discussion. The simulations are performed on PRECON dataset, and the performance is measured through different performance metrics. Figure 16 depicts the loss of the proposed hybrid model. The decline in loss is noted on every epoch. The epoch is the main controlling parameter during the training phase. The optimal epoch value is 25, which is founded by PSO during parameter tuning. It is observed that the proposed model efficiently learns the EC patterns on each iteration by using the strong feature learning and temporal correlation abilities of the CNN and GRU models, respectively. Similarly, Figure 17 shows the proposed model accuracy. The accuracy tells about how accurately data samples are classified. The higher accuracy means higher correct predictions. The accuracy is increasing gradually on test and train data after each epoch. The optimal epoch value found by PSO is 25. Figure 18 illustrates $F1$-score, which is the harmonic mean of precision and recall. It helps the model to accurately identify the energy thieves. The higher $F1$-score is beneficial for power utilities to recover maximum revenue. The AUC score of the proposed and baseline models is presented in Figure 19. The AUC measures the separability between the positive and negative classes. The proposed model obtains 0.95 of AUC score, which is higher than all benchmark models. This implies that the proposed model efficiently distinguishes two classes and reduces the miss classification rate to a minimal level. Furthermore, Table 15 describes the performance results of baseline models and the proposed model on the PRECON dataset. It is seen from the results that the SVM and LR also perform well because the dataset has 48 features after reducing the data granularity to half hour. The SVM and LR capable to deal this data dimensional effectively and learn EC pattern for efficient ETD. The regular LSTM captures temporal correlation from historical EC data and reduces high FPR. The hybrid of CNN-LSTM and CNN-GRU achieves satisfactory performance towards efficient ETD due to using the CNN feature extraction ability and storing temporal correlations abilities of LSTM and GRU models. However, the proposed CNNGRU-PSO outperforms the other models because of utilizing PSO for hyperparameter tuning. The best selection of hyperparameters boosts the performance of the classification model. PSO finds the optimal set of parameters, where the models obtain the highest performance. It is concluded that the proposed model efficiently performs on both case studies. The performance results indicate that our method is an ideal solution for efficient ETD.

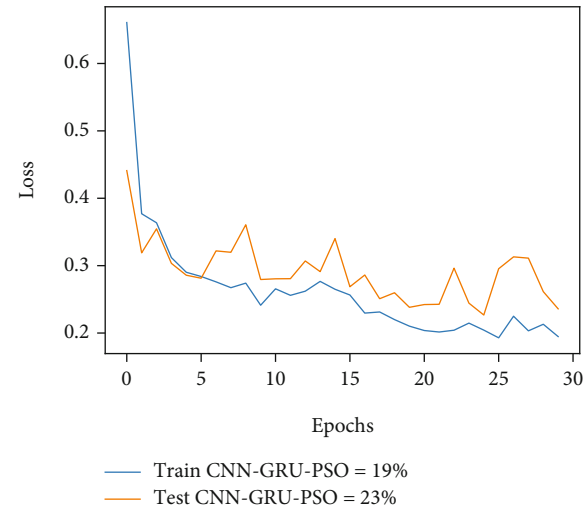

Figure 16: CNN-GRU-PSO loss on PRECON dataset.

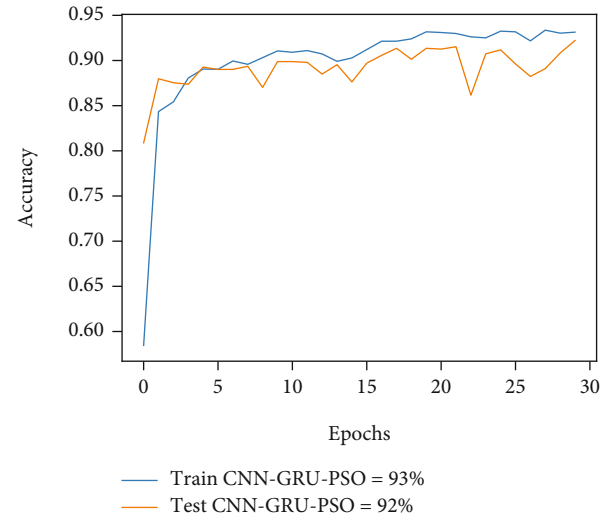

Figure 17: CNN-GRU-PSO accuracy on PRECON dataset.

4.5. Statistical Test Evaluating the Significant of Method Result. Statistical tests are performed to check the significant and robustness of different learning algorithms. These tests determine whether one learning algorithm outperforms another learning algorithm by chance (due to randomness) or in real. The core principal of these tests is null hypothesis. The null hypothesis is constituted according to the problem. For instance, for classification task, the null hypothesis would be the difference between the mean performance of two algorithms is probably real or not. Numerous statistical tests are available to judge the significant of different case studies. However, in this work, our intention is to compare and judge the performance of different classification algorithms. A good statistical test has capable to judge the randomness in different classifiers' results, random variation in classification error, and randomness in the selection of

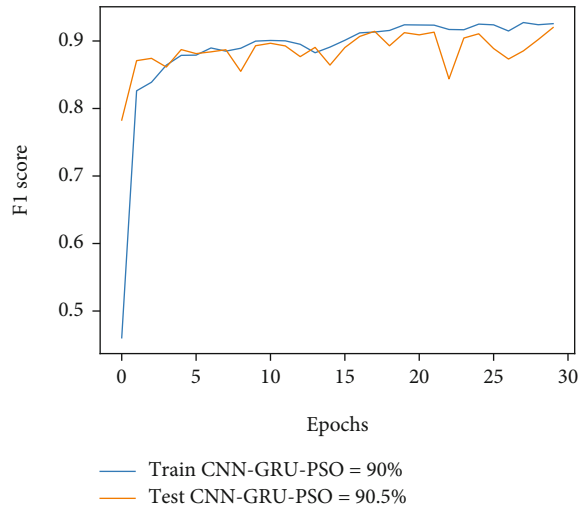

Figure 18: CNN-GRU-PSO *F*1-score on PRECON dataset.

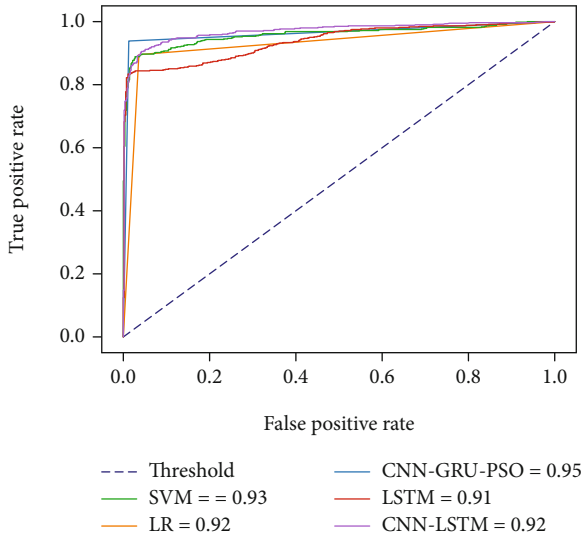

Figure 19: AUC score of CNN-GRU-PSO and baseline models on PRECON dataset.

Table 15: Comparison analysis of proposed model with benchmark schemes on PRECON dataset.

Models	Accuracy	Precision	Recall	*F*1-score	AUC
SVM	0.94	0.91	0.92	0.92	0.93
LR	0.93	0.90	0.92	0.91	0.92
LSTM	0.83	0.92	0.91	0.91	0.91
CNN-LSTM	0.91	0.92	0.93	0.92	0.92
CNN-GRU	0.92	0.93	0.93	0.93	0.93
CNN-GRU-PSO	0.93	0.94	0.95	0.94	0.95

test data. In this regard, three suitable statistical tests are opted that are closely related to the classification task. The detailed description of these test is given as follows.

(1) *5x2cv Paired t Test*. A well-known statistical test for evaluating the performance of classification and regression models. It is introduced by Dietterich in 1998 [41]. In this study, this test is conducted to judge the performance of different classifiers. It consist of twofold cross-validation with five repeats. For each fold, the classifier is trained and the results are recorded. Afterward, the 5x2 paired t test is applied on the final result to accept or reject the null hypothesis. In this case, the null hypothesis defines as the difference between the mean performance of two algorithms is probably real or not. In this test, the p value is calculated against each t value. The p denotes the probability value, which decides that the result of your sample data is occurred by chance or not. The p value ranges from 0 to 1. The smallest p values are suitable. In general, the p value should be less than 0.05 (5%). However, this test is suitable where sample size is small and the mean is known

(2) *5x2cv Combined f Test*. This basic principal of this test is similar to the 5x2cv paired t test except some improvement. This test is introduced by Alpaydin [42]. It is suitable for large sample size and capable to compare more than two populations at the same time. This test is conducted to compare the variance of two population while the 5x2cv paired t test judges the mean of two populations. The working mechanism of this test is similar to the 5x2cv paired t test. The two-fold cross-validation is conducted with five repeats. The results of each fold are recorded and then f test is applied to evaluate the null hypothesis. The null hypothesis is based on the p value. The null hypothesis is accepted if p value is smaller than 0.05 (5%) otherwise rejected

(3) *McNemar's Test*. This test is nonparametric and distribution free statistical test. This test is conducted to check the disagreement of the classifiers on the performance results. In this test, a contingency table is constructed to judge its homogeneity. According to our scenario, this test judges how two or more classifier agree or disagree on the same sample. For this, the contingency table is designed according to the confusion matrix of two classifiers. The structure of contingency shown in Table 16 is given below. The table is finalized by filling the required values, and these values are derived from the confusion matrix. Afterward, the McNemar's test statistic is calculated from the contingency table by using following formula

$$Ms = \left(\frac{(\text{Yes/No} - \text{No/Yes})^2}{(\text{Yes/No}) + (\text{No/Yes})} \right), \tag{21}$$

where Ms denotes McNemar's test statistic. Similar to abovementioned test, a null hypothesis is formulated. The null hypothesis ($H0$) is defined as the classifiers have similar proportion of errors in test set or vice versa. The p value is

Table 16: Contingency table.

	Cf2 correct	Cf2 incorrect
Cf2 correct	Yes/yes	Yes/no
Cf2 incorrect	No/yes	No/no

Table 17: Statistical test on proposed with existing models.

	5x2cv paired t test	
Models	t-statistic	p value
Proposed and SVM	-0.678	0.528
Proposed and LR	0.223	0.489
Proposed and LSTM	0.449	0.672
Proposed and CNNLSTM	0.295	0.780
	5x2cv paired f test	
Proposed and SVM	1.253	0.078
Proposed and LR	1.968	0.049
Proposed and LSTM	2.035	0.215
Proposed and CNNLSTM	2.295	0.0236
	McNemar's test	
Proposed and SVM	1.789	0.0569
Proposed and LR	2.365	0.0422
Proposed and LSTM	4.369	0.1258
Proposed and CNNLSTM	3.269	0.056

calculated to accept or reject the $H0$. The $H0$ is rejected if the value of p is less than 0.005 and vise versa.

Table 17 describes the results of different statistical tests. It is seen that the results of 5x2cv paired f test outperform the other tests because it is suitable for large population size. The value of probability (p) is almost lesser that 0.005 (5%). The smallest value of p indicates that the models' results are that occurred by chance. The t test does not perform well because the available dataset has large in both population and sample size. The McNemar's test also yields better value of p and proves that the models' results are not occurred by chance. All the results are real and do not depend on any noise factor.

5. Conclusion and Future Work

This work presents a HDNN based model in order to detect electricity theft in the smart grid. For this, dataset is taken from SGCC, which provides the real EC data gathered using intelligent antenna-based smart meters installed at the consumers' end. The proposed model works in several steps. In the preprocessing step, the raw data is normalized, and the outliers and missing values are handled. The preprocessing is done by the local average method and min-max normalization technique. Then, the feature engineering step is performed using CNN. Once the most relevant and normalized data is obtained, the classification process is done using PSO-GRU integrated CNN. In this step, the normal and fraudulent consumers are classified. The proposed model is validated in terms of several performance metrics like accuracy, recall, precision, AUC, and $F1$-score. Moreover, comparison of the proposed and existing hybrid models is done. The models include CNN-GRU, CNN-LSTM, and CNN-GRU-GA.

The comparison results show the efficiency, accuracy, robustness, and generalization of the proposed hybrid model for handling imbalanced class issue in terms of ETD. Despite that, our proposed method is an ideal solution towards efficient ETD. However, it has incurred little bit higher computational cost because the proposed model's modules are integrated in a sequential manner (CNN-GRU-PSO). First, CNN takes time while capturing potential features from high-dimensional EC data. Second, GRU processes the CNN's extracted features map for final classification. Meanwhile, PSO tunes the hyperparameters of both CNN and GRU models. This working flow of the proposed model consumed a little bit higher execution time as compared to the existing methods. Moreover, the proposed method has a lack in some complex real-world scenarios by accurately identifying the electricity thieves due to the addition of simulated theft data (in minority class using SMOTE). For the future, more robust techniques will be utilized to efficiently handle the overfitting issue.

Data Availability

The datasets used in this study are openly available in [henryRD- lab/ElectricityTheftDetection] at [23] [Page 3, Section III-A1].

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] P. Jokar, N. Arianpoo, and V. C. M. Leung, ""ETD in AMI using customers" consumption patterns," *IEEE Transactions on Smart Grid*, vol. 7, no. 1, pp. 216–226, 2015.

[2] S. Kazmi, N. Javaid, M. J. Mughal, M. Akbar, S. H. Ahmed, and N. Alrajeh, "Towards optimization of metaheuristic algorithms for IoT enabled smart homes targeting balanced demand and supply of energy," *IEEE Access*, vol. 7, pp. 24267–24281, 2017.

[3] M. S. Saeed, M. W. Mustafa, U. U. Sheikh, T. A. Jumani, and N. H. Mirjat, "Ensemble bagged tree based classification for reducing non-technical losses in multan electric power company of Pakistan," *Electronics*, vol. 8, no. 8, p. 860, 2019.

[4] H. O. Henriques, R. L. S. Correa, M. Z. Fortes, B. S. M. C. Borba, and V. H. Ferreira, "Monitoring technical losses to improve non-technical losses estimation and detection in LV distribution systems," *Measurement*, vol. 161, article 107840, 2020.

[5] Z. Aslam, F. Ahmed, A. Almogren, M. Shafiq, M. Zuair, and N. Javaid, "An attention guided semi-supervised learning mechanism to detect electricity frauds in the distribution systems," *IEEE Access*, vol. 8, pp. 221767–221782, 2020.

[6] L. Raggi, F. Trindade, V. C. da Cunha, and W. Freitas, "Non-technical loss identification by using data analytics and customer smart meters," *IEEE Transactions on Power Delivery*, vol. 35, no. 6, pp. 2700–2710, 2020.

[7] K. M. U. Ghori, R. A. Abbasi, M. Awais, A. Ullah, and L. Szathmary, "Performance analysis of different types of machine learning classifiers for non-technical loss detection," *IEEE Access*, vol. 8, pp. 16033–16048, 2019.

[8] K. M. U. Ghori, M. Imran, A. Nawaz, R. A. Abbasi, A. Ullah, and L. Szathmary, "Performance analysis of machine learning classifiers for non-technical loss detection," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–16, 2020.

[9] A. Aldegheishem, M. Anwar, N. Javaid, N. Alrajeh, M. Shafiq, and H. Ahmed, "Towards sustainable energy efficiency with intelligent electricity theft detection in smart grids emphasising enhanced neural networks," *IEEE Access*, vol. 9, pp. 25036–25061, 2021.

[10] P. F. M. Simões, R. C. Souza, R. F. Calili, and J. F. M. Pessanha, "Analysis and short-term predictions of non-technical loss of electric power based on mixed effects models," *Socio-Economic Planning Sciences*, vol. 71, article 100804, 2020.

[11] Z. Yan and W. He, "Electricity theft detection base on extreme gradient boosting in AMI," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–9, 2021.

[12] Z. Qu, H. Li, Y. Wang, J. Zhang, A. A. Siada, and Y. Yao, "Detection of electricity theft behavior based on improved synthetic minority oversampling technique and random forest classifier," *Energies*, vol. 13, no. 8, article 2039, 2020.

[13] O. Samuel, N. Javaid, A. Khalid et al., "Towards real-time energy management of multi-microgrid using a deep convolution neural network and cooperative game approach," *IEEE Access*, vol. 8, pp. 161377–161395, 2020.

[14] Z. Aslam, N. Javaid, A. Ahmad, A. Ahmed, and S. M. Gulfam, "A combined deep learning and ensemble learning methodology to avoid electricity theft in smart grids," *Energies*, vol. 13, no. 21, article 5599, 2020.

[15] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: a survey," *Heliyon*, vol. 4, no. 11, article e00938, 2018.

[16] K. Wang, C. Xu, Y. Zhang, S. Guo, and A. Y. Zomaya, "Robust big data analytics for electricity price forecasting in the smart grid," *IEEE Transactions on Big Data*, vol. 5, no. 1, pp. 34–45, 2017.

[17] Z. A. Khan, M. Adil, N. Javaid, M. N. Saqib, M. Shafiq, and J.-G. Choi, "Electricity theft detection using supervised learning techniques on smart meter data," *Sustainability*, vol. 12, no. 19, article 8023, 2020.

[18] A. Ullah, N. Javaid, O. Samuel, M. Imran, and M. Shoaib, "CNN and GRU based deep neural network for electricity theft detection to secure smart grid," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, Limassol, Cyprus, 2020.

[19] "State grid corporation of China," https://github.com/henryRDlab/ElectricityTheftDetection.

[20] M. Hasan, R. N. Toma, A. A. Nahid, M. M. M. Islam, and J. M. Kim, "Electricity theft detection in smart grid systems: a CNN-LSTM based approach," *Energies*, vol. 12, no. 17, p. 3310, 2019.

[21] S. K. Singh, R. Bose, and A. Joshi, "Energy theft detection for AMI using principal component analysis based reconstructed data," *IET Cyber-Physical Systems: Theory & Applications*, vol. 4, no. 2, pp. 179–185, 2019.

[22] J. V. Spiric, S. S. Stankovic, and M. B. Docic, "Identification of suspicious electricity customers," *International Journal of Electrical Power & Energy Systems*, vol. 95, pp. 635–643, 2018.

[23] K. V. Blazakis, T. N. Kapetanakis, and G. S. Stavrakakis, "Effective electricity theft detection in power distribution grids using an adaptive neuro fuzzy inference system," *Energies*, vol. 13, no. 12, p. 3110, 2020.

[24] A. Grewal, M. Kaur, and J. H. Park, "A unified framework for behaviour monitoring and abnormality detection for smart home," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 1734615, 16 pages, 2019.

[25] X. Lu, Z. Yu, Z. Wang, Y. Yi, L. Feng, and F. Wang, "Knowledge embedded semi-supervised deep learning for detecting non-technical losses in the smart grid," *Energies*, vol. 12, no. 18, p. 3452, 2019.

[26] N. Ding, H. X. Ma, H. Gao, Y. H. Ma, and G. Z. Tan, "Real-time anomaly detection based on long short-term memory and Gaussian mixture model," *Computers & Electrical Engineering*, vol. 79, article 106458, 2019.

[27] N. F. Avila, G. Figueroa, and C.-C. Chu, "NTL detection in electric distribution systems using the maximal overlap discrete wavelet-packet transform and random undersampling boosting," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 7171–7180, 2018.

[28] C. C. O. Ramos, D. Rodrigues, A. N. de Souza, and J. P. Papa, "On the study of commercial losses in Brazil: a binary black hole algorithm for theft characterization," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 676–683, 2016.

[29] K. Zheng, Q. Chen, Y. Wang, C. Kang, and Q. Xia, "A novel combined data-driven approach for electricity theft detection," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, article 18091819, 2018.

[30] C. Li, X. Zheng, Z. Yang, and L. Kuang, "Predicting short-term electricity demand by combining the advantages of ARMA and XGBoost in fog computing environment," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 5018053, 18 pages, 2018.

[31] G. Fenza, M. Gallo, and V. Loia, "Drift-aware methodology for anomaly detection in smart grid," *IEEE Access*, vol. 7, pp. 9645–9657, 2019.

[32] K. Wang, C. Xu, and S. Guo, "Big data analytics for price forecasting in smart grids," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, DC, USA, 2016.

[33] H. Liu, G. Tianlong, Y. Liu, J. Song, and Z. Zeng, "Fault-tolerant privacy-preserving data aggregation for smart grid," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8810393, 10 pages, 2020.

[34] A. Jamil, T. A. Alghamdi, Z. A. Khan et al., "An innovative home energy management model with coordination among appliances using game theory," *Sustainability*, vol. 11, no. 22, p. 6287, 2019.

[35] G. Micheli, E. Soda, M. T. Vespucci, M. Gobbi, and A. Bertani, "Big data analytics: an aid to detection of non-technical losses in power utilities," *Computational Management Science*, vol. 16, no. 1-2, pp. 329–343, 2019.

[36] M. Adil, N. Javaid, U. Qasim, I. Ullah, M. Shafiq, and J.-G. Choi, "LSTM and bat-based RUSBoost approach for electricity theft detection," *Applied Sciences*, vol. 10, no. 12, p. 4378, 2020.

[37] H. Gul, N. Javaid, I. Ullah, A. M. Qamar, M. K. Afzal, and G. P. Joshi, "Detection of non-technical losses using SOSTLink and bidirectional gated recurrent unit to secure smart meters," *Applied Sciences*, vol. 10, no. 9, p. 3151, 2020.

[38] M. Ismail, M. F. Shaaban, M. Naidu, and E. Serpedin, "Deep learning detection of electricity theft cyber-attacks in renewable distributed generation," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3428–3437, 2020.

[39] A. Thengade and R. Dondal, "Genetic algorithm-survey paper," in *MPGI National Multi Conference*, Citeseer, 2012.

[40] J. Genlin, "Survey on genetic algorithm," *Computer Applications and Software*, vol. 2, no. 1, pp. 69–73, 2004.

[41] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[42] E. Alpaydin, "Combined 5×2 cv F test for comparing supervised classification learning algorithms," *Neural Computation*, vol. 11, no. 8, article 18851892, 1999.