IEEE Access on Machine Learning,Volume:10,Issue Date:11.April.2022

Electricity Theft Detection in Smart Grids Based on Deep Neural Network

LELOKO J. LEPOLESA, SHAMIN ACHARI, AND LING CHENG, (Senior Member, IEEE)
School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, Johannesburg 2000, South Africa
Corresponding author: Ling Cheng (ling.cheng@wits.ac.za)

This work was supported in part by the South African National Research Foundation under Grant 129311 and Grant 132651.

ABSTRACT Electricity theft is a global problem that negatively affects both utility companies and electricity users. It destabilizes the economic development of utility companies, causes electric hazards and impacts the high cost of energy for users. The development of smart grids plays an important role in electricity theft detection since they generate massive data that includes customer consumption data which, through machine learning and deep learning techniques, can be utilized to detect electricity theft. This paper introduces the theft detection method which uses comprehensive features in time and frequency domains in a deep neural network-based classification approach. We address dataset weaknesses such as missing data and class imbalance problems through data interpolation and synthetic data generation processes. We analyze and compare the contribution of features from both time and frequency domains, run experiments in combined and reduced feature space using principal component analysis and finally incorporate minimum redundancy maximum relevance scheme for validating the most important features. We improve the electricity theft detection performance by optimizing hyperparameters using a Bayesian optimizer and we employ an adaptive moment estimation optimizer to carry out experiments using different values of key parameters to determine the optimal settings that achieve the best accuracy. Lastly, we show the competitiveness of our method in comparison with other methods evaluated on the same dataset. On validation, we obtained 97% area under the curve (AUC), which is 1% higher than the best AUC in existing works, and 91.8% accuracy, which is the second-best on the benchmark.

INDEX TERMS Deep neural network, electricity theft, machine learning, minimum redundancy maximum relevance, principal component analysis, smart grids.

I. INTRODUCTION

ELECTRICITY theft is a problem that affects utility companies worldwide. More than $96 billion is lost by utility companies worldwide due to Non-Technical Losses (NTLs) every year, of which electricity theft is the major contributor [1]. In sub-Saharan Africa, 50% of generated energy is stolen, as reported by World Bank [2].

The ultimate goal of electricity thieves is to consume energy without being billed by utility companies [3], or pay the bills amounting to less than the consumed amount [4]. As a result, utility companies suffer a huge revenue loss due to electricity theft. [5] reports that in 2015, India lost $16.2 billion, Brazil lost $10.5 billion and Russia lost $5.1 billion. It is estimated that approximately $1.31 billion (R20 billion) revenue loss incurred by South Africa (through Eskom) per year is due to electricity theft [2].

Apart from revenue loss, electricity theft has a direct negative impact on the stability and reliability of power grids [3]. It can lead to surging electricity, electrical systems overload and public safety threats such as electric shocks [4]. It also has a direct impact on energy tariff increases, which affect all customers [3]. Implementation of smart grids comes with many opportunities to solve the electricity theft problem [4]. Smart grids are usually composed of traditional power grids, smart meters and sensors, computing facilities to monitor and control grids, etc., all connected through the communication network [6]. Smart meters and sensors collect data such as electricity usage, grid status, electricity price, etc. [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Mahdi Pourakbari Kasmaei.

Many Utilities sought to curb electricity theft in traditional grids by examining meters' installation and configurations, checking whether the power line is bypassed, etc. [4]. These methods are expensive, inefficient and cannot detect cyber attacks [4], [7]. Recently, researchers have worked towards detecting electricity theft by utilizing machine learning classification techniques using readily available smart meters data. These theft detection methods have proved to be of relatively lower costs [8]. However, existing classification techniques consider time-domain features and do not regard frequency-domain features, thereby limiting their performance.

Regardless of the fact that there is active ongoing research on electricity theft detection, electricity theft is still a problem. The major cause of delay in solving this problem may be that smart grids deployment is realized in developed nations while developing nations are lagging behind [9]. The challenges of deploying smart grids include the lack of communication infrastructure and users' privacy concerns over data reported by the smart meters [10]. However, [10] reports that smart meters are being considered by many developed and developing countries with aims that include solving NTLs. [11] predicted smart grids global market to triple in size between 2017 and 2023, with the following key regions leading smart grids deployment: North America, Europe and Asia.

In this paper, we present an effective electricity theft detection method based on carefully extracted and selected features in Deep Neural Network (DNN)-based classification approach. We show that employing frequency-domain features as opposed to using time-domain features alone enhances classification performance. We use a realistic electricity consumption dataset released by State Grid Corporation of China (SGCC) accessible at [12]. The dataset consists of electricity consumption data taken from January 2014 to October 2016. The main contributions are as follows:

- Based on the literature, we propose a novel DNN classification-based electricity theft detection method using comprehensive time-domain features. We further propose using frequency-domain features to enhance performance.
- We employ Principal Component Analysis (PCA) to perform classification with reduced feature space and compare the results with classification done with all input features to interpret the results and simplify the future training process.
- We further use the Minimum Redundancy Maximum Relevance (mRMR) scheme to identify the most significant features and validate the importance of frequency-domain features over time-domain features for detecting electricity theft.
- We optimize the hyperparameters of the model for overall improved performance using a Bayesian optimizer. We further employ an adaptive moment estimation (Adam) optimizer to determine the best ranges of values of the other key parameters that can be used to achieve good results with optimal model training speed.
- Lastly, we show 1% improvement in AUC and competitive accuracy of our model in comparison to other data-driven electricity theft detection methods in the literature evaluated on the same dataset.

The remainder of this paper is organized as follows. Section II covers the related work done in literature to tackle the electricity theft problem. In Section III, we briefly introduce techniques used in this paper. Section IV covers step by step method taken in this work; which includes dataset analysis and work done to improve its quality and customers' load profile analysis which lead to features extraction and classification. In Section V, we show and discuss the results. We finally conclude the paper in Section VI.

II. RELATED WORK

Research on electricity theft detection in smart grids has attracted many researchers to devise methods that mitigate against electricity theft. Methods used in the literature can be broadly categorized into the following three categories: hardware-based, combined hardware and data-based detection methods and data-driven methods.

Hardware-based methods [13]–[19] generally require hardware devices such as specialized microcontrollers, sensors and circuits to be installed on power distribution lines. These methods are generally designed to detect electricity theft done by physically tampering with distribution components such as distribution lines and electricity meters. They can not detect cyber attacks. Electricity cyber attack is a form of electricity theft whereby energy consumption data is modified by hacking the electricity meters [7].

For instance, in [13], an electricity meter was re-designed. It used components that include: a Global System for Mobile Communications (GSM) module, a microcontroller, and an Electrically Erasable Programmable Read-Only Memory (EEPROM). A simulation was done and the meter was able to send a Short Message Service (SMS) whenever an illegal load was connected by bypassing the meter. Limited to detecting electricity theft done by physically tampering with distribution components such as distribution lines and electricity meters. Authors in [16] used the GSM module, ARM-cortex M3 processor and other hardware components to solve the electricity theft problem done in the following four ways: bypassing the phase line, bypassing the meter, disconnecting the neutral line, and tampering with the meter to make unauthorized modifications. A prototype was built to test all four possibilities. The GSM module was able to notify with SMS for each theft case.

Authors in [17] designed ADE7953 chip-based smart meter which is sensitive to current and voltage tempering, and mechanical tempering. ADE7953 was used to detect overvoltage, dropping voltage, overcurrent, the absence of load and other irregularities in voltage and current. It sent an interrupt signal to the Microcontroller Unit (MCU) which

reported tampering status. Mechanical tampering was overcome by connecting a tampering switch to MCU's IO ports so that it can send alarm signals to MCU once tampered with. The design was tested with tampering cases such as connecting the neutral and the phase lines, connecting the meter input and output in reverse mode, and bypassing the phase line to load. The probability of detection failure was 2.13%.

Authors in [15] used a step down transformer, voltage divider circuit, microchip and other hardware components to design a circuitry to detect electricity theft by comparing forward current on the main phase line with reverse current on the neutral line. The circuitry was installed before the meter.The design was tested on Proteus software and on actual hardware. When the meter was bypassed, the problem was detected and an alarm sounded. In [14], a circuit to detect electricity theft done by bypassing the meter was designed. The transformers, rectifiers, microcontroller, GSM module and other hardware components were used. The GSM controller notified the operator with SMS when the meter was bypassed.

Authors in [18] proposed putting the Radio Frequency Identification (RFID) tags on ammeters and capturing unique data about each ammeter. Ammeters were to be tracked and managed real-time. Electricity theft was to be inspected onsite. Damaged, removed or a tag with a different information from the original one means high possibility that an electricity theft happened. Evaluation based on analysis on cost of deployment. With a case study made on utility company in China, Return on Investment (ROI) was found to be >1. In [19], An Arduino-based real-time electricity theft detector was designed. The following hardware was used: Arduino Uno, GSM module, current sensors and LCD. The Arduino Uno obtained measurements from current sensors which were located one on the secondary side of the transformer and the other on the electric service cap. If the difference between current sensors' measurements exceeded a set threshold, the message would be sent to the operator via a GSM module. The simulation was done using Proteus 8 software and the prototype was built on hardware, which was able to report theft cases when tested.

Apart from their inability to detect cyber attacks, these methods are also expensive due to their need for special hardware deployment and maintenance. Combined hardware and data-based electricity theft detection methods [20]–[22] employ the use of hardware, machine learning and/or deep learning techniques to tackle the electricity theft problem. Due to hardware requirements, these methods also pose the challenge of being expensive to deploy and maintain.

In [20], a method to measure the total consumption of a neighbourhood and compare the results with the usage reported by the smart meters in that neighbourhood was proposed. A significant difference between smart meters' and transformers' measurements would mean the presence of unfaithful customers in the neighbourhood. To locate the unfaithful customers in the neighbourhood, the authors proposed using a Support Vector Machine (SVM) classifier. The classifier was tested on a dataset of 5000 (all faithful) customers. A maximum detection rate of 94% and a minimum false positive rate of 11% were achieved.

Authors in [22] developed a predictive model to calculate TLs. To get NTL, TLs would be subtracted from total distribution network losses. Based on an assumption that distribution transformers and smart meters share data to the utility after every 30 minutes, a smart meter simulator was used to generate data for 30 users in 30 minutes intervals for 6 days. On the simulator, unfaithful users stole electricity by bypassing the meter. Stolen electricity was varied between 1% and 10% of the total consumption. For stolen electricity value over 4%, the detection rate was 100%, which diminished as stolen electricity percentage was decreased.

In [21], a method which would use an observer meter that would be installed on a pole away from households and record the total amount of electricity supplied to n households where it is suspected that one or more meters have been tampered with was proposed. The observer meter would have camera surveillance to protect it from being tampered with. A mathematical algorithm that utilizes data from an observer meter and smart meters to detect a smart meter tempered with was developed. A mathematical algorithm was tested with a real-world consumption dataset by increasing the consumption of some meters which were picked randomly. The algorithm was able to detect the meters with altered consumption.

Due to high-cost demand in the above categories, many researchers work on data-driven methods to overcome the electricity theft problem. For instance, the authors in [3] designed an electricity theft detection system by employing three algorithms in the pipeline: Synthetic Minority Over-sampling Technique (SMOTE), Kernel function and Principal Component Analysis (KPCA) and SVM. They used SMOTE to generate synthetic data for balancing an unbalanced dataset, KPCA to extract features and SVM for classification. They obtained maximum overall classifier quality characterized by Area Under the Curve (AUC) of 89% on validation.

Authors in [4] used wide and deep Convolutional Neural Networks (CNN) model to detect electricity theft. Based on that normal electricity consumption is periodical while stolen electricity consumption data is not periodical, wide was to learn multiple co-occurrences of features for 1-D series data, while deep CNN was used to capture periodicity with data aligned in a 2-D manner by weeks. They varied training and validation data ratios, to obtain maximum AUC value of 79%. By utilizing the same dataset used in [3] and [4], the method we present in this paper achieves AUC results beyond 90% on both validation and testing.

In [23], PCA was used to transform original high-dimensional consumption data by extracting Principal Components (PCs) which retained the desired variance. An anomaly score parameter that was defined between set minimum and maximum thresholds was introduced. For

each test sample, the anomaly score parameter was calculated. If the result was not between the set thresholds, the sample would then be treated as malicious. The true positive rate (TPR) was used to evaluate the method, which hit the best-recorded value of 90.9%. Authors in [24] used One-Class SVM (O-SVM), Cost-Sensitive SVM (CS-SVM), Optimum Path Forest (OPF) and C4.5 tree. From customer consumption data, different features were selected, and the performance of each classifier was analyzed independently on a different set of features, followed by combining all classifiers for the best results. Best results were achieved when all classifiers were combined, with 86.2% accuracy.

Authors in [25] employed a combination of CNN and Long Short-Term Memory (LSTM) recurrent neural network deep learning techniques. Seven hidden layers were used, of which four of them were used by CNN and three were utilized by LSTM. This method relied on CNN's automatic feature extraction ability on a given dataset. Features were extracted from 1-D time-series data. On model validation, the maximum accuracy achieved was 89%. The authors in [26] used a combination of Local Outlier Factor (LOF) and k-means clustering to detect electricity theft. They used k-means clustering to analyze the load profiles of customers, and LOF to calculate the anomaly degrees of customers whose load profiles were from their cluster centres. On the evaluation of the method, they attained an AUC of 81.5%. Our model achieves a maximum value of 91.8% accuracy and 97% on validation.

In [27], two electricity theft models were developed. The first model is based on Light Gradient Boosting (LGB) classifier. A combination of SMOTE and Edited Nearest Neighbour (ENN) was used to balance the dataset. Feature extraction was done using AlexNet, followed by classification with LGB. This proposed model was named as SMOTEENN-AlexNet-LGB (SALM). The second model is based on the Adaptive Boosting classifier. Conditional Wasserstein Generative Adversarial Network with gradient penalty (CWGAN-GP) was used to generate synthetic data that resembled the minority class data to balance data of the unbalanced classes. Feature extraction was performed using GoogleNet, then classification by AdaBoost followed. The proposed model was named as GAN-NETBoost. The models were evaluated with SGCC data used in this work. SALM and GAN-NetBoost attained an accuracy of 90% and 95%, and AUC of 90.6% and 96% respectively on validation.

Although these models were able to achieve impressive results, their consideration of time-domain features alone limited their performance. Our solution shows that adding frequency-domain features on time-domain features improves classification performance.

III. PRELIMINARIES

In this section, we give a summary of the main techniques used, which are: Deep Neural Networks (DNNs), Principal Component Analysis (PCA), and Minimum Redundancy Maximum Relevance (mRMR).

A. DEEP NEURAL NETWORKS

Artificial Neural Networks (ANNs) are a class of machine learning techniques that have been built to imitate biological human brain mechanisms [28], [29]. They are typically used for extracting patterns or detecting trends that are difficult to be detected by other machine learning techniques [30].

They consist of multiple layers of nodes/neurons which are connected to subsequent layers [29]. A neuron is the basic element of a neural network, which originates from the McCulloch-Pitts neuron, a simplified model of a human brain's neuron [31]. Figure 1 shows a model diagram of a neuron that comprises a layer following the input to the ANN.

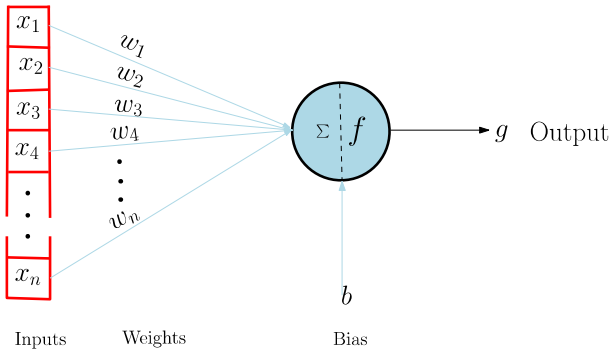


FIGURE 1. First hidden layer neuron model.

It consists of an activation function f, which takes a weighted sum of the real number input signal and gives real number output y given by Equation (1).

$$g = f(\sum (w_i x_i) + b), \tag{1}$$

where $x_i \in \mathbf{x}$, $w_i \in \mathbf{w}$, $\mathbf{x}$ is input vector, $\mathbf{w}$ is weights vector and b is the bias [31]. Neural network nodes mimic the brain's neurons, while connection weights mimic connections between neurons, which are unique for each connection [28], [29]. A neural network stores information in the form of weights and bias.

The Deep Neural Networks (DNNs) concept originates from research on ANNs [32]. DNNs are characterized by two or more hidden layers [28]. They are able to learn more complex and abstract features than shallow ANNs [33]. Oftentimes in classification problems, the output layer is made up in such a way that one neuron represents a certain class [29]. All neural network layers are used to filter and learn the complicated features, except for an output layer which classifies based on learnt features [29], [34]. Before DNNs development, most machine learning techniques explored architectures of shallow structures which commonly contain a single layer of non-linear transformation [32]. Examples of these architectures include SVMs, logistic regression and ANNs with one hidden layer.

DNNs have different architectures, which are used to solve different problems. Examples of DNN architectures include feed-forward DNN, convolutional DNN and recurrent DNN. In this research work, a fully connected feed-forward

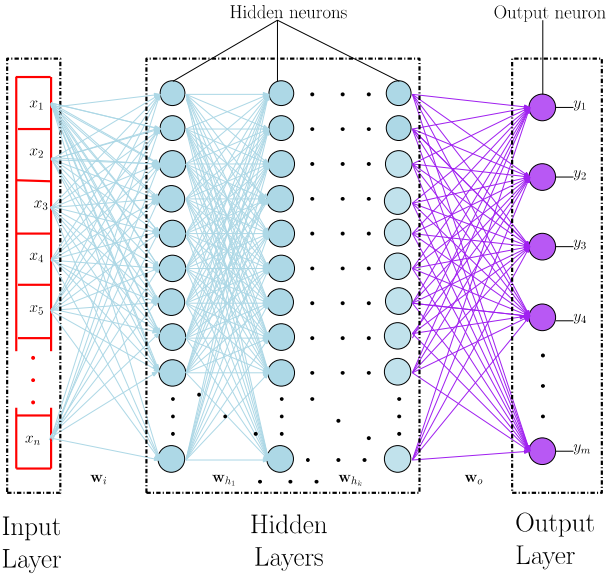


FIGURE 2. Fully connected feed-forward DNN general architecture.

DNN was used. the typical structure of a fully connected feed-forward DNN is shown in Figure 2.

The DNN given in Figure has the following major parts:

- Input layer ($\mathbf{x}$)
  A layer that comprises input data features or representation.
- Input weights ($\mathbf{w}_i$)
  Weights of the connections between the input layer and the first hidden layer of a DNN.
- Hidden layers
  The layers of neurons between the input and output layers. They are used to analyse the relationship between the input and output signals [30].
- Hidden neurons weights ($[\mathbf{w}_{h_1}, \cdots, \mathbf{w}_{h_k}]$)
  Weights of the connections between the hidden layers.
- Output weights ($\mathbf{w}_o$)
  Weights between the last hidden layer and the output layer.
- Output layer ($\mathbf{y}$)
  The last layer of a DNN. It gives the output of the network from network inputs.

In a feed-forward architecture, computation is a sequence of operations on the output of a previous layer. The final operations generate the output. For a given input, the output stays the same, it does not depend on the previous network input [33].

1) HISTORY OF DNNs DEVELOPMENT

[33] reports that ANNs were first proposed in the 1940s, and research on DNNs emerged in the 1960s. In 1989, the LeNet network, which used many digital neurons, was built for recognizing hand-written digits. Major breakthroughs were seen in the years beyond 2010, with examples such as Microsoft's speech recognition system, AlexNet image recognition system, and DNN accelerator research such as Neuflow and DianNao brought into play.

The following reasons are reported by [30], [32], [33] as major contributors to DNNs' improved development:

- Advancements in semiconductor devices and computer architecture, leading to parallel computing and lower costs of computer hardware.
- Huge amount of data obtained by cloud providers and other businesses, making large datasets that train DNNs effectively.
- Advances in machine learning and signal/information processing research which leads to the evolution of techniques to improve accuracy and broaden the domain of DNNs application.

With present technology permission, DNNs can have a count of layers that is beyond a thousand [33].

2) DNN TRAINING

A large dataset and high computational abilities are the major requirements in training the DNN since weight updates require multiple iterations [33]. DNN training process is concerned with adjusting the weights between the neurons [30]. Through the training process, the DNN learns information from the data. Learning can be in the following major four ways: supervised, semi-supervised, unsupervised or reinforcement [33]–[36].

In this work, supervised learning was used. The typical procedure for supervised learning in DNNs as given by [28], [34] is as follows:

1) The weights $\mathbf{W} = [\mathbf{w}_i, \mathbf{w}_{h_1}, \cdots, \mathbf{w}_{h_k}, \mathbf{w}_o]$ are initialized with adequate values.
2) Input signal $\mathbf{x}$ is fed to the network's input layer.
3) Output error is calculated, and then weights adjusted with an aim to reduce an error.
4) Steps 2 and 3 are repeated for all training data.

3) BACKPROPAGATION

A loss function of a multi-layered ANN is composed of weights from successive layers between input and output layers [36]. Backpropagation uses chain rule to obtain the gradient of the loss function in terms of summation of local gradient products over different nodes connections between input and output layers [28], [29], [36]. Backpropagation algorithms typically use gradient-based optimization algorithms to update the neural network parameters on each layer [37].

4) ACTIVATION FUNCTIONS

An activation function takes an input signal, by simulating a response of a biological neuron, transforms the signal into an output signal which may be an input to another neuron [38], [39]. There are many activation functions, which can be generally divided into two kinds: the linear and non-linear activation functions. The type of activation function used in a DNN plays a major role in the prediction accuracy of the

model [39]. The selection of an activation function depends on the reasons such as computational power, analytic flexibility and whether the desired output should be continuous or discrete [30]. Let $z = \sum(w_i x_i) + b$. Then Equation (1) can be re-written as shown in Equation (2).

$$g = f(z) \tag{2}$$

5) LINEAR ACTIVATION FUNCTIONS

Linear activation functions usually have an activation that is directly proportional to the input. They can be expressed in the form of Equation (3).

$$f(z) = Cz \tag{3}$$

where C is a constant. The output of the linear activation function is in the range $(-\infty, \infty)$ and its derivative is $f'(z) = C$. Since the gradient is not related to the input, an error can not be minimized by the use of a gradient [40]. This activation function is normally used in regression problems [41].

6) NON-LINEAR ACTIVATION FUNCTIONS

Non-linear activation functions are widely used in DNNs because of their ability to adapt to data variations and differentiate outputs [40]. Among the many developed non-linear activation functions, the most popular are described as follows [38]–[41].

- *Sigmoid activation function*
  Sigmoid activation function is given by Equation (4)

$$f(z) = \frac{1}{1 + \mathrm{e}^{-z}} \tag{4}$$

  where input $z \in (-\infty, \infty)$ and an activation $f \in (0, 1)$. Sigmoid is continuous and differentiable everywhere. Its derivative is given by Equation (5).

$$f'(z) = \frac{e^{-z}}{(1 + \mathrm{e}^{-z})^2} \tag{5}$$

  Due to less computation in finding its derivative, this activation function is widely used in shallow neural networks. It is rarely used in DNNs' hidden layers because of its soft saturation property which makes DNNs delay converging during training.
- *Hyperbolic tangent activation function*
  Like the sigmoid, hyperbolic tangent is continuous and differentiable everywhere. It is given by Equation (6). Its derivative is given by Equation (7).

$$f(z) = \tanh(z) = \frac{\mathrm{e}^{z} - \mathrm{e}^{-z}}{\mathrm{e}^{z} + \mathrm{e}^{-z}}. \tag{6}$$

$$f'(z) = \frac{4\mathrm{e}^{-2z}}{(1 + \mathrm{e}^{-2z})^2}. \tag{7}$$

  The input $z \in (-\infty, \infty)$ and an activation $f \in (-1, 1)$. Using a hyperbolic tangent for activation makes the neural networks converge faster than when using a sigmoid, therefore a hyperbolic tangent is more preferred than a sigmoid.
- *Rectified linear unit activation function*
  Rectified linear unit (ReLU) activation function is given by Equation (8) and is derivative by Equation (9).

$$f(z) = \max(0, z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{8}$$

$$f'(z) = \max(0, z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{9}$$

  Compared to sigmoid and hyperbolic tangent activation functions, ReLU is the simplest and most commonly used in DNNs because of its good property of being close to linear, hence better convergence. It is more efficient since it activates less number of neurons at the same time. For $z > 0$, its gradient is constant thereby avoiding the vanishing gradient problem. Its gradient is cheaper to compute as there are no calculations that involve exponents.
- *Softmax activation function*
  Softmax activation function is given by Equation (10).

$$f(z)_i = \frac{\mathrm{e}^{z_i}}{\sum_{j=1}^{K} \mathrm{e}^{z_j}} \tag{10}$$

  where K is the number of classes.
  Softmax is typically used in the output layer of a DNN for classification purposes. The output of a softmax is a probability of a particular class j, therefore if the softmax activation function is used in the output layer, all of the output layer activations sum to 1.

B. PRINCIPAL COMPONENT ANALYSIS

PCA [42] is used to extract important information from a data table of inter-correlated features/variables that represent observations. This extracted information is represented as a new set of orthogonal variables known as Principal Components (PCs). In this work, PCA uses a Singular Value Decomposition (SVD) algorithm [43] which works in the following manner: for input feature matrix $\mathbf{X}$, SVD decomposes it into three matrices, i.e., $\mathbf{X} = \mathbf{PQR}^{\intercal}$, such that:

- $\mathbf{P}$ is the normalized eigen vectors of the matrix $\mathbf{XX}^{\intercal}$,
- $\mathbf{Q} = \mathbf{E}^{\frac{1}{2}}$ where $\mathbf{E}$ is a diagonal matrix of eigen values of matrix $\mathbf{XX}^{\intercal}$, and
- $\mathbf{R}$ is the normalized eigen vectors of matrix $\mathbf{X}^{\intercal}\mathbf{X}$.

When PCA is applied to a matrix $\mathbf{X}$ of size $m \times n$, n PCs $\{c\}_{i=1}^{n}$ are obtained, which are ordered in descending order with respect to their variances [23]. A PC at position p is given by

$$c_p = \arg\max_{||c||=1} ||(\mathbf{X} - \sum_{i=1}^{p-1} \mathbf{X} c_i c_i^{\intercal})c||, \tag{11}$$

and its variance is obtained by evaluating $||\mathbf{X}c_p||^2$.

The main goals achieved with PCA are as follows:

- Extraction of most important information from data/feature table, thereby compressing and simplifying dataset description, and
- Analysis of observations and variables' structure.

For dimensionality reduction purposes, the first $r \leq n$ PCs that retain acceptable variance can accurately represent feature matrix $\mathbf{X}$ in a reduced r-dimensional subspace.

C. MINIMUM REDUNDANCY MAXIMUM RELEVANCE

An mRMR [44], [45] is a feature selection scheme that selects features that have a high correlation with the response variable and low correlation with themselves. It ranks features based on mutual information of a feature and a response variable, and pairwise mutual information of features. Mutual information between variables $\mathbf{A}$ and $\mathbf{B}$ is given by

$$\mathbf{MI}(\mathbf{A}, \mathbf{B}) = \sum_{a \in \mathbf{A}} \sum_{b \in \mathbf{B}} p(a, b) \log \frac{p(a, b)}{p(a)p(b)}. \tag{12}$$

For all features $\{\mathbf{X}_i\}$, maximum relevance R_l is implemented using mean value of their mutual information with an output class $\mathbf{O}$.i.e.,

$$R_l = \frac{1}{|X|} \sum \mathbf{MI}(\mathbf{O}, \mathbf{X}_i). \tag{13}$$

Minimum redundancy R_d helps to select features that are mutually maximally dissimilar. It is given by:

$$R_d = \frac{1}{|X|^2} \sum \mathbf{MI}(\mathbf{X}_i, \mathbf{X}_j). \tag{14}$$

where $\mathbf{X}_i, \mathbf{X}_j \in \mathbf{X}$. mRMR feature selection goal is achieved by optimizing relevance and redundancy in the following manner: $\max(R_l - R_d)$.

IV. DNN-BASED ELECTRICITY THEFT DETECTION METHOD

The electricity theft detection method outlined in this section consists of the following three steps: Data Analysis and Preprocessing, Feature Extraction, and Classification. Figure 3 shows the workflow diagram.

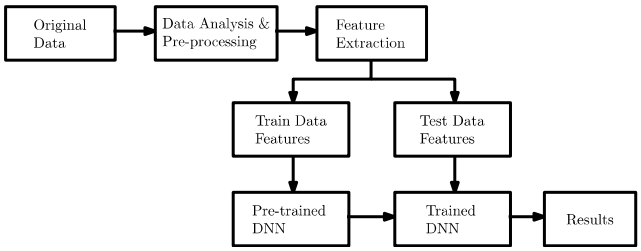


FIGURE 3. Electricity theft detection workflow diagram.

A. DATA ANALYSIS AND PRE-PROCESSING

In this sub-section, we present the dataset used and its quality improvement by identifying and removing observations that had no consumption data. In this work, an observation refers to a single instance/record in the dataset, for the duration of measured consumption. i.e., given a dataset $\mathbf{A}$ of size N, $\mathbf{a}_i \in \mathbf{A}$; where $\mathbf{a}_i$ is the i^{th} observation of $\mathbf{A}$ and $1 \leq i \leq N$.

We show customers' load profiles analysis. We further present data interpolation and synthetic data generation details that have been undertaken.

1) DATASET ANALYSIS AND PREPARATION

As stated in Section I, we used a realistic electricity consumption dataset released by SGCC, which is accessible at [12]. The dataset consists of daily electricity consumption data taken from January 2014 to October 2016, summarized in Table 1. The sampling rate of the data is uniform for every customer, it is one measurement per day; which corresponds to the total power consumption for that day. The used dataset consists of 42372 observations, of which 3615 observations are electricity consumption data of unfaithful customers and the remaining observations are electricity consumption data of faithful customers.

TABLE 1. Dataset summary table.

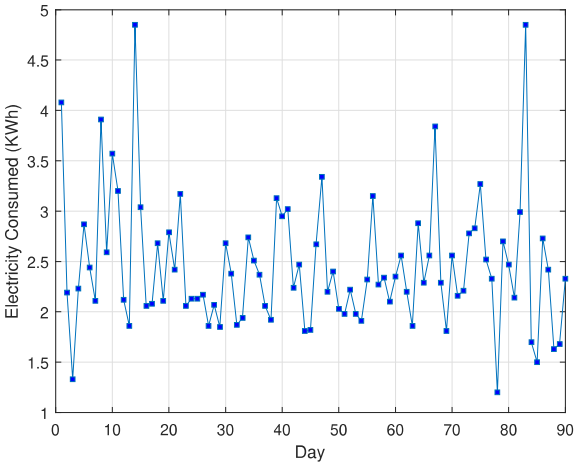
Number of observation days: 1,034

Customer Class	**Number Of Observations**		
	From Original Dataset	*After Removing Empty Observations*	*After Synthetic Data Generation*
Faithful	38,757	36,679	36,679
Unfaithful	3,615	3,579	36,679
Total	**42,372**	**40,258**	**73,358**

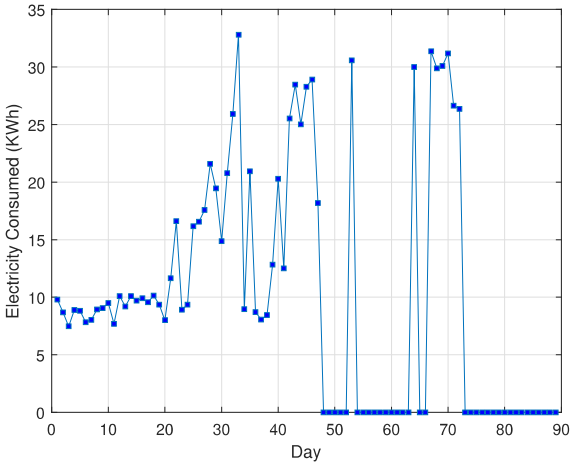
As with many datasets used in the literature, data comes with many errors caused by factors such as smart meters failures, data storage problems, data transmission issues and unscheduled systems maintenance [4]. Dataset used in this work is no exception. It consists of traces of non-numerical or null values.Using data analysis methods, we found approximately 5.45% of observations in this dataset to either have only null values, or zeros, or a combination of both, for the whole duration of 1034 days. These observations were regarded as *empty observations*. i.e., An observation $\mathbf{a}$ is regarded as an empty observation if $a_i = 0$ or $a_i \notin \mathbb{R}$ for all $a_i \in \mathbf{a}$. These observations do not have any differentiating characteristics between the classes since they do not have any consumed electricity record greater than 0 kWh. To improve the dataset quality, these observations were removed. They could not be identified with any class as they were labeled with either of the classes, therefore they were discarded. The third column of Table 1 shows a summary of observations left after the removal of empty observations.

Figure 4 shows line plots of consumption data of a faithful customer and an unfaithful customer against the consumption days, for the duration of three months. Comparing the two graphs, we observed that the consumption behaviour of the honest customer is mostly uniform and has a predictable trend, while electricity thief's consumption behaviour takes different forms and is not predictable. We further carried out histogram analyses for both classes of customers, as shown in Figure 5.

From the histograms shown, we observe that for faithful customer's consumption data, statistical parameters mean, mode, and median are generally closer to the histogram centre as compared to unfaithful customer's consumption data. We did a similar analysis for many customers and found that an observation presented here is true for most of the dataset. From these observations, we argue that by defining outliers

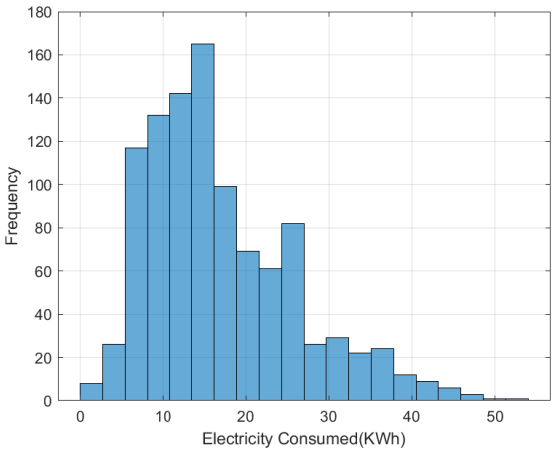


(a) Faithful customer's consumption plot

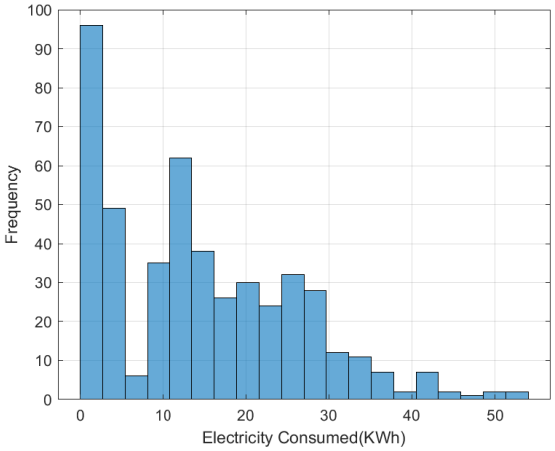


(b) Unfaithful customer's consumption plot

FIGURE 4. **Faithful and unfaithful customers' consumption plots.**



(a) Faithful customer's consumption histogram



(b) Unfaithful customer's consumption histogram

FIGURE 5. **Faithful and unfaithful customers' consumption histograms.**

as values beyond three Median Absolute Deviations (MAD), honest customers can be characterized as having fewer outliers percentage in a given data, than unfaithful customers.

2) DATA INTERPOLATION

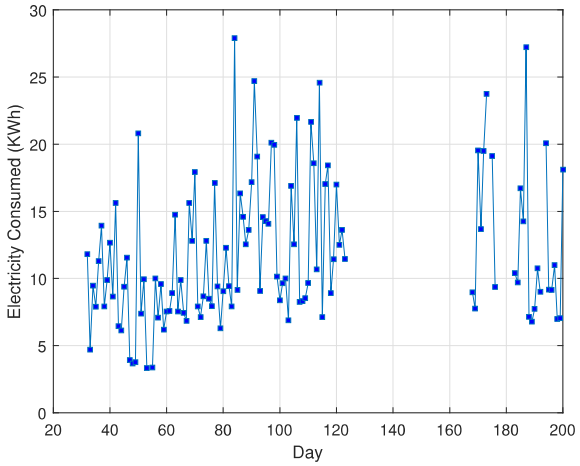
For all observations consisting of a combination of null or non-numerical values and real consumption values, data were interpolated. Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) [46] was used to fill in missing data during data interpolation while preserving consumption patterns.

A cubic Hermite interpolating polynomial $H(x)$ is a shape-preserving interpolant which preserves data monotonicity on a sub-interval $x_i \leq x \leq x_{i+1}$ applied to. For the data consumption vector containing *NaN* values at the beginning, the raw data *mean* was evaluated excluding *NaN* values and then inserted as the first vector element. The rest of the elements were filled in using PCHIP. This helped to maintain consumption shape and avoided adding outliers to data.

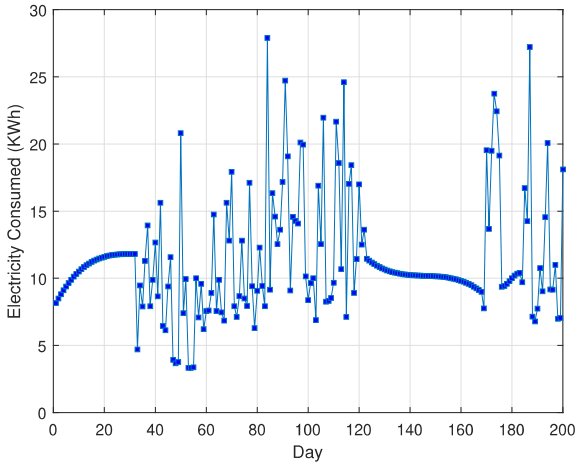
Figure 6 shows an example of one observation taken randomly before and after interpolation. A consumption duration of 200 days around days with missing consumption data is shown for clear presentation. Interpolated data points make a smooth curve that lies between the minimum and maximum near points with no overshooting, as can be seen from Figure 6b. In this manner, the consumption data is preserved from the addition of outliers and data points that can make interpolated data pattern to resemble unfaithful customer's consumption pattern of the minority class of unfaithful customers, such shown in Figure 4b.

3) SYNTHETIC DATA GENERATION

After eliminating empty observations and interpolating data, we carried out the initial classification process. Experimenting with the dataset as is, we observed that the classifier satisfactorily classified faithful customers and performed badly on unfaithful customers due to a *class imbalance problem* [20], [25]. A class imbalance problem is a situation whereby the number of observations in one class is much greater than



(a) Consumption data before interpolation



(b) Consumption data after interpolation

FIGURE 6. **Plots of consumption data before and after interpolation.**

the number of observations in the other class. In a class imbalanced problem, classification models classify the majority class on a dataset successfully, while performing badly on the minority class [25]. Dataset used in this work has faithful customers number that is much greater than that of unfaithful customers.

We solved the class imbalance problem in the following manner:

1) Define q and r as the number of faithful and unfaithful customers respectively and evaluate the difference $p = q - r$.
2) From a set of faithful customers observations, randomly select p observations represented by $p \times 1034$ matrix $\mathbf{O}$ defined by Equation (15), such that

$$\mathbf{O} = \begin{bmatrix} o_{1,1} & o_{1,2} & \cdots & o_{1,1034} \\ o_{2,1} & o_{2,2} & \cdots & o_{2,1034} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ o_{p,1} & o_{p,2} & \cdots & o_{p,1034} \end{bmatrix}. \quad (15)$$

3) Inspired by [20] and dataset analysis observations in IV-A, we evaluated synthetic observations $\mathbf{O}^s$ by Hadamard product in Equation (16).

$$\mathbf{O}^s = \mathbf{C} \odot \mathbf{O} \quad (16)$$

where $\mathbf{C}$ is a matrix of randomly generated numbers of size $p \times 1034$ with elements between 0 and 1.
This helps to distort the pattern of consumption as observed through faithful customers' consumption data line plots shown in IV-A; hence the result better represents unfaithful customers' consumption data.

This approach of generating synthetic is cheap and fast to do as it uses the available data of faithful customers' class to generate data for the opposite class. It involves a single operation on the measured data, which is multiplication of measured data by a matrix of randomly generated numbers. The resulting data was added to the original dataset, labeling it as belonging to unfaithful customers consumption class. The fourth column in Table 1 shows a summary of observations after synthetic data generation.

B. FEATURE EXTRACTION

Electricity consumption data used in this project is *univariate* time-series data. A univariate measurement is a single measurement frequently taken over time [47]. For solving classification problems, data can be represented by its features (properties), which can then be fed as input to the classifier, as is the case in [29], [34] and [48]. Data is classified based on the similarity between features [47] given a dataset of different samples. In this work, time-domain and frequency-domain features were extracted and used as input to a deep neural network for classification. Classification performance comparison between time-domain, frequency-domain and combined features from both domains was carried out.

1) TIME-DOMAIN FEATURE EXTRACTION

As shown in IV-A, faithful and unfaithful customers' consumption data differs clearly by a pattern of consumption, as shown by line plots and histogram graphs. Based on this information, time-domain features stipulated in Table 2 can collectively be used to differentiate between the two classes of customers. Apart from an observation that consumption data of faithful customers roughly follow a predictable pattern, and

TABLE 2. **Time-domain and frequency-domain features table.**

Time-domain features	Frequency-domain features
Standards probability (stdsProb)	Harmonic frequency (hfInd)
Standards mean (stdsMean)	Harmonic frequency amplitude (hfAmp)
Standards standard deviation (stdsDev)	99% spectrum bandwidth (bww99)
Outliers probability (outsProb)	Lower bound frequency (flb)
Outliers mean (outsMean)	Upper bound frequency (fub)
Outliers standard deviation (outsDev)	99% bandwidth power (bwwpwr)
Data mean (dataMean)	50% bandwidth (bw50)
Data mode (dataMode)	Median frequency (fmed)
Data median (dataMedian)	Mean frequency (fmean)
Average of pchip interpolant curve fitted parameters (cfpMean)	

unfaithful customers consumption behaviour is not predictive, as shown in Figure 4, customers do not consume an equal amount of energy per given time. Energy needs per customer may differ due to different reasons such as the number of appliances used, kind of appliances per household, household size, etc. To achieve higher accuracy in classifying features, all observations are made to fit within the same axes. This is achieved by normalizing data for each observation using the Min-Max method [49] given by Equation (17). The Min-max method shrinks the data between 0 and 1 while keeping the original consumption pattern.

$$f(x_i) = \frac{x_i - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})} \tag{17}$$

2) FREQUENCY-DOMAIN FEATURE EXTRACTION

Fourier theorem states that a periodic signal $x(t)$ can be represented by a summation of complex sinusoidal signals with frequencies that are an integer multiple of fundamental frequency f_T [50]. Using the Fourier theorem, the consumption data graphs shown in IV-A can be seen as a time series signal that can be transformed into the frequency-domain by using Fourier transform. Represented in frequency-domain, we extracted frequency-domain features from each observation. Since neural networks are sensitive to diverse input data, using Equation (17), features were normalized after being extracted so that they could be fed as input to the classifier. Table 2 shows features extracted from both domains.

C. CLASSIFICATION

1) NETWORK ARCHITECTURE

A fully connected feed-forward DNN architecture shown in Figure 7 was used for the classification process.

In order to avoid network underfitting and overfitting [35], the following rule of thumb methods [35], [51] were considered in the design of hidden layers of a deep neural network classifier shown in Figure 7:.

- Number of hidden neurons should be between the size of the input layer and size of the output layer,
- Number of hidden neurons should be approximated to the summation of $\frac{2}{3}$ size of input layer and size of the output layer.
- Number of hidden neurons should be less than twice the size of the input layer.

Rectified Linear unit (ReLU) activation function was used in the hidden neurons because of its better convergence property in comparison to other activation functions [28].

2) TRAINING

The maximum number of training iterations was limited to 1000. The classification approach was split into four parts. In the first part, only time-domain features were used for classification. In the second part, only frequency-domain features were used. The third part comprised of combined features from both domains, while in the last part, classification was performed in reduced feature space by incorporating PCA.

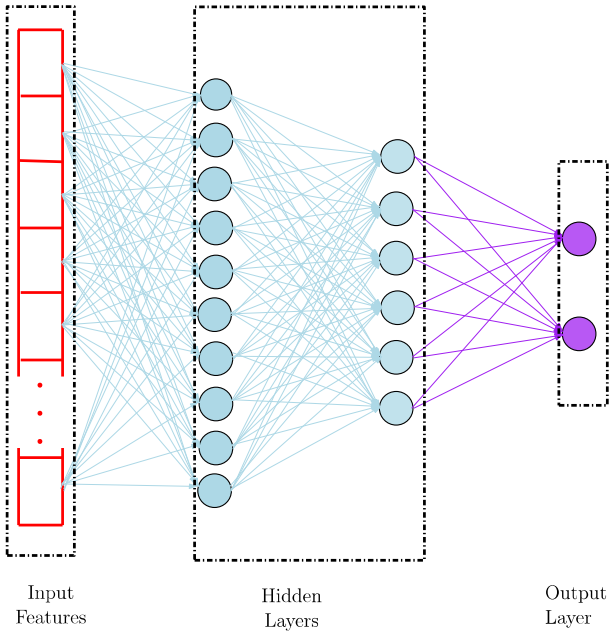


FIGURE 7. DNN classifier architecture.

Holdout validation scheme was used as follows: in all the procedures, as a rule of thumb, 80% of the whole data was used for training and validation, while 20% of the whole data was used for testing. Within training and validation data, 80% was used for training while 20% was used for validation. Similar results were obtained when using k-fold cross-validation scheme with $k = 5$. More about using k-fold cross-validation scheme with $k = 5$ can be seen in [52] as an example.

3) PERFORMANCE METRICS

Based on true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) obtained from a confusion matrix [41], we used the following performance metrics to evaluate the classifier's performance: Recall/True Positive rate (TPR) [53], [54], Precision/Positive Predictive Value (PPV) [53], [54], F1-Score [55], Matthews Correlation Coefficient (MCC) [25], Accuracy and Area Under the Curve of Receiver Operator Characteristic (AUC-ROC) curve [56]. We briefly introduce performance metrics used as follows.

Recall/True Positive Rate (TPR): is the measure of the fraction of positive examples that are correctly labeled. It is given by:

$$TPR = \frac{TP}{TP + FN}. \tag{18}$$

Precision/Positive Predictive Value (PPV): is the measure of the fraction of examples classified as positive that are truly positive. It is given by:

$$PPV = \frac{TP}{TP + FP}. \tag{19}$$

F1-Score: shows the balance between precision and recall. It is given by:

$$F1\text{-}Score = 2 * \frac{TPR * PPV}{TPR + PPV}. \quad (20)$$

Accuracy: shows the fraction of predictions classified correctly by the model. It is given by:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (21)$$

Matthews Correlation Coefficient (MCC): a single digit that measures a binary classifier's performance. Its value ranges from -1 to $+1$, with values closer to $+1$ signifying good performance, while values closer to -1 signify bad performance. MCC is given by:

$$\text{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \quad (22)$$

Area Under the Curve (AUC): measures the classifier's overall quality. Larger AUC values indicate better classifier performance.

4) HYPERPARAMETERS OPTIMIZATION

To achieve the best classification performance at a reasonable amount of time, we used the Bayesian optimization method [57] to tune the following hyperparameters: number of hidden layers, size of each layer, regularization strength and activation function. Bayesian optimization is derived from Bayes' theorem which states that for events A and B,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (23)$$

This optimization method determines the distribution of hyperparameters by assuming that an optimization function obeys the Gaussian distribution. To get the best combination of hyperparameters, 100 optimization steps were conducted. The resultant optimized network was trained and tested in a similar manner as the network in Figure 7.

5) IMPACT OF KEY PARAMETERS INVESTIGATION

Using adaptive moment estimation (Adam) optimizer [58], an impact of the following three key parameters were investigated on the optimized network: initial learning rate, minibatch size and l2-regularization parameter. Data was divided into two parts: the training and validation data.

The volume of the training and validation/test data plays an important role in classification success. The higher the correlation between input features and the class label, the lesser the data needed for training [59]. However, given a dataset, the training data portion of less than 50% is not adviced for as it will negatively affect the test results [59]. With this in mind, we therefore determined parameters' impact with different training data percentages.

TABLE 3. Investigated parameters table.

Parameter	Initial Value	Log Step	Final Value	Fixed Value
Initial learning rate	10^{-5}	$10^{0.03}$	10^{-2}	10^{-4}
Minibatch size	10^{1}	$10^{0.04}$	10^{5}	128
L2-regularization	10^{-8}	$10^{0.06}$	10^{-2}	10^{-5}

We carried out the following procedure for 60%, 70% and 80% training data portions. For each parameter, its impact was investigated by determining training and validation accuracies with varied parameter values. Parameters were logarithmically varied in 100 steps between the initial and final values. For each step, the number of training epochs was limited to 30. The other parameters were held at fixed values while adjusting a parameter under study. Table 3 shows investigated parameters' initial values, step values, final values as well as fixed values.

V. RESULTS AND DISCUSSION

In this section, we show and discuss the experimental results. In Section V-A, we present results obtained before synthetic data generation. In Section V-B, we show a comparison between classification performance when using time-domain features, frequency-domain features and combined features from both domains as inputs to the classifier. We analyze PCA dimensionality reduction impact on experimental results in Section V-C. We present Bayesian optimization results as well as best results attained with optimized classifier in Section V-D. We present an investigation of optimal parameter settings for best classification performance by varying different parameters using Adam optimizer in Section V-E, and we finally show how our method compares to data-based electricity theft detection methods in the literature in Section V-F.

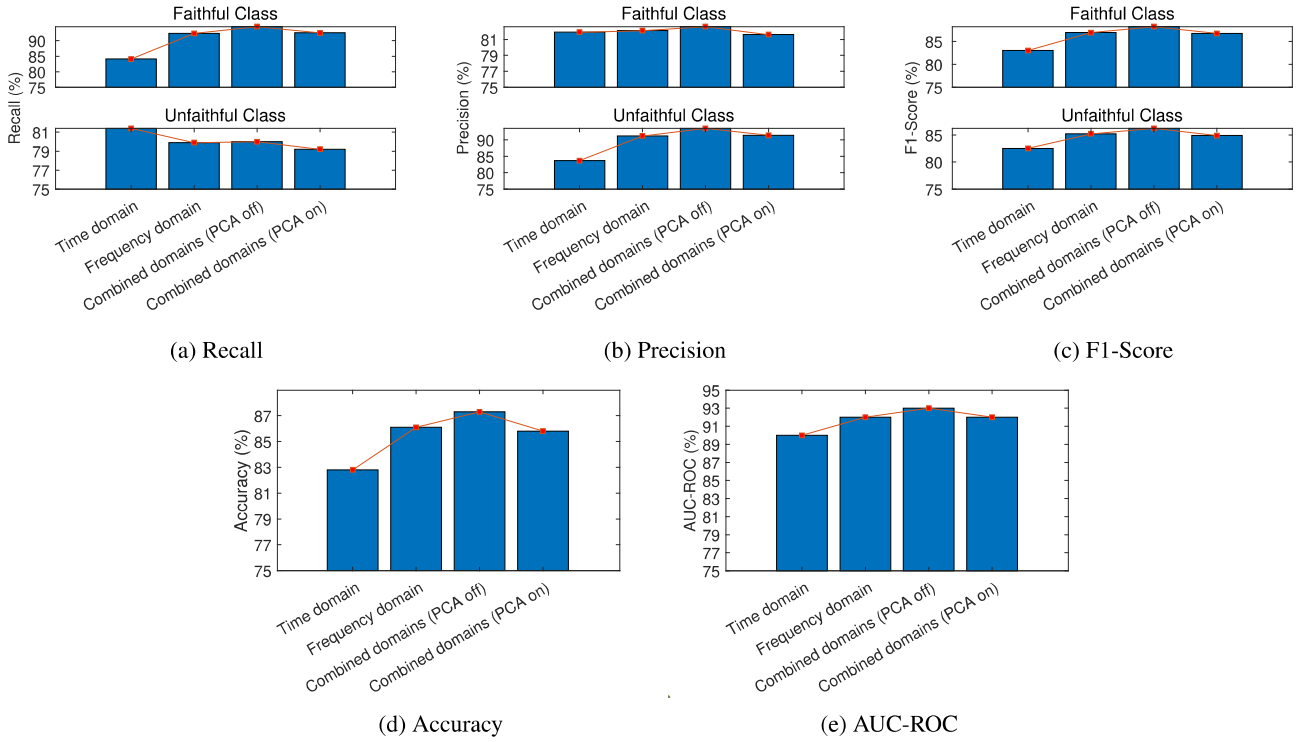
A. VALIDATION RESULTS BEFORE SYNTHETIC DATA GENERATION

As stated in Section IV, when there was an imbalance in the number of observations between two classes, the classifier performed badly on the class with a relatively lower number of observations. The classifier shown in Figure 7 was trained with features extracted from an original dataset with no augmented synthetic data. 80% of the data was used for training while 20% was used for validation. The third column of Table 4 shows the validation results. For the faithful customers class, validation results are much better than the unfaithful class. This can be seen by a comparison between faithful and unfaithful customers' recall, precision and F1-score shown.

Compared with validation results in combined domains before the incorporation of PCA, there was no significant change in the recall, precision and F1-score for faithful customers' class since the difference in corresponding values was within 1% margin. However, for the unfaithful class,

TABLE 4. Performance evaluation table.

Parameter	Class	Before synthetic data generation	After synthetic data generation							
			Time-domain		Frequency-domain		Combined Domains			
							PCA Not Used		*PCA Used*	
		Val(%)	Val(%)	Test (%)	Val(%)	Test (%)	Val(%)	Test (%)	Val(%)	Test (%)
Recall	Faithful	94.6	85.8	84.1	92.8	92.3	94.2	94.5	93.0	92.5
	Unfaithful	4.3	89.2	81.4	90.4	79.9	90.0	80.0	89.0	79.2
Precision	Faithful	91.4	88.8	81.9	90.6	82.1	90.4	82.6	89.4	81.6
	Unfaithful	6.9	86.3	83.7	92.7	91.2	93.9	93.5	92.7	91.4
F1-Score	Faithful	93.0	87.3	83.0	91.7	86.9	92.3	88.2	91.2	86.7
	Unfaithful	5.3	87.7	82.5	91.5	85.2	91.9	86.2	90.8	84.9
Accuracy		86.9	87.5	82.8	89.9	86.1	91.1	87.3	90.5	85.8
AUC-ROC		66	94	90	96	92	97	93	96	92
MCC = 0.84 (on validation) and 0.75 (on test).										



(a) Recall (b) Precision (c) F1-Score

(d) Accuracy (e) AUC-ROC

FIGURE 8. Performance metrics graphs.

which was the minority class, validation results in terms of recall, precision and F1-score were not good at all before balancing the classes. A significant improvement was obtained after balancing the classes. This shows that the sensitivity of the classifier to the minority class was not as good as its sensitivity to the opposite class.

The subsequent subsections show the results which were obtained after augmenting synthetic data to the original dataset to balance classes.

B. DIFFERENT DOMAINS FEATURES' CONTRIBUTION ANALYSIS

To ensure the reliability and robustness of the method introduced in this work, we present experimental results based on widely-accepted performance metrics summarized in Table 4. To simplify the analysis, classification performance between time-domain, frequency-domain and combined features from both domains is graphically presented in Figure 8.

From Table 4 and Figure 8, it can be seen that the classification process taken with time-domain features gave impressive validation and test results for both faithful and unfaithful customers classes. An experiment done with frequency-domain features alone showed improved results. The best results were obtained when all features from both domains were combined. For example, on validation, accuracy was 87.5%, which improved to 89.9%, and finally 91.1% when the experiment was done with time-domain features, frequency-domain features and all features from both domains respectively. The red trend line in Figure 8 graphs portrays significant improvement on results obtained from experiments done with time-domain features, frequency-domain features and all features from both domains. This improvement can be explained by a

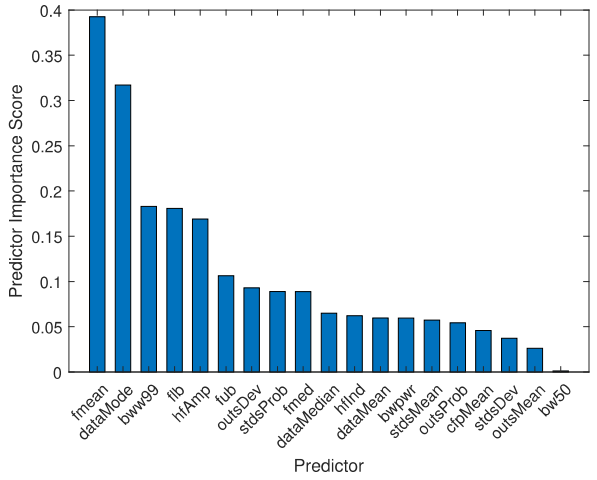


FIGURE 9. Features presented in order of their prominence.

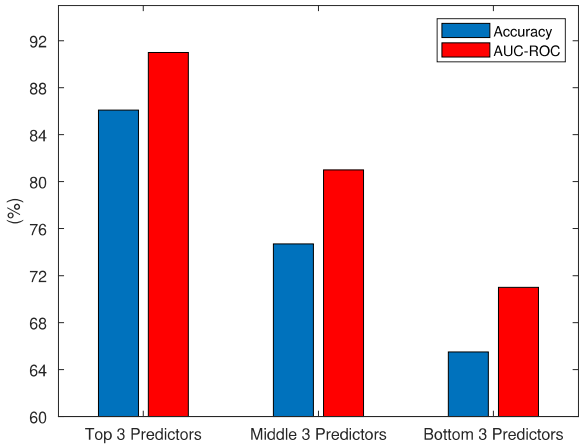


FIGURE 10. Classification results comparison of features ordered by mRMR scheme.

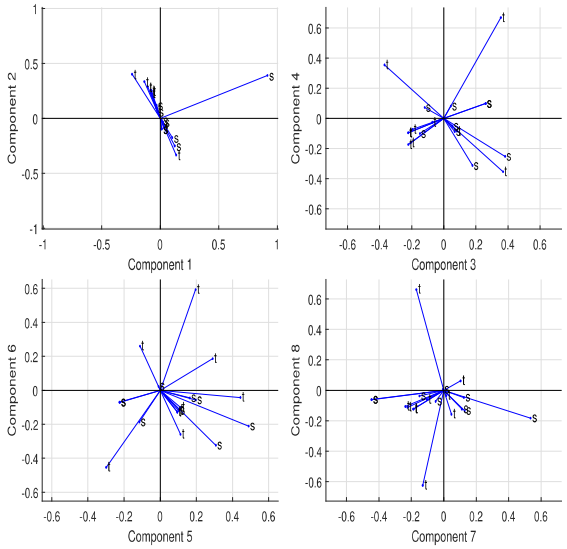


FIGURE 11. Graphical display of original features' contribution to principal components.

bar chart of predictors presented in order of their prominence shown in Figure 9, which has been produced through the mRMR scheme.

As shown by Figure 9 bar chart, there are more frequency-domain features to the left of the bar chart (i.e., features with the best scores) than time-domain features, with mean frequency achieving the highest predictor score. We confirmed the exactness of features' ranking through the mRMR scheme by doing classification tasks using top 3, middle 3 and bottom 3 features on the same network in Figure 7. Figure 10 bar chart shows classification accuracy and AUC-ROC results.

Comparing the results in Figure 10, we observed that accuracy and AUC-ROC are best for the top 3 features and worst for the bottom 3 features, as expected. MCC was determined on the last experiment when all features were combined. Its values were found to be 0.84 and 0.75 on validation and test respectively, which are closer to $+1$ than -1. AUC-ROC values were found to be 97% and 93% on validation and test respectively. These results portray a satisfactory overall classification task.

C. ANALYSIS OF COMPONENTS REDUCTION WITH PCA

When PCA was incorporated with the component reduction criterion of leaving enough components to explain 95% variance, seven components were left, having the following percentages contributions to total variance: 35.84%, 27.02%, 15.55%, 7.69%, 4.87%, 3.30% and 1.81%. Figure 11 shows 2-D biplots of original features contribution to each of the components in the principal components space.

Frequency-domain feature vectors are labeled with 's', while time-domain features are labeled with 't'. The contribution of each feature to the principal component is signified by that feature's vector direction and length. From Figure 11, we observed that frequency-domain features contributed more to principal components. This was also confirmed by features importance scores analysis shown by Figure 9 based on the mRMR scheme. The last two columns in Table 4 shows both validation and test results obtained after components reduction with PCA. We observed that with just seven principal components, we were able to achieve results very close to when no feature reduction criterion was used.

D. HYPERPARAMETERS OPTIMIZATION RESULTS

Following the hyperparameters optimization procedure stipulated in Section IV-C4, Figure 12 shows observed objective function values vs optimization steps.

The best hyperparameters combination was obtained at the 26^{th} optimization step and remained unchanged till the 100^{th} step. Their values are shown in Table 5.

An improved classification network architecture constructed with optimized hyperparameters achieved maximum validation and test accuracies of 91.8% and 88.1% respectively, which are 0.7% and 0.8% higher than an unoptimized

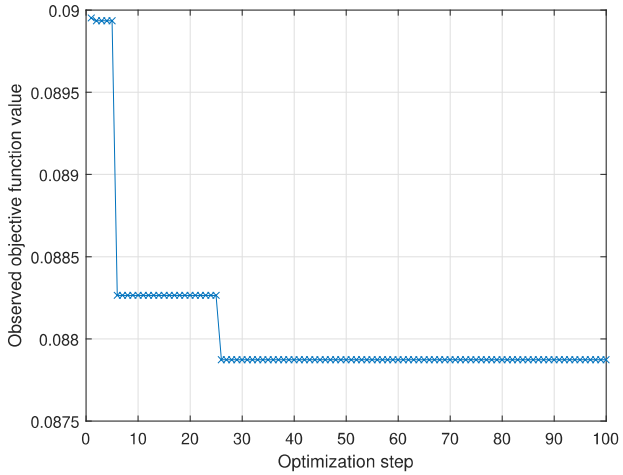


FIGURE 12. Objective function value vs optimization steps.

TABLE 5. Optimized hyperparameter values.

Parameter	Value
Fully-connected Layers	[41 21]
Regularization strength	5.6882×10^{-7}
Activation function	Sigmoid

architecture. The classifier obtained a maximum AUC-ROC value of 97%.

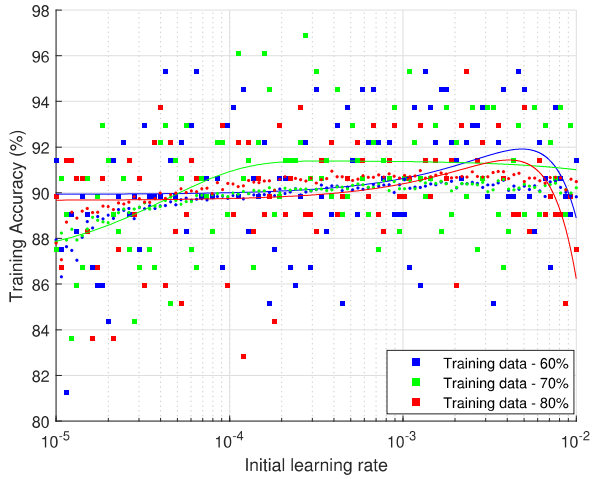
E. KEY PARAMETERS' IMPACT ANALYSIS

1) IMPACT OF INITIAL LEARNING RATE

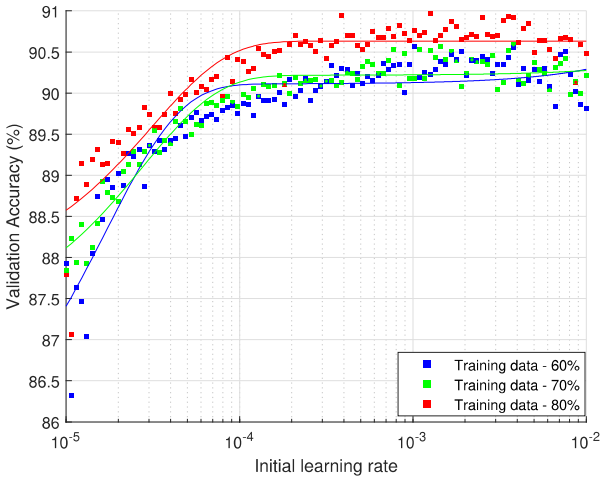
To determine the impact of the initial learning rate on training and validation accuracies, the initial learning rate was varied between 10^{-5} and 10^{-2} in 100 steps. Figure 13 shows scatter plots of the results with fitted curves to simplify analysis. For all tested training data portions, training and validation accuracies values were lowest for the lowest initial learning rates, with recorded values less than 90%. Significant improvement in both accuracies was seen for initial learning rate values between 10^{-5} and 10^{-4}. Low values of accuracies were attained in this range because lower learning rates require higher training iterations (hence more training time) for the model to converge to good results, therefore accuracy was mainly limited by the limited number of epochs allowed to train the network. Beyond the initial learning rate of 10^{-4}, average training and validation accuracies improved beyond 90%. For all training data portions, the best accuracy values were obtained for initial learning rate values in the range $[10^{-3.7}, 10^{-2.5}]$. Both accuracies started dropping as the initial learning rate approached 10^{-2}. For optimal results, an initial learning rate in the range $[10^{-3.7}, 10^{-2.5}]$ is recommended for best accuracy.

2) IMPACT OF MINIBATCH SIZE

To determine the impact of the minibatch size on the accuracy, the minibatch size was varied between 10^1 and 10^5 in 100 steps. We present training and validation accuracy versus



(a) Training



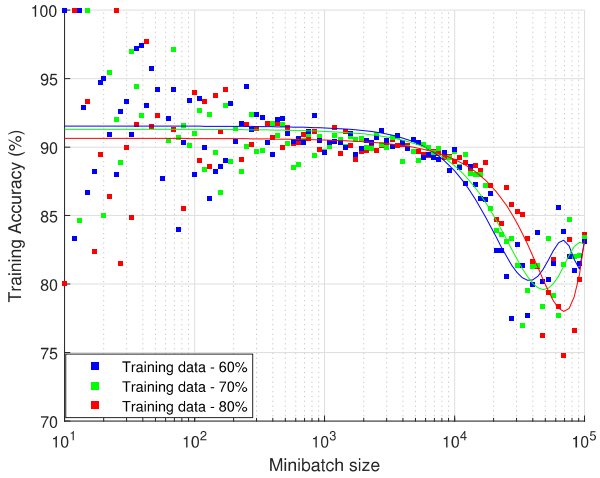
(b) Validation

FIGURE 13. Impact of varying initial learning rate on accuracy at different training ratios.

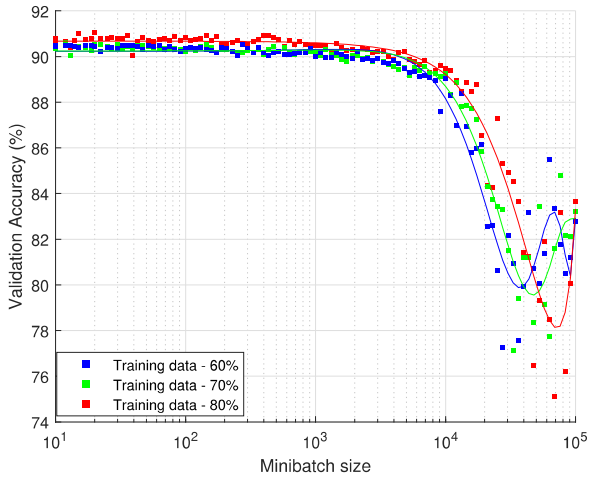
minibatch size parameter plots in Figure 14. For all tested training data portions, the training and validation accuracies averages were a little bit higher than 90% for minibatch size values less than 10^3. For minibatch sizes closer to 10^1, the training accuracy varied significantly between 80% and 100% for each training task, however, this did not have an impact on validation as validation accuracy stayed the same just above 90%. Both training and validation accuracies declined drastically as minibatch size increased beyond 10^4. This is because as the value of the minibatch size increased, the model had to learn from increased data size, resulting in poor generalization. However, smaller minibatch size values required relatively much time to train a model. A minibatch size less than but closer to 10^3 is recommended to balance efficiency and generalization.

3) IMPACT OF L2-REGULARIZATION PARAMETER

To determine the impact of the L2-regularization parameter on validation accuracy, the L2-regularization parameter

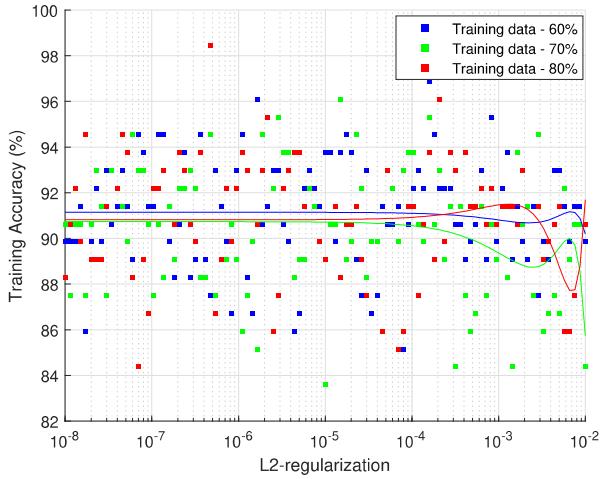


(a) Training

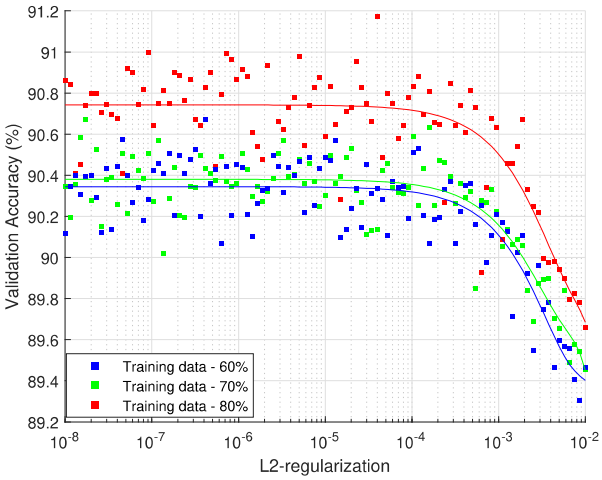


(b) Validation

FIGURE 14. Impact of varying minibatch size on accuracy at different training ratios.



(a) Training



(b) Validation

FIGURE 15. Impact of varying l2-regularization parameter on accuracy at different training ratios.

was varied between 10^{-8} and 10^{-2} in 100 steps. Figure 15 shows the results. For all training data portions, training accuracy laid between 83% and 99%, with an average value at around 91% for l2-regularization parameter values in the range $[10^{-8}, 10^{-4})$. Unstable average values of training accuracy were observed for l2-regularization parameter values $\geq 10^{-4}$. On the other hand, validation accuracy significantly decreased for l2-regularization parameter values $\geq 10^{-4}$. This may be caused by the fact that when the l2-regularization parameter $\geq 10^{-4}$, at each training iteration, a significantly large number of weights was left not updated, thereby making it hard for the model to converge to a good solution. Best results were obtained when L2-regularization parameter values were in the range $[10^{-8}, 10^{-4}]$.

For all investigated parameters, the best validation accuracy was obtained for the 80% training data portion, followed by the 70% training data portion and lastly 60% training data portion. This shows that the more data is available for training the model, the more accurate the model becomes in detecting electricity theft.

F. COMPARISON WITH EXISTING DATA-BASED ELECTRICITY THEFT DETECTION METHODS

Based on electricity customers consumption data, different data-driven methods have been used to tackle the electricity theft problem. Due to the scarcity of datasets containing both faithful and unfaithful customers' consumption data, many methods have been evaluated on different uncommon datasets. In Table 6, we present an analysis in the difference between our work and the recent works in the literature. For each work, dataset details are given. We look at the techniques and/or algorithms used, as well as features extracted from the data in respective methods.

For the four methods which used the same dataset as ours (References [3], [4], [27]), we compare the results in terms of AUC and accuracy percentages. We obtained AUC that is

TABLE 6. Comparison with existing data-based electricity theft detection methods.

Reference	Techniques/ Algorithms Used	Features used	Evaluation Dataset Details		Performance Evaluation
			Source	# of Customers	
[3]	SMOTE + KPCA + SVM.	Extracted from the original time-series data with KPCA.	SGCC	42372	Accuracy: 89% Precision: 85% Recall: 88%
[4]	Wide + deep CNN.	Wide and deep CNN used to learn features from the time-series data.	SGCC	42372	AUC: 79% Mean Average Precision (MAP): 96.9%
[23]	PCA + Calculation of anomaly score	PCs extracted from the original time-series data using PCA.	Irish smart meter data	5000	TPR: 90.9%
[24]	O-SVM + CS-SVM + OPF + C4.5 tree	Manually selected features from the original time-series data	Uruguayan electric power company	1504 3338 (two independent datasets)	Best accuracy: 86.2%
[25]	CNN + LSTM	CNN used to learn features from the time-series data.	SGCC	9956	Accuracy: 89% F1-score: 58.8%
[26]	LOF + k-means clustering	N/A	SGCC	3500	AUC: 81.5% MAP: 73.35%
[27]	SALM	AlexNet used to extract features from the original time-series data.	SGCC	42372	Accuracy: 90% AUC: 90.6%
[27]	GAN-NetBoost	GoogleNet used to extract features from the original time-series data.	SGCC	42372	Accuracy: 95% AUC: 96%
This work	**Feed forward DNN**	**Manually extracted time-domain and frequency-domain features.**	**SGCC**	**42372**	**Accuracy: 91.8% AUC: 97%**

1% higher than the best AUC in the benchmark and accuracy that is the second best. The results show that our work is very competitive against other methods recently undertaken.

VI. CONCLUSION

In this work, the detection of electricity theft in smart grids was investigated using time-domain and frequency-domain features in a DNN-based classification approach. Isolated classification tasks based on the time-domain, frequency-domain and combined domains features were investigated on the same DNN network. Widely accepted performance metrics such as recall, precision, F1-score, accuracy, AUC-ROC and MCC were used to measure the performance of the model. We observed that classification done with frequency-domain features outperforms classification done with time-domain features, which in turn is outperformed by classification done with features from both domains.

The classifier was able to achieve 87.3% accuracy and 93% AUC-ROC when tested. We used PCA for feature reduction. With 7 out of 20 components used, the classifier was able to achieve 85.8% accuracy and 92% AUC-ROC when tested. We further analyzed individual features' contribution to the classification task and confirmed with the mRMR algorithm the importance of frequency-domain features over time-domain features towards a successful classification task. For better performance, a Bayesian optimizer was also used to optimize hyperparameters, which realized accuracy improvement close to 1%, on validation. Adam optimizer was incorporated and optimal values of key parameters were investigated.

In comparison with other data-driven methods evaluated on the same dataset, we obtained 97% AUC which is 1% higher than the best AUC in existing works, and 91.8% accuracy, which is the second-best on the benchmark. The method used here utilizes consumption data patterns. Apart from its application in power distribution networks, it can be used in anomaly detection applications in any field. Our work brings a small contribution towards accurately detecting energy theft as we detect theft that only took place over time. We wish to extend our method to detect real-time electricity theft in the future. Since this method was evaluated based on consumption patterns of SGCC customers, it can further be validated against datasets from different areas to ensure its applicability anywhere.

REFERENCES

[1] S. Foster. (Nov. 2, 2021). *Non-Technical Losses: A $96 Billion Global Opportunity for Electrical Utilities*. [Online]. Available: https://energycentral.com/c/pip/ non-technical-losses-96-billion-global-opportunity-electrical-utilities

[2] Q. Louw and P. Bokoro, "An alternative technique for the detection and mitigation of electricity theft in South Africa," *SAIEE Afr. Res. J.*, vol. 110, no. 4, pp. 209–216, Dec. 2019.

[3] M. Anwar, N. Javaid, A. Khalid, M. Imran, and M. Shoaib, "Electricity theft detection using pipeline in machine learning," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2020, pp. 2138–2142.

[4] Z. Zheng, Y. Yang, X. Niu, H.-N. Dai, and Y. Zhou, "Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1606–1615, Apr. 2018.

[5] P. Pickering. (Nov. 1, 2021). *E-Meters Offer Multiple Ways to Combat Electricity Theft and Tampering*. [Online]. Available: https://www.electronicdesign.com/technologies/meters

[6] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid—The new and improved power grid: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 944–980, 4th Quart., 2012.

[7] M. Ismail, M. Shahin, M. F. Shaaban, E. Serpedin, and K. Qaraqe, "Efficient detection of electricity theft cyber attacks in AMI networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[8] A. Maamar and K. Benahmed, "Machine learning techniques for energy theft detection in AMI," in *Proc. Int. Conf. Softw. Eng. Inf. Manage. (ICSIM)*, 2018, pp. 57–62.

[9] A. Jindal, A. Schaeffer-Filho, A. K. Marnerides, P. Smith, A. Mauthe, and L. Granville, "Tackling energy theft in smart grids through data-driven analysis," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2020, pp. 410–414.

[10] I. Diahovchenko, M. Kolcun, Z. Čonka, V. Savkiv, and R. Mykhailyshyn, "Progress and challenges in smart grids: Distributed generation, smart metering, energy storage and smart loads," *Iranian J. Sci. Technol., Trans. Electr. Eng.*, vol. 44, no. 4, pp. 1319–1333, Dec. 2020.

[11] M. Jaganmohan. (Mar. 3, 2022). *Global Smart Grid Market Size by Region 2017–2023*. [Online]. Available: https://www.statista.com/statistics/246154/global-smart-grid-market-size-by-region/

[12] Z. Zheng, Y. Yang, X. Niu, H.-N. Dai, and Y. Zhou. (Sep. 30, 2021). *Electricity Theft Detection*, [Online]. Available: https://github.com/henryRDlab/ ElectricityTheftDetection

[13] D. O. Dike, U. A. Obiora, E. C. Nwokorie, and B. C. Dike, "Minimizing household electricity theft in Nigeria using GSM based prepaid meter," *Amer. J. Eng. Res.*, vol. 4, no. 1, pp. 59–69, 2015.

[14] P. Dhokane, M. Sanap, P. Anpat, J. Ghuge, and P. Talole, "Power theft detection & initimate energy meter information through SMS with auto power cut off," *Int. J. Current Res. Embedded Syst. VLSI Technol.*, vol. 2, no. 1, pp. 1–8, 2017.

[15] S. B. Yousaf, M. Jamil, M. Z. U. Rehman, A. Hassan, and S. O. G. Syed, "Prototype development to detect electric theft using PIC18F452 microcontroller," *Indian J. Sci. Technol.*, vol. 9, no. 46, pp. 1–5, Dec. 2016.

[16] K. Dineshkumar, P. Ramanathan, and S. Ramasamy, "Development of ARM processor based electricity theft control system using GSM network," in *Proc. Int. Conf. Circuits, Power Comput. Technol. (ICCPCT)*, Mar. 2015, pp. 1–6.

[17] S. Ngamchuen and C. Pirak, "Smart anti-tampering algorithm design for single phase smart meter applied to AMI systems," in *Proc. 10th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol.*, May 2013, pp. 1–6.

[18] B. Khoo and Y. Cheng, "Using RFID for anti-theft in a Chinese electrical supply company: A cost-benefit analysis," in *Proc. Wireless Telecommun. Symp. (WTS)*, Apr. 2011, pp. 1–6.

[19] J. Astronomo, M. D. Dayrit, C. Edjic, and E. R. T. Regidor, "Development of electricity theft detector with GSM module and alarm system," in *Proc. IEEE 12th Int. Conf. Humanoid, Nanotechnol., Inf. Technol., Commun. Control, Environ., Manage. (HNICEM)*, Dec. 2020, pp. 1–5.

[20] P. Jokar, N. Arianpoo, and V. C. M. Leung, "Electricity theft detection in AMI using customers' consumption patterns," *IEEE Trans. Smart Grid*, vol. 7, no. 1, pp. 216–226, Jan. 2015.

[21] W. Han and Y. Xiao, "A novel detector to detect colluded non-technical loss frauds in smart grid," *Comput. Netw.*, vol. 117, pp. 19–31, Apr. 2017.

[22] S. Sahoo, D. Nikovski, T. Muso, and K. Tsuru, "Electricity theft detection using smart meter data," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf. (ISGT)*, Feb. 2015, pp. 1–5.

[23] S. K. Singh, R. Bose, and A. Joshi, "PCA based electricity theft detection in advanced metering infrastructure," in *Proc. 7th Int. Conf. Power Syst. (ICPS)*, Dec. 2017, pp. 441–445.

[24] M. Di Martino, F. Decia, J. Molinelli, and A. Fernández, "Improving electric fraud detection using class imbalance strategies," in *Proc. ICPRAM*, 2012, pp. 135–141.

[25] M. N. 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.

[26] Y. Peng, Y. Yang, Y. Xu, Y. Xue, R. Song, J. Kang, and H. Zhao, "Electricity theft detection in AMI based on clustering and local outlier factor," *IEEE Access*, vol. 9, pp. 107250–107259, 2021.

[27] 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.

[28] K. Phil, *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*. Seoul, South Korea: Apress, 2017.

[29] S. Notley and M. Magdon-Ismail, "Examining the use of neural networks for feature extraction: A comparative analysis using deep learning, support vector machines, and K-nearest neighbor classifiers," 2018, *arXiv:1805.02294*.

[30] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," 2017, *arXiv:1710.02913*.

[31] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Speech Recognition, Computational Linguistics and Natural Language Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2020.

[32] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Trans. Signal Inf. Process.*, vol. 3, no. 1, pp. 1–29, 2014.

[33] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[34] PARISlab@UCLA. (Jul. 9, 2021). *Training an Artificial Neural Network With MATLAB—Machine Learning for Engineers*. [Online]. Available: https://youtu.be/xOzh6PMk21I

[35] J. Heaton, *Introduction to Neural Networks With Java*, 2nd ed. Chesterfield, U.K.: Heaton Res., 2008.

[36] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham, Switzerland: Springer, 2018.

[37] J. Zhang, "Gradient descent based optimization algorithms for deep learning models training," 2019, *arXiv:1903.03614*.

[38] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Jun. 2018, pp. 1836–1841.

[39] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci.*, vol. 6, no. 12, pp. 310–316, 2017.

[40] J. Feng and S. Lu, "Performance analysis of various activation functions in artificial neural networks," *J. Phys., Conf. Ser.*, vol. 1237, no. 2, Jun. 2019, Art. no. 022030.

[41] P. Dangeti, *Statistics for Machine Learning*. Birmingham, U.K.: Packt Publishing, 2017.

[42] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.

[43] H. Abdi, "Singular value decomposition (SVD) and generalized singular value decomposition," in *Encyclopedia of Measurement and Statistics*, N. Salkind, Ed. Thousand Oaks, CA, USA: Sage, 2007, pp. 907–912.

[44] M. Billah and S. Waheed, "Minimum redundancy maximum relevance (mRMR) based feature selection from endoscopic images for automatic gastrointestinal polyp detection," *Multimedia Tools Appl.*, vol. 79, nos. 33–34, pp. 23633–23643, Sep. 2020.

[45] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," in *Proc. Comput. Syst. Bioinf., IEEE Bioinf. Conf.*, Aug. 2003, pp. 523–528.

[46] C. Moler. (Mar. 3, 2023). *Splines and Pchips*. [Online]. Available: https://blogs.mathworks.com/cleve/2012/07/16/splines-and-pchips/

[47] G. Dong and H. Liu, *Feature Engineering for Machine Learning and Data Analytics*. Boca Raton, FL, USA: CRC Press, 2018.

[48] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 3026–3037, Apr. 2014.

[49] Codecademy. (Sep. 9, 2021). *Normalization*. [Online]. Available: https://www.codecademy.com/articles/normalization

[50] M. Fitz, *Fundamentals of Communication Systems*. New York, NY, USA: McGraw-Hill, 2007.

[51] B. Vega-Márquez, I. Nepomuceno-Chamorro, N. Jurado-Campos, and C. Rubio-Escudero, "Deep learning techniques to improve the performance of olive oil classification," *Frontiers Chem.*, vol. 7, p. 929, Jan. 2020.
[52] K. T. Chui, D. C. L. Fung, M. D. Lytras, and T. M. Lam, "Predicting at-risk university students in a virtual learning environment via a machine learning algorithm," *Comput. Hum. Behav.*, vol. 107, Jun. 2020, Art. no. 105584.
[53] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 233–240.
[54] Google-Developers. (Sep. 6, 2021). *Classification: Accuracy*. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy
[55] J. Brownlee. (Sep. 6, 2021). *Classification Accuracy is Not Enough: More Performance Measures You Can Use*. [Online]. Available: https://machinelearningmastery.com/ classification-accuracy-is-not-enough-more-performance-measures-you-can-use/
[56] A. Bhandari. (Sep. 6, 2021). *AUC-ROC Curve in Machine Learning Clearly Explained*. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/06/ auc-roc-curve-machine-learning/
[57] W. Jia, C. Xiu-Yun, Z. Hao, X. Li-Dong, L. Hang, and D. Si-Hao, "Hyper-parameter optimization for machine learning models based on Bayesian optimization," *J. Electron. Sci. Technol.*, vol. 17, no. 1, pp. 26–40, 2019.
[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.
[59] M. K. Uçar, M. Nour, H. Sindi, and K. Polat, "The effect of training and testing process on machine learning in biomedical datasets," *Math. Problems Eng.*, vol. 2020, pp. 1–17, May 2020.

LELOKO J. LEPOLESA received the B.Eng. degree in electronics from the National University of Lesotho (NUL), in 2017. He is currently pursuing the M.Sc. degree in electrical engineering with the University of the Witwatersrand (WITS), Johannesburg, South Africa.

He worked at Econet Telecom Lesotho as an OSS Engineer, from 2017 to 2021. His research interests include artificial intelligence and the Internet of Things (IoT).

SHAMIN ACHARI received the B.Sc. degree in electrical and information engineering from the University of the Witwatersrand (WITS), Johannesburg, South Africa, in 2015, where he is currently pursuing the Ph.D. degree in electrical engineering.

His M.Sc. studies was based in the field of visible light communication with special focus on error correction schemes for visible light systems, which was subsequently upgraded to the Ph.D. studies with a focus on channel modeling and methods of correcting synchronization errors. His research interests include visible light communications, machine learning, and the Internet of Things (IoT).

LING CHENG (Senior Member, IEEE) received the B.Eng. degree *(cum laude)* in electronics and information from the Huazhong University of Science and Technology (HUST), in 1995, and the M.-Ing. degree *(cum laude)* in electrical and electronics and the D.-Ing. degree in electrical and electronics from the University of Johannesburg (UJ), in 2005 and 2011, respectively. His research interests include telecommunications and artificial intelligence. In 2010, he joined the University of the Witwatersrand, Johannesburg, where he was promoted to a Full Professor, in 2019. He serves as an associate editor for three journals. He has published more than 100 research papers in journals and conference proceedings. He has been a visiting professor at five universities and the principal advisor for over 40 full research master's students. He was awarded the Chancellor's medals, in 2005 and 2019, and the National Research Foundation ratings, in 2014 and 2020. The IEEE ISPLC 2015 Best Student Paper Award was made to his Ph.D. student in Austin. He is the Vice-Chair of IEEE South African Information Theory Chapter.