REAL-TIME BIG DATA ANALYTICS WITH COMPUTATIONAL INTELLIGENCE APPROACHES FOR ENERGY LOAD FORECASTING

A Dissertation

by

DABEERUDDIN SYED

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2021

Major Subject: Electrical Engineering

ABSTRACT

In a real-time scenario of load forecasting, it is crucial to determine the future electric energy consumption in power distribution electrical networks. The electric energy forecasting models need to be updated with real-time trends of energy consumption as the analyzed energy consumption data exhibits high variability between historical and current data. This work proposes a multi-stage supercomputing-based big data analytics service for parallel and real-time load forecasting. Moreover, theoretical and experimental perspectives are proposed for multi-core parallel short-term load forecasting. Additionally, the knowledge from existing load forecasting based on deep learning models is used to innovatively develop highly accurate transfer learning models at different distribution nodes. Transfer learning models present practical applicability and productive possibilities in cases when sufficiently large data is not available. A novel approach based on deep neural network models is employed for load forecasting. Firstly, the electrical distribution nodes are grouped into different clusters with an aim to decrease the number of deep learning models to be trained. Secondly, network architecture information, weights, and biases are transferred from the first developed clustered model to subsequent models with an aim to reduce the training time of a large number of clustered models. And incremental learning is employed to incorporate newer data points for real-time processing and improving the forecasting accuracy of the clustered models on individual distribution points. Furthermore, parallel pool-based processing is employed to make efficient utilization of computing cores and to reduce the model development time further. The proposed big data real-time analytics methodology is evaluated on real-world energy consumption data collected from 105,148 Spanish electrical distribution transformers. The proposed methodology aims to reduce the number of trained models, training time, and execution time while still maintaining high prediction accuracy.

DEDICATION

This work is dedicated to my parents.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Haitham Abu-Rub, my committee co-chair, Dr. Ali Ghrayeb, and my committee members, Dr. Othmane Bouhali, Dr. Erchin Serpedin, Dr. Le Xie, and Dr. Shady S. Refaat for their guidance and support throughout this research.

Thanks also go to my friends, colleagues, department faculty, and staff for making my time at Texas A&M University and Texas A&M University at Qatar a great experience.

Finally, thanks to my mother and father for their encouragement and my wife for her patience and love. Special thanks to my sister who instills the spark in me serenely and I wish her all the luck.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Haitham Abu-Rub, Professor Ali Ghrayeb, Professor Erchin Serpedin, and Professor Le Xie of the Department of Electrical and Computer Engineering, and Professor Othmane Bouhali of the Department of Physics.

All the work conducted for the dissertation was completed by the student independently.

Funding Sources

This publication was made possible by NPRP grant [NPRP10-0101-170082] from the Qatar National Research Fund (a member of Qatar Foundation), and co-funding by IBERDROLA QSTP LLC. The statements made herein are solely the responsibility of the authors.

NOMENCLATURE

ANN	Artificial Neural Networks
AR	Autoregressive Model
AREM	Averaging Regression Ensemble Model
ARIMA	Autoregressive Integrated Moving Average
Bi-LSTM	Bi-directional Long Short-Term Memory
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DNN	Deep Neural Networks
DWT	Discrete Wavelet Transform
ELM	Extreme Learning Machine
EMD	Empirical Mode Decomposition
ICA	Independent Component Analysis
IoT	Internet of Things
ISOMAP	Manifold Isometric Mapping
LDA	Linear Discriminant Analysis
LR	Linear Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLR	Multi-Linear Regression
NMF	Non-negative Matrix Factorization

NN	Neural Networks
NPSO	Nested Particle Swarm Optimization
nRMSE	Normalized Root Mean Square Error
PCA	Principal Component Analysis
PULSE	Penalizing Underestimation Logarithmic Square Error
RBFNN	Radial Basis Function Neural Network
ReLU	Rectified Linear Unit
RF	Random Forests
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
RVFL	Random Vector Functional Link
SARIMA	Seasonal Autoregressive Integrated Moving Average
SSE	Sum of the Squares Error
SVM	Support Vector Machines
SVR	Support Vector Regression
TL	Transfer Learning
t-SNE	t-Distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection
WCSSE	Within-Cluster Sum of Square Errors

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION*[1]

1.1 Introduction

Electrical power system has recently witnessed massive developments. Technical developments have been witnessed not only in the power generation side but also in the transmission and distribution. Furthermore, the new technology is expected to revolutionize the end user side with adopting various demand management programs and techniques. For instance, the renewable energy sources such as solar and wind are not just added to the generation side by the utility companies, but also by the end consumers and microgrids. Moreover, vehicle to grid technology has provided the opportunities of power flow management and its flow from vehicles to the grid.

Once the electricity from renewable sources is increased to a large quantity, it would bring variability in the electrical system. This variability requires that innovative flexibility measures are considered to balance the demand and supply all the time. Novel approaches are required to improve the flexibility of the energy system ranging from supply to demand side. The concept of a smart grid and the use of big data analytics will help to manage the power systems better and also to increase resilience.

There has been and will be more massive installation of smart meters at the customer premises. These meters monitor the near real-time usage of energy and also collect and communicate the data to the electric utilities. The advent of the power system

[1] Reprinted with permission from "Smart Grid Big Data Analytics: Survey of Technologies, Techniques, and Applications." by Dabeeruddin Syed, Ameema Zainab, Ali Ghrayeb, Shady S. Refaat, Haitham Abu-Rub, and Othmane Bouhali, 2021. IEEE Access, 9, 59564-59585, Copyright 2021 by Dabeeruddin Syed.

deregulation on the delivery side and the moving away of the vertically integrated utility business model has also contributed to the development of the smart grids. Smart grid principle solves the power demand problems by providing two-way power and information flow between the consumers and utility [1]. Smart meters have been installed across the world in the past years along with the transformation of the traditional power grid into smart grid. The development of smart grid is fully associated with the big data flow. There are various potential applications of big data analytics on smart grid data such as the real time and automatic processing of the electrical consumer's energy consumption, automatic billing, intelligent energy planning and pricing, detection of outages due to faults and anomalies, load and generation forecast under high unpredictability, load management with demand response, and asset management [2]. High volume of data obtained from various smart grid sources satisfy the characteristics of big data. This grid data not only displays the Volume, Velocity and Variety characteristics but also the V's characteristics of Veracity, Visibility, and Value [3]. These characteristics are the challenges when dealing with the big data analytics along with other major concerns such as security, privacy [4].

The smart grid allows for the two-way energy and information flow between the consumers and utilities [5]. However, managing the real time data for making business valued decisions is still a persisting challenge [6], [7]. Currently, there are many utilities installing large number of smart meters and some of them have efforts to use the data. As an example, IBERDROLA has installed more than 11 million smart meters in Spain, generating 240 million registers every day [8]. Big data techniques over an estimated volume of 90 billion registers per year is being used to improve revenue collection and to optimize energy use. In smart grid, the number of smart sensors is much higher, and the generated data is significantly much larger.

More data utilization helps improving grid reliability and performance and ensures better decisions by the utility provider and customers which allows for effective demand side management and demand response [9]. However, the high volume of raw data is not directly comprehensible or useful without a dependable and consistent ability to process, analyze, and understand the information contained within such huge amount of data. Therefore, the data should be transformed into useful information before action can be taken based on it which is a complicated process because this beneficial information is not obvious from the data. The factors that contribute to the complications are the visualization and the use of data itself. Some information needs to be used by the automated systems while other information needs to be visualized and presented to people. Also, the time scales for different applications are different, from milliseconds to days. The challenges involved with the use of smart grid data for analytics can be categorized as 1) decisions on corresponding the data collection infrastructure to the desired applications, 2) application of new architecture and tools to manage grid data as streams in real-time, 3) transforming processes throughout the utilities to support the big data infrastructure, 4) managing the humongous amounts of data to make decisions that allow the benefits from the information obtained from smart grids data.

The smart grid contains a large number of sensors for various monitoring, communication, control, and management functionalities which enable effective, stable, reliable, and efficient operation of the power grid [10], [11]. Load forecasting plays a crucial role in the effective operation and management of large electrical systems [12], therefore this topic has been of high research and development focus during the past years [13]. A particular use of load forecasting is demand-side management (DSM) that plays an important role in creating the next smart grid energy paradigm and in improving the current grid efficiency and reliability.

The main challenge with the DSM at the residential and distribution levels is the need for high precision control and management which requires precise short-term and long-term load forecasting. Such DSM ensures proper and timely decision making on the power purchase, generation, and consumption. However, it is very difficult to predict the operation and consumption of the largely expanding electrical systems. The factors that affect the electric power demand and makes its forecasting a challenge are the differences in weather, season changes, weekends and holidays, operation scenarios of the power plant, faults occurring on the networks, economic growth, population growth, incentives, and others. The better demand prediction gets, the more effective is the DSM also in reducing the outages and blackouts.

Load forecasting helps the utilities to plan the amount of generation, load switching and infrastructure development. The load forecast can be divided into three main categories: Short-term forecast: ranges from few hours to few weeks [14]; Medium forecast: ranges from few weeks to few years [15]; Long-term forecast: ranges more than a year [16].

The main features for different categories of the forecast are short-term forecast includes time factor, weather data, and possible customer classes. Short-term load consumption depends on the time of the day or day of the week. For hotter regions, the energy consumption is higher during the noon time. Weather usually refers to temperature, humidity, cloud cover, wind speed, and Ultraviolet (UV) index.

The major factors that can be used for medium and long-term load forecast include [17]: historical load data, historical weather data, number of customers in different categories, age and characteristics of customers, electrical appliances in the area, appliances sales data, and economic and demographic categories.

The load forecast enables the service providers to control and plan the generation of electricity ahead of time. This in turn helps providers to manage the peak load [18]. Deep learning techniques could be used for load forecasting. They can be incorporated in the modern electrical systems easily as the grids are getting smarter with various smart sensors, devices, and meters. These smart devices generate data at a very high rate and in high quantities and can benefit the forecasting process. The data are collected from various sources that include the generation plants, transmission systems, and distribution systems.

1.2 Problem Definition – Research Objectives

Electrical energy must be generated whenever there is a demand for it. It is crucial for electric utilities to estimate the load demand on their systems. To minimize the operating cost, electrical utilities use load forecasting to control the number of operating generator units. The electric energy consumption data are collected by smart meters at high velocity, variety, and volume; making the data characterized as big data. The smart meter data, representing the energy consumption and customer consumption behavior at the household level, enable the electrical utilities to perform capacity planning, capacity building, and efficient operations. The various and different data collection points in a large electrical grid require parallel and real-time processing of the generated data for creating an accurate load forecasting. However, providing parallel and real-time load forecasting is a challenge for the operation and planning of electrical power generation. It is highly crucial to optimize the tradeoff between the accuracy of forecasting models and the execution time for economic operation of power system.

1.3 Research Contribution

The goal of this research is to provide a big data analytics methodology for parallel and real-time load forecasting in smart grid. The key contributions of this research work are summarized as follows.

1. Machine learning pipeline for predictive models is developed. Insights to set the strategic direction for the enhancement of energy forecasting accuracy are generated.
2. A multi-stage deep-learning and clustering-based transfer learning methodology to forecast short-term energy consumption at distribution nodes in a large electrical network.
3. The proposed clustering layer aims to identify the distribution nodes that have similar trends/profiles of energy consumption and cluster these nodes together. The objective of clustering approach is to reduce the number of deep learning

models required to be trained and developed while still achieving accurate load forecasting.

4. A hybrid cross-model adaptation layer capable of transferring the knowledge from one deep learning forecasting model to others is proposed. This layer is aimed at and proven to reduce the training time of load forecasting models.
5. A decoupled weight regularized optimizer is proposed to eliminate the negative transfer learning between the dissimilar clusters in transfer learning layer.
6. The incremental learning is proposed to avoid the regeneration and retraining of the forecasting models. This layer allows for real-time and online fine tuning of the models and for enhancing the forecasting accuracy.
7. A data mining module that integrates the weather and location API is developed to study the impact of feature selection.
8. Multiprocessing and parallel processing strategy is developed to enhance the scalability of the proposed methodology.
9. A novel objective function is proposed to penalize the tendency of deep learning models to underestimate.

CHAPTER II

LITERATURE REVIEW*[2,3]

In this chapter, background and related works on smart grid data flow, big data analytics process, technologies in the literature for big data analytics and industrial applied solutions are discussed. Additionally, an introduction to machine learning, supervised algorithms, unsupervised methods, and dimensionality reduction techniques is provided. Moreover, the chapter discusses scope of big data analytics in smart grid, different applications possible and different literature efforts that utilize various data-driven

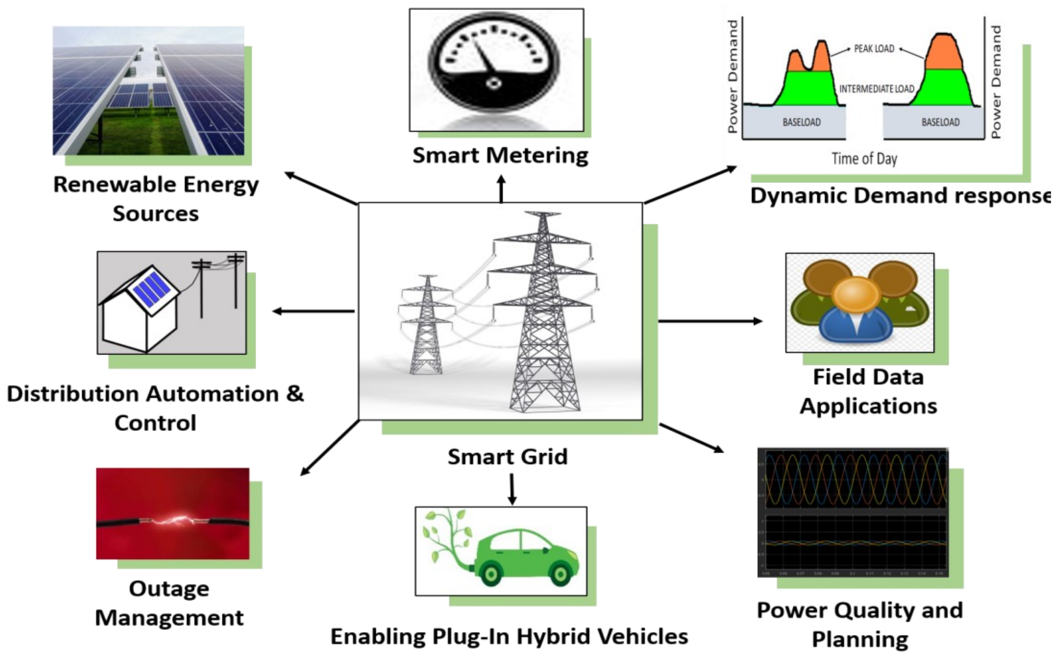


Figure 1 Smart grid as enabling engine - depiction of opportunities

[2] Reprinted with permission from "Smart Grid Big Data Analytics: Survey of Technologies, Techniques, and Applications." by Dabeeruddin Syed, Ameema Zainab, Ali Ghrayeb, Shady S. Refaat, Haitham Abu-Rub, and Othmane Bouhali, 2021. IEEE Access, 9, 59564-59585, Copyright 2021 by Dabeeruddin Syed.

[3] Reprinted with permission from "Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid with Clustering and Consumption Pattern Recognition." by Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Shady S. Refaat, Mahdi Houchati, Othmane Bouhali, and Santiago Bañales, 2021. IEEE Access 9, 54992-55008, Copyright 2021 by Dabeeruddin Syed.

methodologies for load forecasting, in addition to their methodologies of data acquisition, analysis, and performance.

2.1 Smart Grid Data Flow

A smart grid is formed by the integration of information and communications technology, electrical networks, and automation. The smart grid as an enabling engine is depicted in Figure 1. The electrical networks in the smart grid require the deployment of smart meters, sensors, devices, and control strategies. These have evolved due to the integration of renewable energy sources that are normally considered variable and unreliable sources of energy and are completely clean to the environment [19]. The smart grid aims to incorporate all the energy sources to match not only the baseline load but also the intermediate and peak loads.

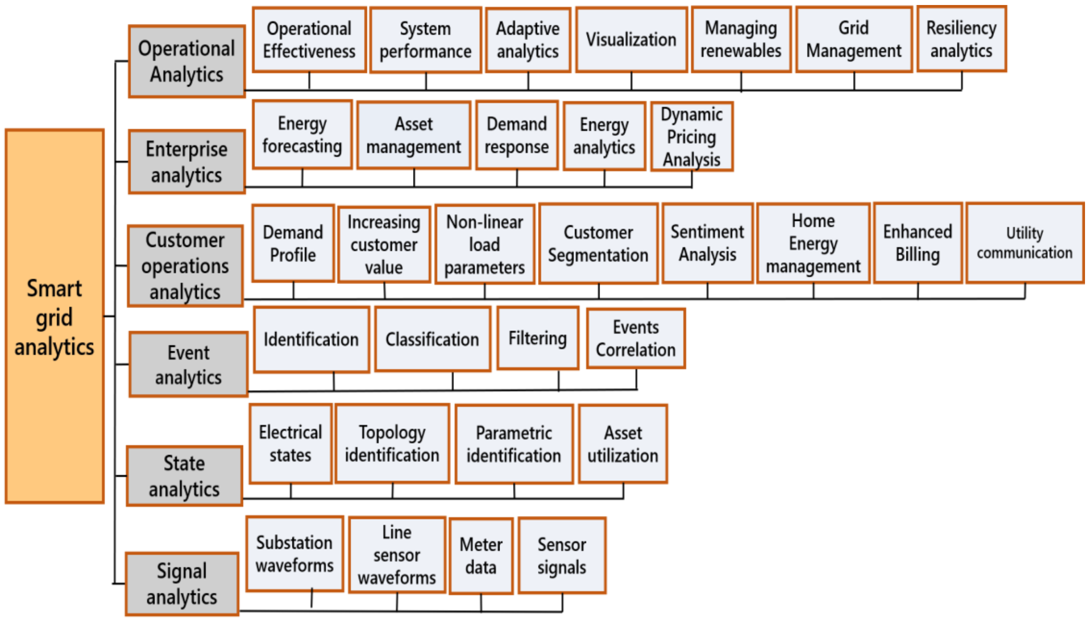


Figure 2 Scope of Big Data Analytics in Smart Grid

In the smart grid, there is a lot of scope with big data analytics apart from creating intelligence and obtaining information from the raw data [20], [21]. The scope of big data analytics has been illustrated in Figure 2. It is required that the big data methodology provides the potential to perform different types of analytics on the voluminous data to interpret it and derive business-valued applications. The different application areas for big data analytics in a smart grid will be discussed.

2.1.1 Big Data Sources

The data from the smart grids is generated at a very high rate and volume and in real time [22]. Extracting information from smart grid data which is required for specific applications calls for deep insight into the sources of smart grid data. The data is obtained from the sensors, smart meters, grid devices, detectors, and SCADA. The collected signals relate to power utilization habits of consumers, phasor measurement, energy consumption, energy pricing and bidding, operation or financials for running the utility. Types of sensors and information obtained from those sensors are described in Table 1. Also, large data sets not related to grid such as weather data, GIS data should be used for situational awareness and decision making. Owing to security and privacy concerns, the electric utilities do not share the smart meter data publicly and this poses a challenge to the research community. There are several benchmarks and publicly accessible data that have been anonymized or semi-anonymized and that the researchers can use to validate the performance of their proposed modeling and data analytics methodologies. The summarization of the list of public data sources is given in Table 2 [23], [24].

Table 1 Smart Grid Sensors and Devices.

Sources	**Quantity being measured**	**Information extracted and applications**
Advanced Metering Infrastructure (AMI)	E, Cumulative energy usage, peak load, load curve, phase, failure counts & logs, P.F., tamper factor, last interval demand	Market pricing, real-time on demand, remote meter configuration, demand-side management, electric usage, power quality monitoring, and local control
Distributed Generation Sensors	V, P.F.	Load balancing
Digital Fault Recorder (DFR)	Power swing, load variation, transient phase angle changes, frequency fluctuations, also records power system events such as time of fault, and power disturbance.	Faults classification
Electrical Measurement Sensors (EMS)	V, I, E, V_{sag}, P.F., Q_{reac}, electric & magnetic fields	Revenue
Fibre Bragg Grating sensor (FBG)	wavelength shift under changes in strain & temperature	Prediction of overheating, sag, vibration, galloping
Geographical Information System (GIS)	GIS data	Asset management & map the location of outages
Hall Effect sensor	V and magnetic field	Current sensing, proximity switching, positioning, speed detection
High Voltage Line Temperature and Weather Condition Sensors	T, record weather conditions	Preventive maintenance
Intelligent electronic device (IED)	Records status changes in substation and outgoing feeders	Relay protection
Line Fault detectors	V, I, P, harmonics, phase angle	Transmission or Distribution faults
Magnetoresistive sensors	Current, power, total energy, frequency, modulation	Transient Magnetic Field, EMI in substation
Phasor Measurement Unit (PMU)	V, I, P, harmonics, phase angle	Time synchronized measurements with phase angles, electrical waves measurement of power grid
Remote Terminal Unit (RTU)	Transmits telemetry data and controllable by micro-processor	system operation status
Smart Capacitor control	V, I, VAR and harmonic monitoring	Monitoring & control of capacitor banks remotely

Sagometer	T	Line Sagging
Smart Sensors for Outage Detection	T, I	Outage detection
SCADA	V, I, E, P.F.	Automatic control, protection, system monitoring, event processing and alarm
Smart Sensors for Transformer Monitoring	V, I, T, load tap changer values, partial discharge, dissolve gas data	Preventive maintenance
Smart Voltage Sensors	V	Voltage Regulation
Wide area monitoring system (WAMS)	Deals with incoming data from PMUs	Dynamic stability of the grid
V = Voltage, I = Current, P = Power, E = Energy, V_{sag} = Voltage sag, P.F. = Power factor, T = temperature		

Table 2 Publicly accessible data sources.

Data Source Name	**Data Description**
Ausgrid network [25]	Load profile data at the substation level.
Commission for Energy Regulation (CER) smart metering project [26]	Smart meter data from Ireland.
Cornell campus smart grid [27]	Smart meter data.
The École polytechnique fédérale de Lausanne (EPFL) smart grid data (Switzerland) [28]	PMU data.
Electric Reliability Council of Texas (ERCOT) data [29]	Market data.
North American SynchroPhasor Initiative (NASPI) data [30]	PMU data.
Pecan Street project [31]	Smart meter data.
Pennsylvania-New Jersey-Maryland (PJM) market data [32]	Market data.
Residential or commercial data [33]	Consumption, electric vehicles, power quality, PV generation, reliability, weather, wind-based generation, and general energy data.
University of California (UC) Berkeley campus smart grid [34]	Smart meter and building consumption data.

2.1.2 Data Structures

Contrary to the traditional data analysis, big data analysis deals with semi-structured, quasi-structured and unstructured data in addition to the structured data [35], [36].

• Structured data: Structured data is the data that comprises clearly defined data types, structure, and format whose patterns make the data easily searchable. Few examples include data that can be stored in spreadsheets, Comma-separated Values (CSV) file, a traditional Relational Database Management System (RDBMS), data cubes in Online Analytical Processing (OLAP), relational tables containing customer information, and electrical consumption data in numbers or strings. Meters' data, distribution management data, equipment parameters, load control data, and marketing system data in relational format are examples of structured data in smart grids.

• Semi-structured data: Semi-structured data is textual data that contains perceptible data patterns and enables parsing. For example, the XML and JSON data files are self-describing and defined by its schema. Web service data, load monitoring, and power quality data are examples of semi-structured data in smart grids.

• Quasi-structured data: Quasi-structure data is textual data that contains erratic data formats but can be properly formatted with tools after time and effort. The only difference between the semi-structured and quasi-structured data is that semi-structured data has metadata associated with them and the metadata can be easily used to structure or format the data. Whereas the quasi-structured data requires intelligence-aware approaches to structure or format them. For example, web clickstream that contains erratic formats and data values, web scrapping data, and search engine results are quasi-structured data.

• Unstructured data: Unstructured data is data that has no pre-defined models or schema. Examples include publicly collected census and text, social media streams and tweets, audio, video, and photographs. Meteorological information, customer service data, and economy data of distribution regions are examples of unstructured data in smart grids.

The high-level view of the data flow into the utility is illustrated in Figure 3 [37]. The first step is the data collection in which the major classes of data are collected from various sources, e.g., the customer data is collected using smart meters, grid data is measured on distribution and transmission lines using PMUs and synchrophasors. Other important data, that are collected, include SCADA data, market data, weather data, and customer feedback in the form of tweets, text, videos, audio, and pictures. The complex and heterogeneous data from multiple sources are then transmitted through various communication networks and stored in the relational database, data warehouse, file servers, application servers, and Hadoop clusters. This comes under the phase of data management where the data undergoes extraction, cleaning, aggregation, and encoding. Finally, the data are loaded into any in-memory distributed databases for further analytics. The third phase is analytics where the actual information stored in data is extracted to represent business value. The data analytics is performed using approaches such as time series analysis, feature selection, feature extraction, machine learning modeling, deep learning modeling, clustering, incremental learning, adaptive learning, and reinforcement learning with an aim to enhance applications for enterprise intelligence, grid operations, and customer insight. The applications may include the following but are not limited to: load profiling, load forecasting, demand response, program marketing, outage management, and bad data detection [21], [37], [38]. Finally, the information should enable action in the form of automation, external communication, and monitoring through visualization and dashboards.

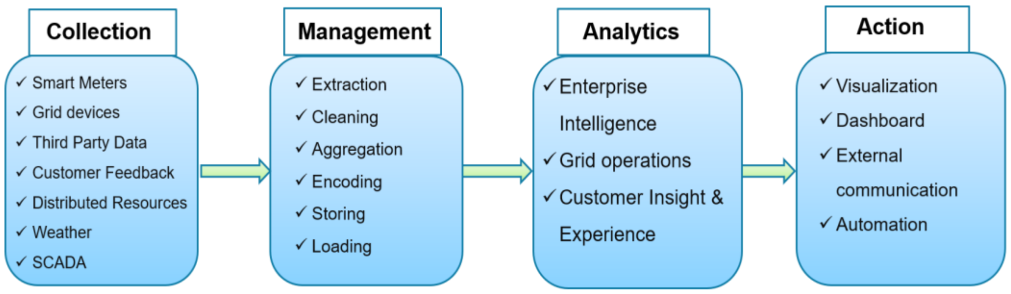


Figure 3 High level view of flow of data into the utility

2.2 Big Data Analytics Process

Big Data analytics requires pre-defined strategies because of the high volume of data. Also, the velocity and variety of data pose challenges in the data analytics process. It is very crucial that the data from the smart grid are processed in real-time because significant patterns can be recognized from the data to make better decisions. Data

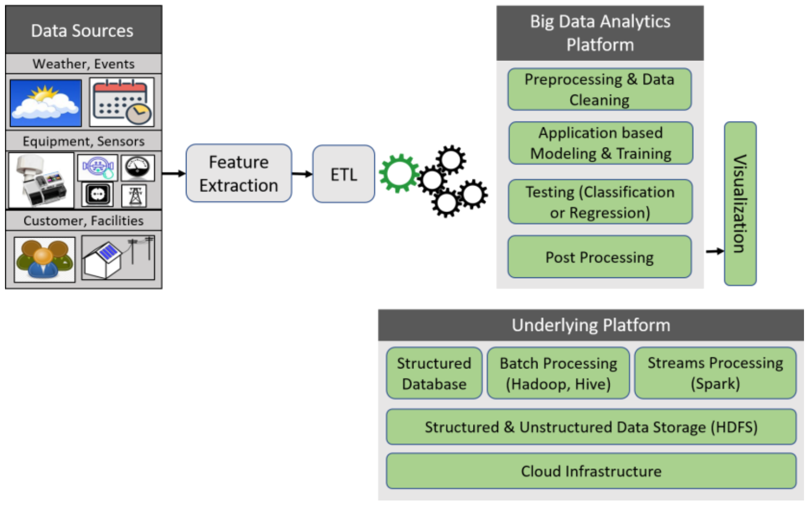

Data Sources
Weather, Events
Equipment, Sensors
Customer, Facilities
Feature Extraction
ETL
Big Data Analytics Platform
Preprocessing & Data Cleaning
Application based Modeling & Training
Testing (Classification or Regression)
Post Processing
Visualization
Underlying Platform
Structured Database
Batch Processing (Hadoop, Hive)
Streams Processing (Spark)
Structured & Unstructured Data Storage (HDFS)
Cloud Infrastructure

Figure 4 Big Data Analytics Process

analytics deals with the extraction of actionable knowledge and patterns from the available data [39]. The big data analytics process is illustrated in Figure 4.

There are four major types of big data analytics [40]. These are described as follows:

a) *Descriptive Analytics:* Descriptive analytics illustrates what happened in the past using the historical data available and shows the data in an easily understandable form or visualization. In general, the data is illustrated using graphs, bar diagrams, pie diagrams, maps, and scatter plots. In short, descriptive analytics is performed to understand or illustrate the patterns in the data.

b) *Predictive Analytics:* It extrapolates from the data available to predict what can happen in the future. The tools that are used for predictive analytics are time-series analysis using statistical methods and other data mining algorithms. Predictive analytics is usually performed to predict which events can happen in the future.

c) *Exploratory Analytics:* It finds hidden correlations or relationships between features in the data. This helps us to estimate values for a dependent feature when information is available for the independent features. Exploratory analytics is basically performed to determine the cause behind the events that have happened in the past.

d) *Prescriptive Analytics:* It is used to discover the best outcome of past events when the features of the data and operating parameters of a system are given. It helps to develop strategies for future events under similar conditions. The techniques involve simulation tools, and these simulate the operating conditions or features to finally come up with the best outcome. The simulation techniques strategize how to plan for similar events in the future. Prescriptive analytics is basically performed to know how preferable events can be made to happen in the future. Example: power flow analysis.

Data analytics starts with the acquisition of data following which the data is processed to reveal information.

2.2.1 Data Acquisition

The first step in any of the data analytics process is the collection of data. The data in the smart grid are collected from various sources as mentioned in the earlier section. With the data collection already in place, the other subtasks in data acquisition are data communication and data pre-processing. The raw data need to be transmitted either to a real-time stream processing system or to a storage system from where the data can be sent to the offline batch processing system for further analysis. Since the data have been collected from diverse and multiple sources, the data aggregation and cleaning are the foremost and crucial steps. Data aggregation services should be in place to integrate the data from varied sources and furnish a unified view of the available data. In data pre-processing, the inconsistent and missing data are to be filled or one among the records, and the features are to be removed to improve the data quality [41]. It is crucial to refine the features in the extracted data as there are noise and redundancy in the collected raw data. Refining the features involve either feature selection or feature extraction. If the data contain highly correlated features, then the machine learning algorithms, in general, perform poorly. Regularization techniques are used to overcome the issues of overfitting whereas underfitting would require the acquisition of more data and that is not an issue in the case of big data [42].

2.2.2 Data Processing

The data collected and transmitted should be stored in storage infrastructure for further processing. The stage, at which the data is processed, classifies data processing into the following types:

1) Batch Processing*:* Batch analytics is fundamentally the analysis of data in batches. It involves the workflow on offline data where all the data are available, pre-extracted, and ingested using scripts and a huge group of data is analyzed in a single execution. Distributed file systems (DFS) provide for the fault-tolerant scalable storage of data across commodity hardware where the storage nodes do not share memory however are connected virtually through networking [43]. MapReduce and Hadoop framework provides such a DFS framework. In MapReduce, a huge amount of data is processed by dividing the job into a set of sub-jobs and each sub-job handles a small portion of data and all the sub-jobs operate in parallel to obtain the intermediate outcomes. The final result is then obtained by the aggregation of the intermediate outcomes. The advantage of the MapReduce paradigm with respect to batch processing is the data locality principle. In this principle, the algorithm or the user code is moved close to data rather than moving the data to the algorithm. This requires the movement of computational resources to where the data is located and thus, prevents overhead from the data transmission. The disadvantage of batch processing is that it cannot provide analytics results in real-time. An example of batch processing in smart grids includes the training of data-driven models using offline data for applications of topology identification, predictive maintenance, and energy forecasting. These models would

require re-training if new data become available and need to be included in the modeling performance. There is no specific time interval defined to term processing as batch analytics. However, it is usually considered that if the processing is scheduled to happen with an interval equal to or greater than 20 minutes, then it is batch processing.

2) *Stream Processing:* Stream processing is primarily the processing of each new data instance as soon as it is available instead of waiting for batches of offline data. The idea behind the stream processing is that the potential worth of information from data relies on the freshness of data [44]. Hence, it is crucial that the stream processing model processes the data as soon as the data instance is available to obtain approximation results. If the data are continuously available in huge streams, a portion of the data can be stored in memory until it is processed.

In the subsequent sections, the technologies that possess the capability of processing big data in real-time are reviewed. They provide a huge advantage of handling data with high-velocity requirements. In one of the proposed works, the Hadoop File system is used as a storage system and spark streaming provides the real-time processing solution along with tools such as Spark Structured Query Language (SQL), Spark Machine Learning Library (MLlib), and GraphX. Examples of stream processing in a smart grid include stateless conversion, stateless filtering, aggregation, pre-processing, and wavelet transformations of the data. The time interval for data processing to be termed as stream processing is typically seconds or milliseconds.

3) *Iterative Processing:* There are few big data problems which require processing of data iteratively and demands for a greater number of read and write operations than the batch processing and stream processing and so involve more Input Output transfer and are time consuming.

For big data analytics on smart grid data, the batch and the stream processing are focused upon and the comparison of these processing types is given in the Table 3.

Table 3 Batch v/s Streams Processing.

	Batch Processing	**Stream Processing**
Input Form	Chunks of data	Streams of data
Size of Data Input	known & finite	unknown & infinite
Is Data Stored?	Stored Data	Data is not stored (or) small streams stored in memory
Hardware Used	multiple CPUs/memory	limited amount of memory
Processing	multiple rounds	single or few passes over data
Time	longer time	seconds or milliseconds
Applications	widely adopted	sensor networks, and web mining.

2.2.3 Data Analytics Techniques

Multiple machine learning algorithms are used as data analytics techniques in smart grids. These techniques are used to map the relationship between the features in the data and the prediction label. If the labels exist, the techniques employed are named as supervised techniques. Whereas the data may not explicitly consist of labels, and it is the algorithm that recognizes the patterns in the data. The techniques that analyze data without labels are termed as unsupervised techniques.

The summarization of the different classes of machine learning techniques, that have been previously applied in smart grids, is presented in Table 4, Table 5, and Table 6.

Table 4 Dimensionality Reduction Algorithms.

Algorithm	**Description**
Principle Component Analysis (PCA) [45]	Most widely used unsupervised technique; heuristic approach to extract variance structure from high-dimensional data; involves 1) feature scaling & mean normalization, 2) calculation of covariance matrix and 3) sorting the eigenvectors that represent components.
Linear Discriminant Analysis (LDA) [46]	Supervised technique; projection of data from higher dimensional space to lower one so that it maximizes between-class & minimizes within-class distances.
Kernel Discriminant Analysis (KDA) [47]	Obtains linear separation by non-linear mapping of input space to high-dimensional feature space.
t-distributed stochastic neighbor embedding (t-SNE) [48]	Converts high-dimensional data into a matrix of pair-wise similarities using conditional probabilities, and variation of stochastic neighbor embedding.

Table 5 Supervised Algorithms.

Algorithm	Description
Linear Regression (LR) [49]	Curve fitting regression technique for linear functions; the hypothesis function is linear.
Polynomial Regression [50]	Curve fitting regression technique for non-linear functions; the hypothesis is a linear model of basis functions (linear, polynomial, Gaussian Radial Basis Function (RBF), and sigmoid)
Logistic Regression [51]	Classification technique to identify decision boundary; the hypothesis function is sigmoid.
Neural Networks (NN) [52], [53]	Performs classification & regression; capable of modeling highly non-linear relationships with large feature space; parametric model; can represent complex logic operations & comprises input, hidden & output layers with activation functions (threshold, logistic, arctan, gaussian & ReLU); and types: convolutional, and recurrent.
Support Vector Machines (SVM) [54]	Large margin classifier; classifies non-linear data by introducing slack variables; SVM is found by minimization formulation under constraints that are overcome using a Lagrangian multiplier. Types: linear, and kernel.

Naive Bayes [55]	The parametric approach for likelihood estimation assumes that the data features are independent.
k-Nearest Neighbor (k-NN) [56]	Non-parametric approach for likelihood estimation; classifies a data point to the majority class among k Neighbors;
Decision Tree (DT) [57]	Recursive, partition-based tree model that predicts a class based on split points; the algorithm takes leaf size and purity threshold as inputs; the process stops when leaf size or purity threshold is reached.
Random Forest (RF) [58]	Collection of low-bias, high-variance trees; and outputs mode of the classes or mean prediction.

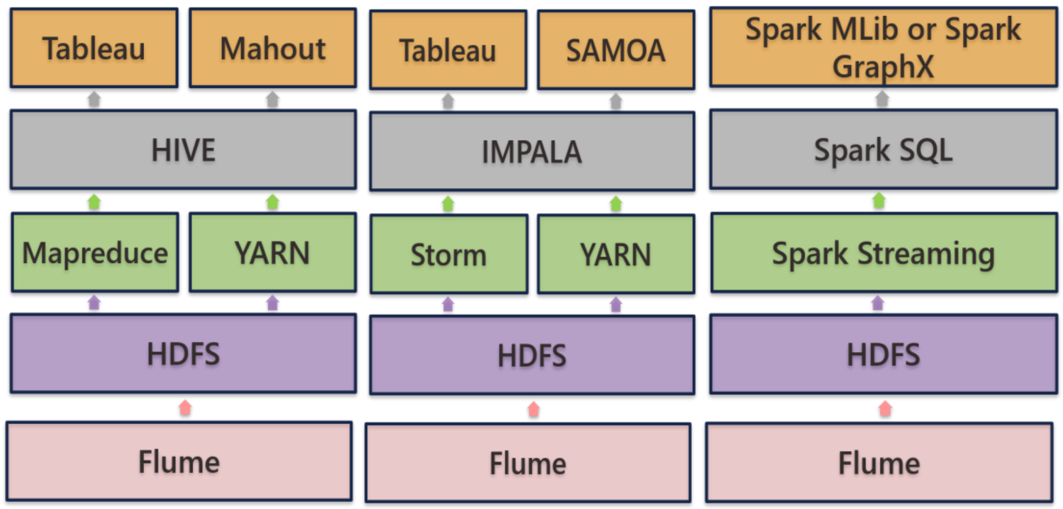
Table 6 Unsupervised Algorithms.

Algorithm	**Description**
K-Means Clustering [59]	The representative-based technique includes steps of initializing cluster centroids, grouping data points to nearest centroids, updating centroids, and uses Euclidean distances & variables are to be quantitative.
Expectation-Maximization Clustering [60]	The representative-based technique includes steps of initializing cluster mean, calculating posterior probability, and re-estimating means, covariance & priors.

Gaussian Mixture Clustering [61]	Fits k-Gaussians to cluster the data. The result is the weighted average of K-gaussian distributions.
Hierarchical Clustering [59]	Involves creating a sequence of nested partitions that can be visualized by a tree or hierarchy of clusters.
Density-based Spatial Clustering of Applications with Noise (DBSCAN) [62]	Density-based clustering that computes neighborhood to classify data points into core, border & noise points while also using a threshold called minimum points.
Association Rules [63]	Usually applied in market basket analysis, text mining, web usage mining, and linguistics mining to determine the co-occurrence relationships or associations between all items in the database.
Collaborative Filtering [64]	Generally employed in recommender systems where preferences of a target user are predicted based on the user searches where users are like the target & mining on their preferences.

2.3 Technologies for Big Data Analytics

In this section, a hierarchical architecture of state-of-the-art core components of big data analysis for smart grids using Hadoop is shown in Figure 5a, and the architecture that uses Storm is depicted in Figure 5b. In one of the works, a big data analytics platform is proposed and the technologies for big data analytics for smart grids using Spark is illustrated in Figure 5c [65], [66]. The major components perform the collection, storage, processing, visualization, and querying of data. There are a variety of workloads present in the scenario of massive-scale data analytics. A combination of these workloads will present a potentially effective solution for the business goals in the scenario of smart grids.



(a) Architecture using Hadoop (b) Architecture using Storm (c) Proposed Architecture

Figure 5 Different layers in one of the proposed platforms

2.3.1 Evolution of Big Data Technologies

When dealing with massive-scaled data, the framework was initially developed for the processing of offline large datasets. Apache Hadoop and MapReduce models provide opensource software frameworks for the distributed processing of offline data spread across data nodes or clusters using simple programming paradigms of the map and reduce functions. MapReduce abstracts from distributed programming but it still requires programming to a certain level. Moreover, MapReduce is efficient for batch tasks and not for adhoc queries or iterative processing. If the offline analysis or background task of indexing websites is required, then MapReduce is a suitable option. Hence, the combination of a distributed file system and MapReduce is suitable for write once and read many, or sequential data access, however not for random reading or write access applications [67]. Yet, random read/access is required for the online analysis of data or the ad-hoc querying.

As a solution to the ad-hoc querying issue, Not only SQL (NoSQL) databases can be used. NoSQL Databases are of two types [68]. These are mentioned in the following.

- Column databases: A column-oriented database is a database that stores data in columns rather than rows. Furthermore, it is very effortless to add columns and these columns can be added row by row as well. The databases offer great flexibility, performance, and efficiency. Also, the performance of the column databases can be significantly enhanced by compression, late materialization, and batch processing.

  Examples of column databases include BigTable, HBase in Amazon Dynamo, Google Bigtable, and Apache HBase.

- Key-value stores: These are distributed data structures that provide key-based access to data and are also called Distributed Hash Tables. An example is Apache Cassandra.

NoSQL Databases are very efficient when dealing with massive-scale data even if the data type is unstructured or semi-structured. However, the only disadvantage is that these do not offer SQL-like querying. To make querying SQL-like, many NoSQL databases have been evolved with the SQL-like interface (Contextual Query Language

(CQL) of Cassandra, Hive, and Pig.). There are developments in the form of SQL interfaces that can directly connect to the NoSQL databases (such as PrestoDB.). The SQL-like interfaced NoSQL databases are termed as NewSQL and these possess the inherent capability of organizing massive-scaled data and sorting to enable efficient offline analyses (H-Store, Google Spanner.) [69].

There has been a massive growth in the availability of digital data and the data are available in continuous streams. Therefore, NoSQL databases have been evolved to cater to the stream-processing solution with the fault-tolerant distributed data ingest systems such as Apache Kafka, and Flume [70]. Examples of stream processing solutions are Apache Storm and Samsa. Also, there are standalone stream processing frameworks that are faster. Additionally, there have been solutions developed to employ OLAP-like processing in the big data landscape. Built on top of data structures, there are currently libraries available for machine learning and big data analytics for real-time analytics processing. For example, there is an Apache Spark framework that contains machine learning libraries and can be used for massive-scale data analytics.

2.3.2 Apache Hadoop and MapReduce

a) Hadoop Framework: Heterogeneity, volume, performance, scaling, cost, and security concerns of big data hinder the process of data analytics at every stage [71]. Apache Hadoop is an open-source framework that renders the distributed storage and analytics of big data. It consists of the core (for storage part) called the Hadoop Distributed File System (HDFS), the processing component that is the MapReduce programming model and resource scheduler called Hadoop YARN (Yet Another Resource Negotiator) [72].

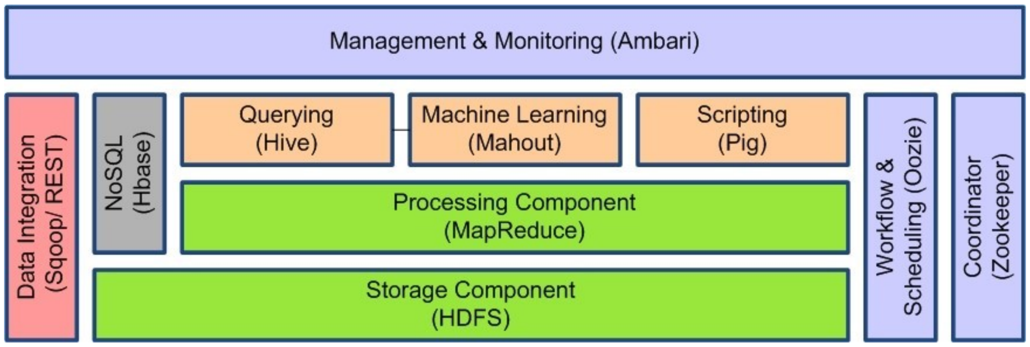

Management & Monitoring (Ambari)
Data Integration (Sqoop/ REST)
NoSQL (Hbase)
Querying (Hive)
Machine Learning (Mahout)
Scripting (Pig)
Processing Component (MapReduce)
Storage Component (HDFS)
Workflow & Scheduling (Oozie)
Coordinator (Zookeeper)

Figure 6 Apache Hadoop Ecosystem

Following is the list of modules in the Apache Hadoop Ecosystem (as shown in Figure 6 [73], [74]):

1) Hadoop core: Hadoop core contains a pre-defined collection of utilities and libraries that can be used by other modules within the Hadoop ecosystem. For instance, if the data access module such as HBase, and Hive needs to access the file storage system in Hadoop, then these are required to build Java Archive (JAR) files stored in the Hadoop core.
2) Hadoop Distributed File System: The default distributed storage system in Apache Hadoop is the HDFS. The huge datasets are dumped in the HDFS and when required, access to the data is provided to other Hadoop modules using utilities [75]. HDFS component provides reliable and quick access to the data by creating several copies of the data block and these copies are distributed across multiple

clusters. HDFS works on the master-slave architecture model and comprises three components namely NameNode, DataNode, and Secondary NameNode [76].

3) Hadoop YARN: YARN is the dynamic resource management component that lets the user run multiple Hadoop applications without having to worry about the aggravating workloads. YARN provides for improved cluster utilization. Key components of YARN are Resource Manager, Application Master, Node Manager, and containers.

4) Hadoop MapReduce: This is a framework for parallel processing of large data set.

b) MapReduce Programming model: MapReduce model is employed for the parallel computation and interpretation of massive-scale data and has three stages: map, shuffle, and reduce [77]. All the jobs are written in a functional programming style to create map and reduce tasks. Dynamic systems for the MapReduce model are commonly clusters that perform tasks such as data partitioning, scheduling of jobs, and communication between the cluster nodes and hence, are more suitable when dealing with massive-scale data. In the map phase, the data are read from the DFS and partitioned into clustered systems where the input is processed to compute the intermediary results which are then stored on the local node of the cluster where the map phase has run and waits for all the map functions to generate output in key-value pairs. The output in key-value pairs is then given as input to the reduce function to generate the final result. The advantage of the MapReduce model is that it takes processing to where the data resides and hence, decreases the transmission of data and improves efficiency. Therefore, the MapReduce model is more apt for the distributed computing of massive-scale data. The summary of the Hadoop module is illustrated in Table 7 [78].

Table 7 Summary of Hadoop module.

Stage	Software	Function
Data Acquisition	Flume	Data acquisition from varied sources to a centralized location
	Sqoop	Data Import & Export between centralized location & Hadoop
Data Storage	HDFS	Distributed File System
	HBase	Non-relational key-value based columnar data store
Computation	MapReduce	A parallel computation programming model
Querying Analysis	Pig	Procedural Data Flow platform
	Hive	SQL-like language for querying
Process Management	Mahout	Machine Learning Library
Querying	Zookeeper	Centralized service to maintain configuration information & synchronization.
	Chukwa	System Monitoring

Hadoop has provided for storing and analyzing data at massive scales. However, data analytics technology cannot be applied to real-time systems [79]. The advent of the Internet-of-Things, smart meters, and devices has led to the possibility of real-time analysis of data for the benefits of business and many other advantages such as smart grid stability, and management. The real-time handling of data falls under one of the categories: Stream processing or Iterative processing. The stream processing framework would work efficiently for big data analytics in the smart grid for real-time decisions about generation, and control.

2.3.3 Apache Storm

It is a scalable and distributed framework for reliable computation and processing of streams of real-time data with processing latencies in the order of milliseconds. Apache Storm can ingest the data from multiple sources using Kafka or Kinesis. A storm cluster is very alike to the data cluster in Apache Hadoop [80]. In Hadoop, MapReduce jobs are executed while topologies are executed in Apache Storm. Topologies are very similar to jobs, but topologies process messages or data forever until these are killed.

In a Storm cluster, there are two types of nodes, namely master node and worker nodes [81]. A background process called Nimbus runs on the master node and this is analogous to Hadoop's job-tracker. Nimbus process distributes the code in the cluster i.e., assigns tasks to the machines and monitors for any failures. On the machines other than the master node, the process called Supervisor runs and it listens for the work assigned to its machine by the Nimbus daemon. It starts and stops the worker node process depending upon the task assigned to the machine. Every worker process runs a subset of topologies. That means the execution of topology requires multiple worker processes that are assigned to different machines across the cluster. It requires coordination between Nimbus daemon and Supervisor processes, and this is taken care of by Zookeeper which is the coordinating service in the distributed environment [82]. Zookeeper takes care of naming, configuration, and synchronization. The important point to note is that all daemons in the Apache Storm are stateless and fail-fast and these come back up even if these are killed by issuing manual commands. This provides for the stable and reliable real-time analysis of big data.

2.3.4 Apache Spark

Apache Spark is an open-source cluster computing framework for analyzing massive-scaled data. It was originally developed by Matei Zaharia at UC Berkeley AMPLab [83]. Spark has the capability for stream processing of big data and has many advantages over Hadoop MapReduce and Storm. In Apache Spark, data analytics is more stream processing than batch processing and hence, it avoids the reprocessing of the data [84]. This provides the stream processing model of Apache Spark to be dynamic and it becomes more crucial during the real-time processing of huge volumes of data collected from different sources. Even for iterative processing, the leading framework currently is Apache Spark as it possesses the capability of processing and holding the data in the memory nodes across the cluster.

1) <u>Characteristics of Apache Spark</u>*:*

- Speed: Spark extends the MapReduce model to support computations of stream processing and interactive querying and is 10 times faster than Hadoop MapReduce model.
- Ease of Use: Applications written in any language Java, python, scala are compatible with Apache Spark.
- Advanced Analytics: Spark supports MapReduce model of Hadoop, SQL-like Querying, streaming data, Machine Learning algorithms, and Graph algorithms too.

- Iterative and Interactive Applications: Spark is designed to execute both in-memory and on-disk. It holds the intermediary results in memory rather than writing on disk to avoid reprocessing the data if required again. Spark operators perform external operations on the data if it does not fit memory.
- In Memory Computation: The data is stored in memory rather than written on disk. Hence, Spark reduces the response time to a great extent when the data is queried.
- Directed Acyclic Graph (DAG): DAG in Apache spark is a set of vertices and edges where the vertices are the representations of the RDDs, and edges represent the operations to be performed on the RDDs. DAGs in spark can contain any number of stages. Even MapReduce model of Hadoop is a DAG of two stages - Map and Reduce. This allows for simple jobs to be completed in one stage and more complex jobs to be completed in one run of many stages unlike multiple jobs in MapReduce model. Thus, jobs in Spark execute faster than they would in the MapReduce model.

2) Spark Framework*:* Other than core Spark, there are multiple components in the Spark ecosystem. These components as shown in Figure 7 [85].

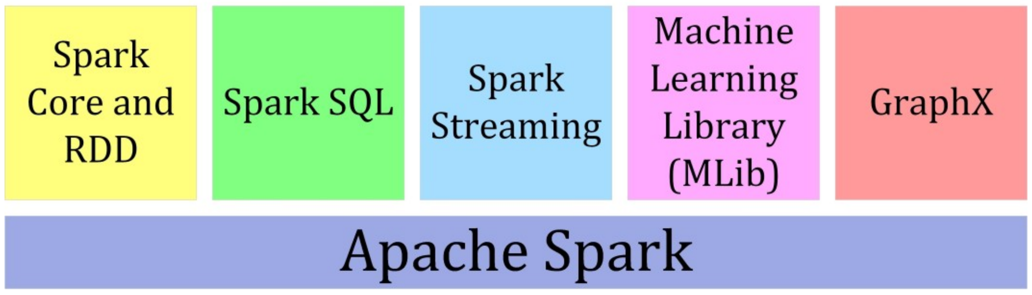


Figure 7 Apache Spark

Spark Core is the base of all the Spark projects, and it allows basic input/output operations, distributed task dispatching, and scheduling through an Application Programming Interface (API) centered on RDD abstraction. RDD is a read-only collection of objects partitioned across a set of machines and it can be rebuilt if any of the partitions are lost [86]. RDDs are fault-tolerant, can be cached in-memory across machines, and can be reused in MapReduce for simultaneous computations.

Spark SQL: Spark SQL is the Apache Spark module that is commonly worked with structured data. It lies on top of Spark core and is used to execute SQL queries. It introduces the schema RDD which can be manipulated. Users can interact with the SQL interface using the command line or over Open Database Connectivity (ODBC), and Java Database Connectivity (JDBC) server.

Spark Streaming: It is the component of the spark that enables the processing of live streams of data (Figure 8). Spark streaming gives a programming interface for processing data streams. It resembles the Spark core's RDD API, pushes data in small chunks, and does RDD transformations on the batches of data.

MLib: Apache Spark comprises a library with common machine learning functionality and this library is called MLib. It processes data faster when compared to Hadoop's disk-based machine learning library called Mahout.

GraphX: The GraphX API provides for users to view data in graphical format and to view RDDs without data movement or duplication. It uses the fundamental operators such as subgraphs, joinVertices, and aggregateMessages. The summary of the proposed module of Spark on top of Hadoop is illustrated in Table 8.

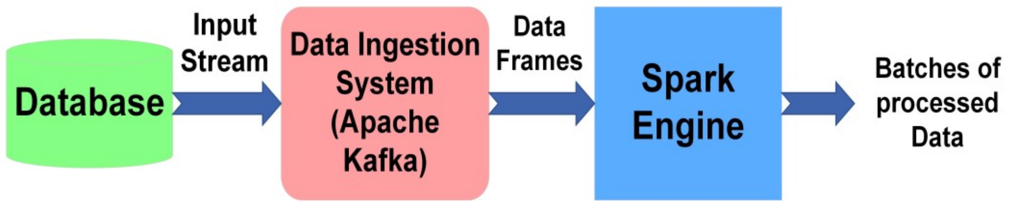

Database
Input Stream
Data Ingestion System (Apache Kafka)
Data Frames
Spark Engine
Batches of processed Data

Figure 8 Spark Streaming

Table 8 Summary of Apache Spark Module.

Stage	**Software**	**Function**
Data Acquisition	Flume	Data Collection from sources to a centralized location
	Sqoop	Data Import and Export between centralized location and HDFS
Data Storage	HDFS	Distributed File System
	HBase	Column-based datastore
Computation	Spark Streaming	Computation Framework
Querying	Spark SQL	SQL-like language for Querying
Analysis	Spark MLib	Machine Learning Libraries
Visualization	Spark GraphX	Visualizations

2.3.5 Apache Drill

It is an open-source software framework that provides for data-driven distributed applications requiring interactive processing of massive-scaled data. Apache Drill is the first and only distributed SQL engine that does not require schemas. Drill automatically understands the data when data are provided. This saves a lot of time and effort in defining schemas, transforming data, and maintaining those schemas. It is designed to handle Petabytes (PBs) of data spread across thousands of clusters and it responds to ad-hoc queries with high performance and low latency.

It is a query layer that functions even when multiple data sources are present. It primarily scans the full tables instead of maintaining indices. The workers in Apache Drill are named Drillbits and run on each of the data nodes in the cluster. The coordination between the drillbits, optimization, scheduling, and execution is performed in a distributed way.

The architecture of Apache Drill contains the following components:

User interface: It provides an interface for the user or application-driven interaction. For example, interface through a command line, Representational state transfer (REST), JDBC, or ODBC.

Processing layer: It comprises SQL Parses, Optimizer, Execution Engine, and Storage Engine.

Data Sources: The data in the pluggable data sources may be spread across thousands of nodes (in-cluster) or they can be local.

The comparison of the different frameworks can be summed up as shown in Table 9.

Table 9 Comparison between different frameworks for big data analytics.

Features	Hadoop	Storm	Spark	Drill
Source Code	Open	Open	Open	Open

Complexity	Simple	Simple	Simple	Complex
Type of Processing	Batch Processing	Real-time Stream Processing	Real-time Stream Processing	Interactive Ad-hoc Querying
Latency	High	Low	Low	Low

2.4 Applied Solutions for Big Data Analytics in Smart Grids

As mentioned before, there are few works that have been reported in the literature for big data analytics specifically in the smart grids. In particular, there are only a few commercial solutions available in the market. One of the earlier practical works on big data analytics was based on the Naive Bayes classification method using the MapReduce paradigm for novel transient power quality assessment [87]. In [88], the authors proposed a cloud-based architecture using Hadoop, Cassandra, and Hive for big data analytics in a smart grid using the data on power usage patterns of customers, historical weather data, supply and demand data.

In [89], Munshi et al. presented an implementation of cloud-based Lambda architecture for smart grid big data analytics using Hadoop data lake. The Lambda architecture is aimed to provide a trade-off between latency throughput and fault tolerance while providing the batch and stream processing capabilities for parallel computation of arbitrary functions on distributed data. The Lambda architecture is based on three layers aptly named as a batch layer, speed layer, and serving layer [80]. The batch layer is required to perform two tasks including the storage of data in a distributed manner and the computation of batch views for the distributed data for low latency. The speed layer utilizes an online technique to store and update the real-time views of the recent data which have not been considered by the batch layer. The serving layer is a specialized distributed database that integrates the data views provided by the batch and speed layers with an aim for real-time and online big data analytics in smart grids. The authors have integrated the capabilities of tools such as Hadoop, Spark SQL, Hive, Impala, and depicted generalized, low latent, scalable, and robust results for smart grid big data analytics.

In [90], several challenges faced at each stage of performing big data analytics are presented. These challenges can be classified into three categories: data acquisition and handling, data processing, and system issues [91], [92].

In data acquisition and handling, the challenges are related to the competent presentation of heterogeneous data to reflect the diversity, hierarchy, and granularity of data. Also, the raw datasets often contain redundancy that needs to be reduced along with data compression without deteriorating the information in the data. Data life cycle management is of utmost importance because of the availability of huge amounts of data and the current storage systems cannot store the massive data available at an unprecedented rate. Therefore, there needs to be a practical system where the data is analyzed on the go and for that, the stream processing framework using HDFS, and Apache Spark has been proposed in this research work. The system challenges for analytics are faced with massive storage and high-speed processing. Furthermore, there are concerns about privacy and security since the data might contain personal information.

In data analytics, the challenges posed are that of huge data and the requirement of real-time processing. One of the solutions to these challenges could be approximate analytics providing approximate but real-time results. Mining on social media and customer feedback could present challenges as the data is generally unstructured

Solving these challenges requires the use of large-scale parallel systems that further brings additional challenges such as energy management, scalability, and real-time collaboration. The energy usage of the large-scale parallel systems has been alarming due to massive data volume and analytics demand. Hence, system-wide energy management techniques should be utilized in big data system solutions.

In the smart grid discipline, a cloud-based platform project has been presented in [22] where the University of South California microgrid was deployed as a testbed to transform the electrical utility into a smart grid in the future.

The challenges and solutions to handle big data from smart grid units have been researched in academics and industrial centers. Solutions have also been implemented at the commercial level by a few utility companies. These utilities always strive to meet the goals of moving to a smarter grid to support distributed generation, distribution automation devices, providing new products and services, improving operational efficiency, and finally enhancing the system reliability. Some of the prominent industry efforts are described in the following:

2.4.1 Accenture Solution

Accenture proposed a system that uses grid observability to drive performance (Figure 9) and to govern five distinct smart grid data classes such as operational data, non-operational data, event message data, meter usage data, and metadata [93]. All the classes of data should be treated and managed differently owing to their inherent characteristics and different sources. The architecture was aimed to overcome the challenges of corresponding the data collection infrastructure to the desired outcome, application of tools to manage massive-scaled data, and analysis of master data to benefit from smart grid potential. The commercial solution is proposed to discover the information through the components as shown in Figure 9.

The provided solution explains the analytical aspects of the proposed architecture; however, it does not provide detailed information on the data treatment, management, and storage processes.

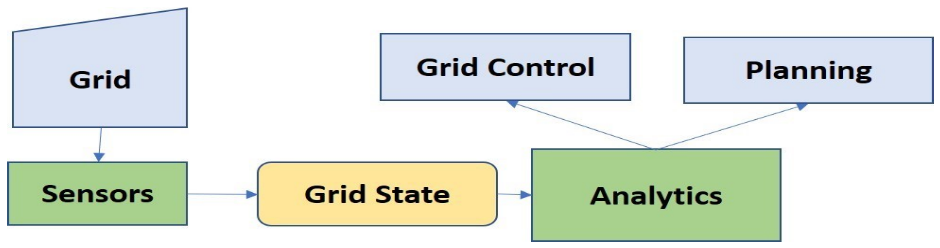


Figure 9 Using observability to ensure performance

The Accenture architecture named Intelligent Network Data Enterprise (INDE) has the following components:

- The software layer in the architecture acts as a layer between the grid data sources and the current utility enterprise IT platforms. It aims to integrate the data from various sources to enhance the utility business operations and customer operations.

- The integration layer is prevalent to provide a unifying platform to the smart grid ecosystem products such as smart meters, communication lines, sensors, and other electrical network components.

- The visualization layer is provided to observe and monitor the different components in the grid. It also aims to recognize patterns in the raw data to correlate with different events and metadata

The implemented solutions by Accenture at their clients' sites indicate their emphasis on the following five major application areas for smart meter data [94]:

- Enhancing outage management: The main goal of smart meter analytics has always been to enhance outage management. Outage management can be enhanced if the disturbances in the electric network are accurately predicted, localized, and restored by integrating the outage notifications, sectionalizing, and reclosing systems.
- Power quality assessment: The smart meter data can be used to monitor the quality of power at every point in the electrical distribution network. The fluctuations in the frequency and voltage can cause damage or failure to the electric equipment. The remote assessment of power quality can help utilities to investigate legitimate claims of customers saving field effort and time.
- Protect customers and detect losses: The system should protect the interests of all customers by detecting different losses including technical and nontechnical losses in the electrical network. The nontechnical losses occur when the customers tamper with electrical meter readings to reduce their bills. The integration of data from feeder meters and smart meters into the work management system will help utilities to identify electricity theft and investigate claims easily.

- Renewable energy forecasting: The generation of renewable energy is increasing in the grids. This calls for the proper management on part of the grid operator. Renewable energy is less predictable. However, accurate forecasting should be in place by integrating the data from smart meters and weather stations. This would help in the operational and investment decisions of utilities. With accurate forecasting, the grid operators can stabilize the supply and quality of power throughout the electrical distribution network.

Future market developments: Long-term planning is required for balancing generation and load demand, and flexible energy tariff planning.

2.4.2 IBERDROLA

Big Data techniques are used to yield knowledge management solutions to control high turnover environments and to minimize the impact on call centres. Iberdrola has a part of its ambitious Digital Transformation Program [95] in the use of big data techniques. The company group targets to invest 4.8 billion euros in the digital transformation between 2019 and 2022 to boost the performance and conservation of its assets using data analytics and artificial intelligence. Digital analytics provides for creating an analytical environment to inspire knowledge that aids to maintain the three lines of business: Networks, Renewables, and Customers. Some examples of these applications are:

- Detection of non-technical losses and design of optimal time-of-use tariffs with the use of customer load curves to improve energy utilization [8]. The company has installed more than 11 million smart meters in Spain, generating 240 million registers every day. Big data techniques over an estimated volume of 90 billion registers per year are being used to improve revenue collection and to optimize energy use.

- Improvement of the operation and maintenance of the utility's assets expanding the availability of its generation plants. For example, in the U.S., Iberdrola is saving $3 million monthly by feeding wind turbine power generation data across multiple wind farms to develop curtailment optimization plans [96]. Iberdrola is leading a five-year project called Romeo, 16 million EU Horizon 2020 project, aiming at the reduction of the preservation cost of wind turbines using predictive machine learning algorithms, artificial intelligence, and cloud computing. Utility's relationship with the customers can be transformed by the development of applications such as managing electricity consumption from mobile phones or scheduling electric vehicle charging [95]. Big Data techniques are also used to provide knowledge management solutions to command high turnover environments and prune the impact on call centers.

2.4.3 ITRON-TERADATA Solution

Itron-Teradata architecture is established on active smart grid analytics (ASA) as depicted in Figure 10. As per the solution, the data warehouse actively provides strategies for the parallel ingestion of massive-scale data from varied sources and executes complex analytics for applications such as energy diversion detection, power quality, demand response, transformer load management, load forecast, and customer profiling. The data arrive triggering actions and activating workflows [97]–[99].

ASA is based upon the comprehensive Utility Logical Data Model (ULDM) of smart grids' data. The ASA solution helps the customers through self-service with insight on how to convert their usage to green energy, and to make savings in energy and billing. The solution assists the utilities to develop communication channels for customer-utility interaction and to invest in assets that boost customer experience. Also, the regulatory agencies benefit from the ASA solution with insights on the efficiency standards of operations, the percentage of energy from alternative sources, and the fair pricing of energy.

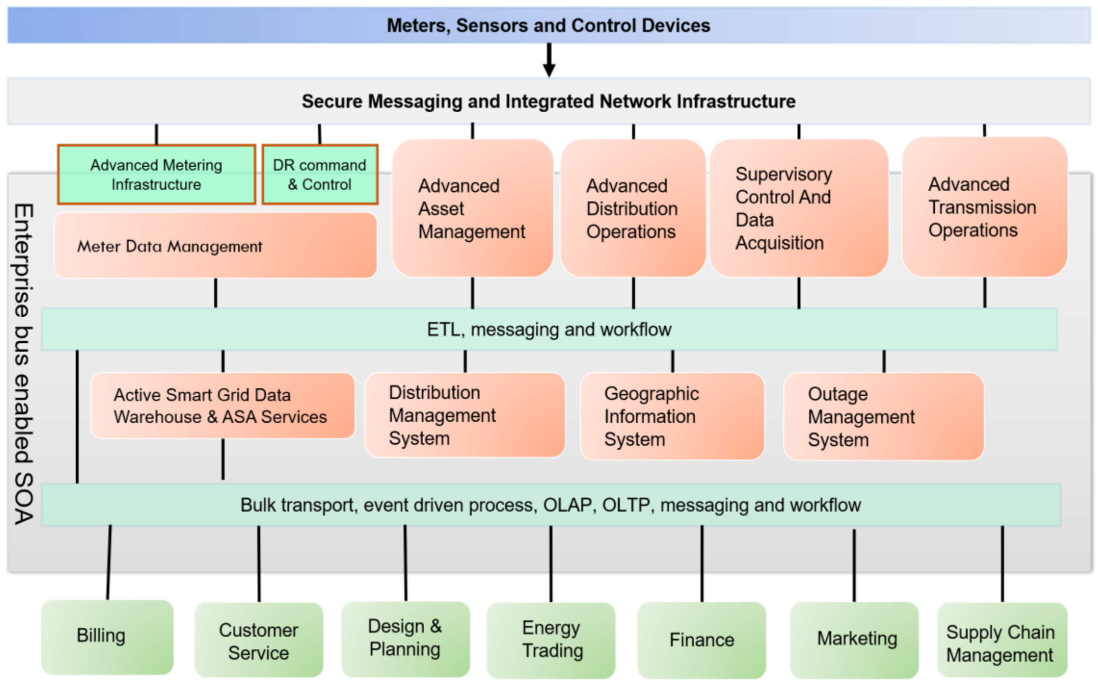

Meters, Sensors and Control Devices
Secure Messaging and Integrated Network Infrastructure
Enterprise bus enabled SOA
Advanced Metering Infrastructure
DR command & Control
Meter Data Management
Advanced Asset Management
Advanced Distribution Operations
Supervisory Control And Data Acquisition
Advanced Transmission Operations
ETL, messaging and workflow
Active Smart Grid Data Warehouse & ASA Services
Distribution Management System
Geographic Information System
Outage Management System
Bulk transport, event driven process, OLAP, OLTP, messaging and workflow
Billing
Customer Service
Design & Planning
Energy Trading
Finance
Marketing
Supply Chain Management

Figure 10 Service-oriented architecture of Itron-Teradata solution

2.4.4 International Business Machines (IBM) Solution

Since 2013, IBM has worked on the smart metering infrastructure on the private cloud for E.ON with an aim to enhance the deployment and management of smart meters and to help incorporate renewable energy sources easily into the current grid [100]. The platform addresses the challenges of high data storage, low speed of report generation and analytics. With the platform, customers have better control of the energy usage with information on their usage profile, on electricity tariff for the time of use, and on changes in consumption patterns when compared to their historical data. IBM intends the platform to be scalable with low start-up and operational costs in order to provide for future growth. The platform has a high emphasis on ensuring the privacy of sensitive customer data, however, the data would be retained for a longer time to help with the emerging regulatory requirements in the future.

2.4.5 USA EXELON

Since 2014, Baltimore Gas and Electric (BGE) and Exelon have been working on a project by employing C3's cloudbased data processing platform to control the working of millions of smart meters installed in the regions of Chicago and Philadelphia Electric Company (PECO) utilities [101]. They have been successfully tapping the data from the smart meters with an aim to locate and avert energy theft. They employ machine learning algorithms to encode every rule of meter tampering and unbilled power delivery as these change over time. The algorithms also integrate various types of data from systems in place for the management of data from meters, outage prevention, user profiling, billing, and asset management. These applications led to the program of Business Intelligence

Data Analytics (BIDA) and the solution of Data Analytics Platform (DAP). The solution supports the domains of business support, customer service, smart energy services, grid management, and AMI with a vision to assist future utilities, energy regulators, and customers.

2.4.6 Korea Electrical Power Corporation (KEPC) Solution

KEPCO launched two projects to use big data analytics on smart grids' data to improve demand management, and load forecasting and has been achieving considerable success in its goals ever since [102]. The first project helps customers to save electricity by comparing similar customers energy consumption data and allows KEPCO to prevent brownouts and manage load demand. The second project involves analyzing the business risks of blackouts, user complaints, weather changes, climate change statistics with the aid of social networking data, internet data, and complaints.

The companies do not explicitly describe their commercial solutions and do not release the information of the components of data management architecture in detail. However, noticing the potential of big data analytics to manage the demand-side response and user service, the utilities have now and again been cooperating with IT companies to tap the potential. This work has additionally presented the proposed architecture aiming for the streams data processing to provide real-time information and visual analytics.

2.5 Applications of Big Data Analytics in Smart Grid

This section discusses a few of the potential application areas which would avail from the big data analytics in the smart grid. It also details the previous application-based works and their proposed methodologies.

2.5.1 Fault Classification and Identification

The invention of the smart grid was driven by the need for clean and alternative forms of energy. The utilization of distributed energy sources in distribution grids brings the integration of renewable energy sources to reality. The microgrids allow for energy generation closer to load and hence, assist the improvement of power delivery and reduction in the power transmission losses. Furthermore, the microgrids can be used in islanded mode, and consequently, the loads can be protected from the damages resulting due to fluctuations in voltage and frequency [103].

The fluctuations of the energy produced by renewable energy sources bring uncertainty in the energy generation from distribution grids. Usually, Inverted Integrated Distribution Grids (IIDG) are used to improve the power quality in microgrids. However, these IIDGs have low inertia and hence if the faults caused in microgrids are not detected and cleared in short times, this is a huge threat to the microgrids. The classical approaches to fault identification and clearing [104] are based on the measurement of overcurrent and negative sequences of current. These approaches are not suited to microgrids due to their low current capacity. The statistical features are extracted using the wavelet transforms on the current measurements in the branches sampled by protective relays. The deep learning model is developed with the training data available on the statistical features to detect faults, classify them, and localize the faults in [105].

2.5.2 Preventive Maintenance

The pieces of equipment of the power grid are vulnerable to failures and a robust plan for preventive maintenance of equipment, and devices in the power grid can play a crucial part in reducing the probability of occurrence of failures in the power grid. Preventive maintenance can signal for and provide maintenance for equipment before this fail and hence, will avert major events and disruption of power supply for long periods. The integration of renewable energy sources at the distribution level of grids through microgrids supply clean energy. Nevertheless, the uncertainty of supply and fluctuations of frequency and voltage increase vulnerability to failure. It is required that the occurrence of failures is detected before failure and the clearance time is averted using preventive maintenance. Preventive maintenance is categorized into two types - time-based and condition-based. In time-based maintenance, the components are subjected to maintenance at periodic intervals of time irrespective of their condition. This approach does not utilize the service life of the components efficiently. Condition-based maintenance monitors the health of the components and draws a correlation between the current status and future faults of the components so that the future maintenance plans are scheduled [106]. One of the approaches to prognostic maintenance is the design of a proposed integrated fault detection system developed after analysis of the data from SCADA and Pole Mounted Auto Reclosers (PMARs) [107]. PMAR is a breaker that trips for intermittent fault currents and closes automatically to supply the power after a short duration of time nonetheless, it stays open for a permanent fault.

A reinforcement learning-based framework is proposed in [108]. The framework monitors the health of the equipment, models the degradation, and computes the remaining useful life of the grid components. The framework tested on a case study on the power grid performs with good approximation capability by using an ANN ensemble model. All of the data or subset of data from grid operations data, weather information, diagnostics data of the relay protection systems, galloping of power lines, fault tolerance current, and voltage signals have been used for the design of data-driven models for preventive maintenance in the power grids. Different machine learning models such as SVM [109], extreme learning machines [[110], Long Short Term Memory (LSTM) [111], hybrid ensemble models [112] are used to build data-driven models. The correlation between the actual faults that have occurred in the past and the features extracted from the data has been studied. These analyses models and studies are required to have high learning without iterative computations to converge faster, predict with higher accuracy and earliness. This would be an ideal solution for big data analysis for predictive maintenance.

2.5.3 Transient Stability Analysis

Transient stability analysis (TSA) is performed to study the safe operation of the power grid. However, the challenges to the TSA these days are the integration of intermittent renewable energy sources at the distribution level, fluctuating demand of load, and deregulated energy market. The efficient approaches that extract information and patterns in the highly redundant records of big data are required for TSA. The techniques for TSA can be classified into automatic learning approaches, direct techniques, and time-domain techniques. Automatic learning approaches have edge over direct and time-domain techniques for real-world applications. The direct techniques [113] have demerits in the construction of energy functions for large-scale power systems whereas time-domain techniques are computationally inefficient for real-time applications [114].

Steady-state variables are used as features for TSA in [115] thus avoiding the use of time-domain simulation. The approach takes into account the size of the electrical network, the topology, the location of a fault, and operating status.

In [116], Yu et al. used time-series synchrophasor measurement data under different simulation contingency models to train the deep learning model of LSTM for online-assessment of transient stability status post-contingencies. Although the training of the TSA LSTM model was computationally expensive and time-consuming, the time adaptive nature and self-learning of temporal dependency by the LSTM model achieve better test accuracy and highly responsive time. Moreover, to reduce the training time, simpler models such as Extreme Learning Machines (ELMs) that are single-layer neural networks are used [117]. To address the uneven class distribution of power systems' data with a higher number of data points representing stability and a lower number of data points representing contingency condition, Baltas et al. proposed a response-based ensemble model of diverse ELM [118].

Rahmatian et al. worked on the implementation of transient stability assessment in real-time using characteristic features of voltage and current phasors from PMU data, Classification and Regression Trees (CART), and Multiregression Adaptive Regression Splines (MARS) models [119]. The models predict if a situation is stable or unstable using CART and applies MARS along with online TSA to indicate the level of severity of a contingency and instability of the system with high accuracy.

2.5.4 Health Monitoring

Failure in crucial components of the power grid such as transformers will lead to brownouts or blackouts in the electrical grid network. It is crucial that the health of the electrical components in the grid is monitored. Classically, the monitoring system is based on a threshold mechanism that monitors different parameters and readings for different grid components.

The uncertainty and intermittent nature of renewable energy sources at the distribution level bring uncertainty in the life estimation of crucial components such as power transformers. In [120], Aizpurua et al. proposed a probabilistic health monitoring framework for power transformers by using a probabilistic forecasting approach along with Monte Carlo-based Kalman-filtering techniques. The lifetime estimation of transformers in these models is adaptive as the dynamics of smart grids is propagated to the power transformers to determine the probabilistic thermal model and lifetime model.

There are different artificial intelligence-based approaches used for health monitoring using big data in smart grids. These include artificial neural networks [121], deep learning models [122], expert systems [123], fuzzy logic [124], [125], and genetic algorithm [126].

Mileta et al. analyzed the Mamdani model and Sugenomodel in the fuzzy expert system to compute the probability of occurrence of faults in the future and to determine the urgency of intervention or maintenance on the transformers based on their current condition [123]. The models utilized the online and offline data on historical and current conditions of transformers' age, lower oil level, frequency response analysis, oil temperature, insulation temperature, insulation degradation, and polarization index.

Hybrid models are utilized to overcome the shortcomings of single models. For instance, a health monitoring system was developed by Allen et al. for the health diagnostics of building automation systems and variable air valve units using a fuzzy logic model [127]. The fuzzy logic model detected anomalies in the operating conditions and generated fault signatures. The neural network-based model was used to classify the fault signatures into different faults. The monitoring of the health of the components at lower granularity ensures that the energy consumption observed at higher levels is reduced and finally helps for energy savings, and efficient monitoring.

2.5.5 Power Quality Monitoring

When the frequency, magnitude, and waveforms of current and voltage are steady and within the prescribed limits, it is defined as power with high quality. Power quality also defines the performance and health of the smart grid components and the accuracy of utility metering. With the integration of non-linear sources of energy and power electronics-based devices, the harmonics appear in the voltage or current waves and it is essential that the power quality of the supply is maintained for the health of the devices, sensors, and appliances connected to the electric network. The power quality issues are currently addressed using dynamic voltage regulator, inverter, power quality monitoring, static synchronous compensator, and unified power quality conditioner.

Power quality monitoring is performed using conventional approaches through the integration of SCADA, AMI, or by using artificial intelligence-based approaches. Multiple machine learning modeling such as Support Vector Machines [128], decision trees [129], Bayesian networks [130], kNearest neighbors [131] have been employed for monitoring power quality disturbances.

Wang et al. employed deep learning in each of the stages of power quality classification i.e. signal analysis, feature selection, and classification [38]. They used a deep convolutional neural network consisting of the 1-D convolutional layer along with pooling and batch normalization layers for the automatic extraction of features from disturbance samples. They presented evidence in terms of accuracy and training time cost that deep Convolutional Neural Networks (CNNs) performs better for applications of automatic power quality classification when compared to other deep learning models such as gated recurrent networks, long short-term memory, ResNet50, and stacked auto-encoders. To overcome the non-distributed computing and feature extraction-based power quality classification, Chen et al. presented an integrated solution based on deep belief networks for real-time and distributed power quality disturbance analysis [132]. The developed models proved to have higher accuracy and more robustness on distributed platforms, however, the training time is also very high.

2.5.6 Topology Identification

The topology identification problem in the smart grid includes the identification of the structure of power distribution network, identification of customer phase connectivity, and associating a customer with a transformer at the distribution level. The identification of phase connectivity is crucial to the analysis of distribution system including distributed network estimation, power flow analysis, optimal power flow, distribution network reconfiguration and restoration, and load balancing. Topology identification could be possible using specialized sensors such as micro-synchrophasors, and phase meters. However, using a special sensor for each customer is impractical and expensive. There are many approaches developed to identify topology using the data made available by the current infrastructure such as AMI, SCADA, GIS, Outage Management System (OMS), and besides machine learning approaches have been developed using training data on field validated phase connectivity.

Voltage time series data have been utilized to extract feature vectors after the application of principal component analysis and the authors have suggested that the voltage data are predictive of the phase connectivity [133]. Afterward, the k-means clustering approach has been applied to cluster the different customers into the three different phases for phase connectivity identification. The innovative model was tested on a real distribution feeder and the test accuracy was about 90 %.

2.5.7 Energy Theft

Energy theft is defined as the act of changing the electricity consumption reading in order to reduce the bill through physical approaches such as bypassing the smart meter, tampering with meters, cyber approaches such as hacking into a smart meter to change the energy consumption values. Data-driven approaches are currently applied to identify energy theft and these approaches are classified into different types depending on the type of available data. When the smart meter data were not available, machine learning models such as fuzzy clustering [134], and SVM [135] have been applied to the annual energy consumption, and credit scores were determined to identify theft detection. When the smart meter data and theft cases data are available, then supervised machine learning models such as neural networks [136], deep learning models [137] can be applied. Usually, energy theft cases are not available or disclosed for research. In such cases, the energy theft identification can be performed using smart meter data, and network topology information can be determined using state-estimation based approaches [138], and other anomaly detection techniques [139].

2.5.8 Renewable Energy Forecasting

The renewable energy (RE) sources are environment friendly, clean, and unlimited replenishable sources of energy. Nevertheless, the uncertain and intermittent behavior of the supply poses many challenges in the generation of power using renewable energy sources. The reliable and accurate RE forecasting helps in the grid operations, load management, planning of capacity, scheduling of generation, and regulation of energy. Multiple approaches including physical models, statistical models, machine learning approach, and hybrid models have been used to date for renewable energy forecasting.

Physical models include the simulation of geographic characteristics of an area. These models utilize weather forecasting, geographical information, and meteorological information. Physical methods require huge computational resources, are less accurate and also, are not suitable for short-term forecasting. Statistical models apply mathematical modeling to recognize the patterns in time-series data of renewable energy sources. The methods such as Auto Regressive Moving Average [140], Kalman Filters [141], and Markov models [142] have been applied previously. With the widespread popularity of machine learning models, these have been applied reliably on renewable energy forecasting. The machine learning algorithms include models such as linear regression

[143], decision trees regression [144], multi-layer perceptrons [145], support vector machines [146]. Owing to the inherent intermittent and non-linear nature of renewable energy supply, deep learning models have been found to be extremely efficient and effective [147]–[149]. Deep learning models such as deep belief networks, autoencoders, convolutional neural networks, long short-term memory, and deep learning ensemble models. have been applied to predict renewable energy from sources. The patterns of temporal changes in renewable energy are captured in the parameters of the deep layers. The high accuracy of renewable energy forecast will help in the planning, and development of reliable, and resilient integration of the sources in the distribution grids through microgrids.

It has been observed that there is a need for a distributed computing and big data analytics platform that utilizes different types of technologies for real-time solutions in smart grids and also a middle-ware software is required to integrate all of the technologies with reliability and stability. Enterprises dealing with big data are required to address the challenges of security, privacy, and data handling. Before any big data techniques are employed in the smart grid, it is always necessary to consider steps such as data acquisition, data management, analytics, and visualization along with the requirement of the real-time processing of data.

2.6 State-of-the-art Short Term Load Forecasting

Short-term load forecasting in smart grid is currently of much research interest owing to the integration of variable energy resources at the distribution level and the stochastic nature of energy consumption behavior. The volatile load demand patterns at the consumer level, if predicted with high accuracy, helps in load balancing and renewable energy efficient utilization. Initial works on short-term load forecasting in smart grid included time-series analyses and traditional statistical approaches. With the research interest shifting to artificial intelligence, various machine learning approaches and deep learning techniques have been utilized to forecast the energy consumption at the household level. In [150], Wang et al. proposed a two-stage forecasting methodology. In the first stage, the traditional time forecasting models were utilized to perform a day ahead load forecasts. To enhance the accuracy of the forecasts, the second stage utilized models such as support vector machines (SVM), linear regression, and quadratic models to generate predictions of deviations. These deviations were integrated with the forecasts from the first stage to yield the overall forecast values with an average Mean Absolute Percentage Error (MAPE) of 5.21%. However, the SVM model is not suitable for big data as the training time for the SVM model scales super linearly with the increase in data records.

In [110], the authors proposed wavelet pre-processing, improved wavelet neural networks, and generalized extreme learning machines (ELMs) on training data. The predictions of the load were provided as intervals considering the uncertainties of the forecasting models and data noise. ELMs are neural networks with a single hidden layer. The usual disadvantages of ELMs are that the forecasting accuracy is heavily dependent on the activation function, and the generalization is poor. These shortcomings were effectively tackled by the introduction of wavelets as the activation functions in their methodology. However, the ELM-based methods do not effectively perform deep extraction of inherent information and features associated with energy consumption data owing to their single layer-based modeling.

In [151], the authors established mathematical models of backpropagation neural networks and Elman neural network. These models were used with small learning rates, and layers to store internal states, and to deal with time-varying characteristics of energy consumption data. Their results concluded that Elman neural networks perform better in dynamic load forecasting than backpropagation neural networks. However, these neural network-based models are bound to converge to local minima rather than global minima. This leads to poor generalization and further, causes overfitting.

Recently, a lot of research attention has been focused on the development of deep learning models to recognize patterns in the energy consumption data and to perform the forecasts with high accuracy and efficiency. Typically, deep learning models suffer from the problem of exploding gradients (i.e., learning diverges) or vanishing gradients (i.e., the learning stops). This problem is taken care of by LSTM networks that introduce memory cells and computing gates. LSTMs are types of Recurrent Neural Networks (RNNs) that have been utilized in the past for time-series analyses and load forecasting problems. In our recent works, multiple efficient and accurate energy consumption forecasting models were developed based on ensemble models, extreme learning machines, LSTMs, deep neural networks, and dimensionality reduction techniques [152]–[154].

In [155], the authors developed hybrid sequential learning based on the deep learning model. Their solution utilizes Convolution Neural Network (CNN) in the first phase to extract the features from the energy consumption dataset and uses Gated Recurrent Unit (GRU) in the second phase to utilize its effective gated structure to make predictions. However, GRU-based models do not have as great volatility as LSTM-based models owing to their simplicity and a smaller number of gates for the gradient flow.

In [156], the authors proposed an advanced domain fusion methodology based on CNN, which derived the time-domain and frequency-domain features representing the changing energy consumption trends, LSTM layers, and Discrete Wavelet Transforms (DWT). The authors reported a MAPE of around 1% on two datasets, which comprise energy consumption (MW) information at aggregated levels.

In [157], Kong et al. proposed an LSTM memory-based framework for short-term energy forecasting at the residential level. They incorporated the appliances energy data from a Canadian household to illustrate the efficacy of their deep learning framework. Although minutely data were available, an aggregation of thirty minutes has been utilized in their work. However, only six appliances' energy data were utilized in the study. Their results were compared against the benchmarking models of Feed Forward Neural Networks (FFNN) and k-Nearest Neighbors (k-NN). The superior performance of the LSTM-based model, with a MAPE of 21.99% was displayed.

The gist is that although numerous methodologies have been researched and incorporated, there are significant limitations such as lack of scalability to big data, as well as limitations to offline analyses rather than real-time analysis. Hence, it is essential to propose a framework with an aim to overcome the existing limitations.

CHAPTER III

APPROACHES AND METHODS OF LEARNING FROM DATA*[4]

In this chapter, different clustering and aggregation solutions are discussed for load forecasting. That is, the clustering algorithms including the k-Means algorithm and the k-Medoids algorithm are proposed. Furthermore, works on transfer learning and incremental learning are presented. The transfer layer enables the faster convergence during training of deep learning models, and the incremental layer enables to update the trained deep learning models with arriving data points, thereby enhancing the forecasting accuracy of load forecasting models. These algorithms will be used in Chapter V in the proposed research in the clustering layer, transfer layer and the incremental layer of the hybrid multi-stage framework.

3.1 Unsupervised Machine Learning – Clustering Algorithm

3.1.1 Overview

The rising number of installed smart meters allows for the collection of data corresponding to consumers' end devices. The smart meter data, representing the customer energy consumption behavior at the granularity of the household level, enable the electrical utilities to perform capacity planning, capacity building, and operations. The integration of the smart meters' capability with the communication infrastructure in smart grids enhances the protection, reliability, efficiency, and safety of the energy supply to the consumers. The collected data have been aggregated to different levels to perform load

[4] Reprinted with permission from "Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid with Clustering and Consumption Pattern Recognition." by Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Shady S. Refaat, Mahdi Houchati, Othmane Bouhali, and Santiago Bañales, 2021. IEEE Access 9, 54992-55008, Copyright 2021 by Dabeeruddin Syed.

forecasting. For aggregated feeder level forecasting, the bottom-up approach is usually implemented. That is, the household level consumption data are aggregated to the feeder level and then the training is performed with the aggregated data. Similarly, the data at the feeder level can be aggregated to the level of the distribution transformers, while several distribution transformers could be aggregated to the level of substation and so on which helps in performing load forecasting at the needed level. The electric utilities rely on short-term forecasts at the distribution feeder and transformer level to support peak planning and grid operation. Load forecasting enables the electric energy utilities to plan, identify the regions with high load demand, match the volatile energy demand by changing the generation capacity, reduce generation cost, regulate energy prices, and manage scheduling.

The energy consumption varies from one location to another owing to different weather and climate conditions. And for the same reason, the energy demand varies on different days of the week and at different times of the day. Many researchers have been interested in grouping the different conditions or different locations based on the similarities between the available features of the data with an objective to reduce the number of forecasting models required for predictions [158], [159]. The clustering techniques intrigue researchers to improve the load forecasting methodology and to enhance accuracy.

Reference [159] proposed a day-ahead forecasting algorithm that uses load fluctuations and feature importance to cluster different customers at the distribution level. Crow search algorithm was utilized to determine the initialization conditions to avoid local minima convergence in the K-means clustering method and finally, an ensemble random forest model was generated to realize the day-ahead forecasting. The authors reported the lowest Mean Absolute Percentage Error (MAPE) of 1.633% for the random forest model and showed that the model performs better compared to Extreme Learning Machine (ELM), Neural Networks (NN), and Support Vector Machines (SVM). Their methodology benefited from the clustering of the 24 hours of a day into different clusters based on the fluctuation of energy consumption. Although the employed clustering method solves the issues of criteria for selection and initialization in the k-means algorithm, there is a scope of improvement in the crow search based k-means clustering algorithm when faced with high multi-modal peaks in the data formulations.

In [160], the authors proposed a long-term energy forecasting methodology that utilizes the spatial clustering algorithm of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to predict year-ahead load values for power system planning. The density-based clustering technique benefits from its inherent ability to effectively dealing with the noise in the data by eliminating the outliers. Similar sub-zones are clustered using DBSCAN based on the features of historical yearly energy consumption profiles, land use types, and geographic information. Eventually, Non-Linear Auto Regressive (NAR) neural network models yield the values of the predicted load. They reported that their proposed model works better when compared to existing models such as exponential smoothing, grey theory, and Linear Regression (LR). However, the short-term load forecasting is not addressed using this methodology. In [161], the author proposed a hybrid model based on a Kalman filter, an artificial neural network, and wavelet transforms. The hybrid model also used clustering techniques for short-term load and renewable energy forecasting. The work provided evidence that the hybrid models involving clustering-based wavelet and artificial neural networks perform better than conventional models and other hybrid model combinations. However, in this work, the clustering was based on geographical zones, rather than the actual patterns of energy consumption.

Empirical Wavelet transformations (EWT) have been used to decompose the load data into Intrinsic Mode Functions (IMF) [162]. Along with LSTM modeling, the IMF functions are used to predict the low and medium frequency components for load predictions. Furthermore, the high-frequency components are highly varying components with uncertain characteristics, and these are clustered using Improved-DBSCAN (IDBSCAN) algorithm. The prediction results of the high, medium and low-frequency components are aggregated to determine the total load predictions for short-term load forecasting. Their methodology has the advantage of employing different prediction methods according to the characteristics and the variance of data. However, the methodology based on IDBSCAN is not effective if the data is scaled to higher dimensions. Also, it is efficient only when the different clusters have varying densities.
Autoregressive Integrated Moving Average (ARIMA) model has been utilized as a baseline method for predicting energy consumption as it is easy to implement and generalize to a wide variety of specifications [163]. Nepal et al. used k-means clustering along with ARIMA modeling for predictions of energy consumption in buildings [163]. The clustering technique is used to cluster the days with similar load characteristics during

the hours of a day. In their work, the days of a year were clustered into 6 clusters. During the prediction phase, their methodology determines the cluster number of the days preceding the testing day and finally predicts the energy consumption of the testing day. The results indicate that the standalone ARIMA model can be improved with the addition of clustering-stage. However, the k-means clustering utilized is sensitive to outliers if present in the data.

Fuzzy c-Shape clustering has been investigated by Fateme et al. to cluster the load data depending on the shape of energy consumption [164]. A horizontal ensemble model consisting of LSTM and XGBoost has been used to perform a day-ahead forecast of 30-minute granular load prediction. A novel feature of apparent temperature is used in their analysis. The apparent temperature is the equivalent weather variable as experienced by humans due to the collective influence of humidity, temperature, water vapor pressure, and wind speed values. They have suggested that the addition of novel features, such as the representative feature of weather, will improve the accuracy of predictions from cluster-based ensemble models. However, their methodology is dependent on the empirically assumed function and formulation of equivalent apparent temperature.

LSTM models have been of interest to many researchers to perform energy forecasting. In another work, an ensemble of LSTM was used to perform short-term energy forecasting [165]. The different branches of the ensemble utilize different clustering algorithms in their initial phases. The employed clustering algorithms include Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), DBSCAN, and KMeans++. In the final phase, a fully connected neural network is employed to aggregate the results from the different branches of the ensemble. The ensemble and deep learning models have been tested to yield better results when compared to nonensemble and classical models. In [154], Syed et al. proposed an averaging ensemble model of the classical algorithm including LR and deep learning algorithms including LSTM and DNNs. The results indicate that the averaging ensemble model overcomes the shortcomings of the individual models and provides synergy to enhance the overall accuracy. However, the ensemble models and the LSTMs are computationally expensive. In [166], a novel fuzzy-based clustering method is employed to cluster data into different clusters using the order of feature importance. The clustered data goes through two different phases of regression. In the first phase, a Radial Basis Function Neural Network (RBFNN) is employed. In the second phase, the output of the first phase is passed to a pooling layer followed by a convolutional layer and finally, through two fully connected neural networks. They tested their proposed method with two case studies to predict the hourly energy consumption for the next seven days with better results as compared to the classical energy forecasting methodologies. However, the clustering method utilizes common space where data are shared between neighboring clusters and this introduces redundancy and requires additional computations.

Clustering has also been applied at the household level. In [167], Bayesian non-parametric clustering model has been applied to cluster the households with similar energy consumption profiles across seasons and neighborhoods. The load profile curves are obtained after the removal of phase variability with the application of elastic shape analysis. The household-level energy consumption has high variability and with the predictions on household-level load, it is difficult to aggregate the prediction results to high levels for the use of optimized utility operations.

In summary, there has been a significant research effort in the application of clustering techniques at different levels of energy distribution networks. The metric for clustering has been a similarity in weather conditions, seasons, days of the week, and hours of the day. However, it is required that the metric for clustering should emphasize the patterns of energy consumption.

3.1.2 k-Means Clustering Algorithm

The objective of the K-means clustering algorithm [168] is to reduce the Error Sum of the Squares (SSE) scoring function that is given by

$$SSE = \sum_{i=1}^{k} \sum_{x_p \in C_i} \left|\left|x_p - \mu_i\right|\right|^2, \tag{1}$$

where k represents total number of clusters, C_i represents each cluster, x_p represents each point in a cluster, and μ_i is mean of all points in a cluster.

K-means applies an iterative greedy approach to reduce the sum of squares error until it reaches a local optimum. K-means starts with the selection of the number of clusters k and the initial k number of centroids assigned to each cluster. This step is followed by the centroid update. At this stage, all the points are assigned to the clusters with the nearest centroids. Once all the points are assigned, the centroids are updated for each cluster as the mean values of all the points in the clusters. The cluster assignment and the centroid update are repeated until there is no change in the centroids in two subsequent loops. This indicates the point of local minima.

The algorithm for the k-means model is given in Algorithm 1. The value of k is selected in such a way that the average distance from points to centroid decreases rapidly till it converges or changes slowly thereafter.

Algorithm 1. K-Means Algorithm

Input: ϵ, k, Data S

Initialize k centroids randomly, μ_1^t, μ_2^t,……, $\mu_k^t \in R^d$

Output:

```
1:  while $\sum_{i=1}^{k} \left|\left|\mu_i^t - \mu_i^{t-1}\right|\right|^2 \leq \epsilon$ do
2:     $t \leftarrow t + 1$
3:     $C_j \leftarrow \Phi$ for all j = 1, 2, 3, ...., k
4:     for all $x_p \in S$ do
5:         $p^* \leftarrow argmin_p\{\left|\left|x_p - \mu_i\right|\right|^2\}$
6:         $C_{p^*} \leftarrow C_{p^*} \cup \{x_p\}$
7:     end for
8:     for all $i = 1 \ to \ k$ do
9:         $\mu_i^t \leftarrow \frac{1}{C_i} \sum_{x_p \in C_i} x_p$
10:    end for
11: end while
```

3.1.3 k-Medoids Clustering Algorithm

It is known that means, as a statistic, is highly sensitive to the outliers. The k-Means algorithm, that determines and utilizes the means of the data points in calculations, is particularly sensitive to the outliers in the data. To overcome this, a technique of using medoids instead of average values in a cluster is devised. Medoids are centrally located points in a cluster and the technique is called k-Medoids clustering algorithm. Although k-Medoids are computationally more demanding, k-Medoids clusters are not particularly sensitive to the presence of outlier points and are applicable to both continuous and discrete domains of data [169]. This algorithm minimizes the sum of dissimilarities between the objects in a cluster with the reference object selected for that cluster. Basically, the input given is the value of k that represents the number of clusters defined for the data. For each of the k-clusters, k-reference points are selected. The remaining points are clustered into the cluster of a reference point such that the sum of the dissimilarities between the reference object and the points in the cluster is minimized. With different initial medoids selected, the clusters obtained are different. The difference between the k-Means algorithm and the k-Medoids algorithm is that k-Means consider the average value in a cluster to be a reference point and k-Medoids consider the points to be a reference object for the clusters. Algorithm 2 presents the sequence of steps performed in the K-Medoids algorithm.

Algorithm 2. K-Medoids Algorithm

```
Input: $\epsilon$, $k$, Data $S$
Initialize $k$ medoids randomly, $\Theta_1^t, \Theta_2^t, \ldots\ldots, \Theta_k^t \in R^d$
Output:
 1:  while $\sum_{i=1}^{k} \left|\left|\Theta_i^t - \Theta_i^{t-1}\right|\right|^2 \leq \epsilon$ do
 2:    $t \leftarrow t + 1$
 3:    $C_j \leftarrow \Phi$ for all j = 1, 2, 3, ...., k
 4:    for all $x_p \in S$ do
 5:        $p^* \leftarrow argmin_p\{\left|\left|x_p - \Theta_i\right|\right|^2\}$
 6:        $C_{p^*} \leftarrow C_{p^*} \cup \{x_p\}$
 7:    end for
 8:    for all $i = 1\ to\ k$ do
 9:        $\Theta_i^t \leftarrow \frac{1}{C_i} \sum_{x_p \in C_i} x_p$
10:    end for
11:  end while
```

3.2 Transfer Learning

3.2.1 Overview

Data-driven methodologies have been used in different works to forecast energy with different time horizons leading to three branches: long, medium, and short term forecasting [170]. The training of the machine learning models and achieving high accuracy of predictions require a huge amount of historical energy consumption data. Machine learning (ML) algorithms are mainly categorized into three types: supervised, unsupervised, and reinforcement learning models.

Power forecasting in smart grids has employed models such as Autoregressive Integrated Moving Average (ARIMA) [171], Linear Regression (LR) [172], Neural Networks (NN) [148], [173], [174], Support Vector Machines (SVM) [175], and Random Forests [176] in supervised learning. In unsupervised learning, dimensionality reduction models [152], [177] such as PCA, and LDA, and clustering models [174], [178] such as k-Means, and k-Medoids have been used.

With the advent of the Internet of Things (IoT) and smart sensors, the data are generated at a very high frame rate [179]. However, sometimes adequate amount of historical data is not available at different distribution nodes in the electric network. In cases of unavailability of large amounts of historical energy consumption data, it is required that the prediction models are trained with limited amounts of data to achieve sufficiently high accuracy. Furthermore, it is important to note that the supervised machine learning algorithms commonly presume that the training points and testing points belong to the same statistical data distribution and that large amounts of historical data are available [180] [181]. However, the statistical data distribution and patterns of energy consumption have high variability between historical and future data points. Hence, it is crucial to transfer the knowledge obtained from models that are trained on historical data to develop and train machine learning models on current energy consumption data points. The transfer of knowledge is obtained through transfer learning which is detailed in the next section.

3.2.2 Traditional Learning and Transfer Learning Types

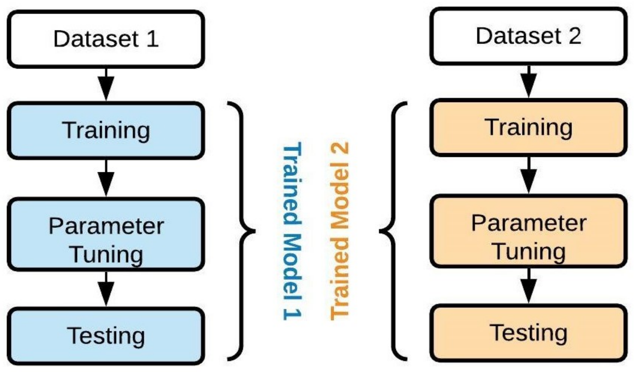


Figure 11 Traditional Learning

Transfer learning is a technique of machine learning in which the knowledge gained during training of a model on a domain of features is leveraged to improve the performance of training another model or task on the same or different domain of features [182]. TL eliminates the assumption that the training data and testing data observe the same data distribution. The merits of TL are the following: training is done with less or little data, training gets faster, and model accuracy increases.

Consider Feature Domain F_s, Label V_s, and Task T_s corresponding to the source application, and Feature Domain F_t, Label V_t, and Task T_t corresponding to the target application. The TL aims to improve the performance of Task T_t using the knowledge obtained in Task T_s where $T_s \neq T_t$.

Figure 11 illustrates the process of traditional machine learning where the knowledge gained after training one model is not retained or reused in further models. The retraining of a newer model or task is executed from scratch. Figure 12 illustrates the process of transfer machine learning where the knowledge gained after training one model (trained model 1 in Figure 12) is transferred to further models (model 2). The weights, knowledge of features, and the network structure are transferred to the training stage for the new task.

The TL process has the benefits of improving the baseline performance of predictions and improving the time to train a machine learning model [181]. There are multiple types of TL algorithms.

1. The Transductive transfer learning (the data features are not the same between the different tasks) [183]: If the tasks T_s and T_t being different infer that the source domain F_s and target domain F_t are also different, then it is called transductive TL.
2. The inductive transfer learning (the data features are the same between different tasks) [184]: If the tasks T_s and T_t being different infer that the source domain F_s and target domain F_t are the same, then it is called inductive transfer learning. If the source label V_s exists, then this learning is called multitask learning. The learning is

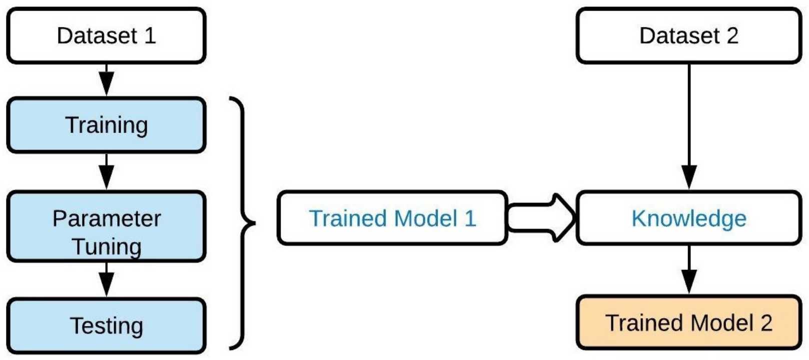


Figure 12 Transfer Learning

unsupervised in the absence of labels in the tasks and in such cases, the algorithm is called self-taught TL.

3. The unsupervised transfer learning [185]: In this type of learning, the source tasks T_s & T_t are different, the domains F_s & F_t are similar, and the labels are not available in both tasks.

3.2.3 Theoretical perspective of TL in cross-model load forecasting using NN

Consider a trained neural network structure with three layers as shown in Figure 13. The input layer with $I+1$ inputs with $(I+1)$th node as bias node, $H+1$ hidden units with $(H+1)$th node as bias node, and P outputs. Consider that the neural network model is already trained on training data with N records, i.e. $\{(x_1,y_1),(x_2,y_2),....,(x_N,y_N)\}$. Since the training is complete, it is safe to assume that the optimal weights have been determined with objective function on minimum training error. Consider that the weights between the input-hidden connections and hidden-output connections are w_{ih} and v_{hp}, where $1 \le i \le I+1, 1 \le h \le H+1$, and $1 \le p \le P$. With transfer learning, it is expected to train the model with training record $N+1$ (refers to training record from new dataset), input x_{N+1} such that the predicted value from the model is equal to the true value of output, i.e., $y_{N+1} = \hat{y}_{N+1}$. The transfer training with data from a new dataset should minimize the effect on training errors (E_n $(1 \le n \le N)$ of previous historical data i.e. minimize the weight sensitivity. The cost objective for weight sensitivity can be given by $T \triangleq \frac{1}{8}\sum_{n=1}^{N}\sum_{p=1}^{P} \triangle E_{np}^2$. The goal of transfer learning is to determine the weights $4w_{ih}(N+1)$ and $v_{hp}(N+1)$ such that these do not have any effect on weight sensitivity represented by the objective function (S) that balances the trade-off between weight sensitivity objective function T and error of prediction, for instance, $N + 1$. The objective function S is given by the following:

$$S \triangleq T + \frac{\lambda}{2}\sum_{p=1}^{P}\left(y_{(N+1)(p)} - \hat{y}_{(N+1)(p)}\right) \tag{2}$$

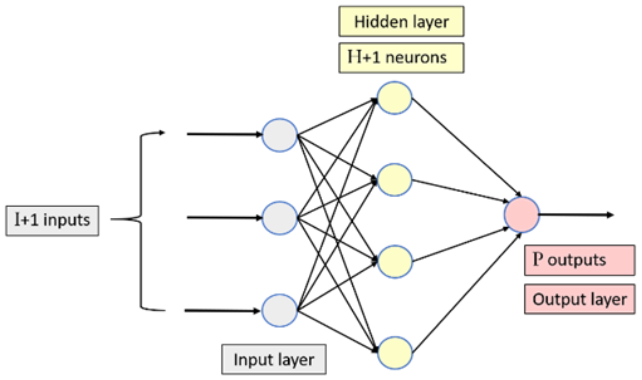


Figure 13 Neural network perceptron

where λ is the trade-off coefficient to balance the evolutionary training error and pre-evolutionary training error.

$$\mathrm{T} \triangleq \frac{1}{8}\sum_{n=1}^{N}\sum_{p=1}^{P}\triangle \mathrm{E}_{np}^{2} \tag{3}$$

$$T \triangleq \frac{1}{8}\sum_{n=1}^{N}\sum_{p=1}^{P}\left(\sum_{ih}^{(I+1)H}\frac{\delta E_{np}}{\delta w_{ih}}\triangle w_{ih}(N+1) + \sum_{h}^{(H+1)}\frac{\delta E_{np}}{\delta v_{hp}}\triangle v_{hp}(N+1)\right)^{2} \tag{4}$$

The weight sensitivities of change in error can be given by (5) and (6).

$$\frac{\delta E_{np}}{\delta w_{ih}} = \frac{\delta}{\delta w_{ih}}\left(y_p(n) - \widehat{y_p}(n)\right)^2 \tag{5}$$

$$\frac{\delta E_{np}}{\delta w_{ih}} = 2 * \left(\mathrm{y_p(n)} - \widehat{\mathrm{y_p}}(\mathrm{n})\right)\left(0 - \frac{\delta \widehat{\mathrm{y_p}}(\mathrm{n})}{\delta \mathrm{w_{ih}}}\right)$$

$$\frac{\delta E_{np}}{\delta w_{ih}} = -2 * \left(\mathrm{y_p(n)} - \widehat{\mathrm{y_p}}(\mathrm{n})\right)\frac{\delta \widehat{\mathrm{y_p}}(\mathrm{n})}{\delta \mathrm{w_{ih}}}$$

$$\frac{\delta E_{np}}{\delta v_{hp}} = \frac{\delta}{\delta v_{hp}}\left(y_p(n) - \widehat{y_p}(n)\right)^2$$

$$\frac{\delta E_{np}}{\delta v_{hp}} = 2 * \left(\mathrm{y_p(n)} - \widehat{\mathrm{y_p}}(\mathrm{n})\right)\left(0 - \frac{\delta \widehat{\mathrm{y_p}}(\mathrm{n})}{\delta \mathrm{v_{hp}}}\right) \tag{6}$$

$$\frac{\delta E_{np}}{\delta v_{hp}} = -2 * \left(\mathrm{y_p(n)} - \widehat{\mathrm{y_p}}(\mathrm{n})\right)\frac{\delta \widehat{\mathrm{y_p}}(\mathrm{n})}{\delta \mathrm{v_{hp}}}$$

From (5) and (6), the equation (4) is modified as the following

$$T \triangleq \sum_{n=1}^{N}\sum_{p=1}^{P}\left[\sum_{ih}\left[-\left(y_p(n) - \widehat{y_p}(n)\right)\frac{\delta\widehat{y_p}(n)}{\delta w_{ih}} \triangle\, w_{ih}(N+1)\right]\right] + \sum_{h}\left[-\left(y_p(n) - \widehat{y_p}(n)\right)\frac{\delta\widehat{y_p}(n)}{\delta v_{hp}} \triangle\, v_{hp}(N+1)\right]^2 \tag{7}$$

Let $y_p(n) \triangleq \sum_h^{H+1} u_h(n)v_{hp}$ and $u_h(n) \triangleq f\left(u_h^*(n)\right)$, where $f(.)$ is the activation function of the hidden layer neuron, and $u_h^*(n) \triangleq \sum_i^{I+1} x_i(n)w_{ih}$. It is important to note that $x_{I+1}(n) = 1$ and $u_{H+1}(n) = 1$ since the input node $I + 1$ and hidden node $H + 1$ are bias neurons in the artificial neural network considered. Therefore, the change of prediction with respect to the weights in the hidden-to-output layer connections is given by the following:

$$\frac{\delta\widehat{y_p}(n)}{\delta v_{hp}} = u_h(n) \tag{8}$$

where $1 \le h \le H + 1, 1 \le p \le P$.

The change of prediction with respect to the weights in the input-to-hidden layer connections is given by the following:

$$\frac{\delta \widehat{y_p}(n)}{\delta w_{ih}} = \left[\frac{\delta \widehat{y_p}(n)}{\delta u_h(n)}\right]\left[\frac{\delta u_h(n)}{\delta w_{ih}}\right]$$

$$\frac{\delta \widehat{y_p}(n)}{\delta w_{ih}} = \left[\frac{\delta \widehat{y_p}(n)}{\delta u_h(n)}\right]\left[\frac{\delta u_h(n)}{\delta u_h^*(n)}\right]\left[\frac{\delta u_h^*(n)}{\delta w_{ih}}\right]$$

$$\frac{\delta \widehat{y_p}(n)}{\delta w_{ih}} = v_{hp}(n)\frac{\delta f(x)}{\delta x} x_i(n)|\{x = u_h\} \tag{9}$$

$$\frac{\delta \widehat{y_p}(n)}{\delta w_{ih}} = v_{hp}(n)u_h(n)\big(1 - u_h(n)\big)x_i(n)$$

where $1 \le i \le I + 1, 1 \le h < H, 1 \le p \le P$.

It implies that,

$$T = \frac{1}{8}\sum_{n=1}^{N}\sum_{p=1}^{P}[-(y_p(n) - \widehat{y_p}(n)\,) * \sum_{ih}^{(I+1)H}\left[v_{hp}(n)u_h(n)\big(1 - u_h(n)\big)x_i(n) \bigtriangleup w_{ih}(N+1)\right] - (y_p(n) - \widehat{y_p}(n) \sum_{ih}^{(I+1)H}\left[u_h(n) \bigtriangleup v_{hp}(N+1)\right]^{2} \tag{10}$$

In transfer learning, the aim is to minimize the objective function S that balances the trade-off between minimizing weight sensitivity T on a historical trained model and the error of predictions on data from a new dataset. i.e., $S \triangleq T + \frac{\lambda}{2}\,(\,\sum_{p=1}^{P}\big[\big(y_p(n+1) - \widehat{y_p}(n+1)\,\big)\,\big\}$.

3.2.4 Related work

In the past decade, transfer learning has gained widespread research interest from researchers in different fields of study owing to its inherent capability of transferring the knowledge gained while training from one application to another. In [186], the authors used TL used with seasonal and trend adjustment to enhance forecasts on energy used in a building with the aid of models trained on data from similar buildings. An improvement of 11.2% in Mean Absolute Percentage Error (MAPE) of predictions was reported after the use of TL. Their work assumes the similarity of buildings in terms of energy consumption to apply TL and did not employ clustering-based techniques to group different buildings. Their case study also limits the application of TL to similar buildings. In this work, the clustering-based techniques are employed and also, TL is applied to similar distribution nodes with an improvement of training time and testing accuracy and between dissimilar clusters with an improvement of training time.

In [187], energy predictive models based on convolutional neural networks (CNN) and TL are proposed. In their work, energy predictive models were tested on a case study of 23 customers against the Seasonal Autoregressive Integrated Moving Average (SARIMA) model and fresh CNN model. The results proved that the performance in terms of accuracy is improved when the models are pre-trained using TL.

In [188], Ye et al. proposed an ensemble model of online TL kernel-based extreme learning machines. The results presented in their work depict that the use of TL improves the performance in terms of accuracy compared to standard machine learning models. Their work utilizes extreme learning machines that are basically neural networks with one hidden layer. The developed approach provided many benefits such as eliminating the need for optimizing the number of hidden layers and a smaller number of parameters to be optimized while using extreme learning machines [154]. However, the deep learning models have displayed high accuracy while dealing with the time-dependent energy forecasting problems if the tendency to over-fitting is controlled [189]. Hence, in this work, the use of TL is extended to deep learning models.

In [190], the authors proposed a two-stage prediction model for wind power based on an ensemble of nine deep auto-encoders in the first phase and deep belief networks in the second phase. The work was based on five datasets from wind farms. The TL was utilized in the training of deep auto-encoders from two to nine using the knowledge obtained during the training of the first deep autoencoder. Their results indicate that the use of TL overperforms the baseline regression models based on ARIMA and Support Vector Regression (SVR). However, the performance comparison of the ensemble model without TL and with TL has not been discussed. It is unclear if the improvement in performance is due to the ensemble of the optimized deep auto-encoders or due to TL.

3.3 Incremental Learning

3.3.1 Overview

In the case of online streams of data available and the predictions to be made after these data are available, it becomes necessary to update the models with the new data points. With a new set of information available, the global minimum of the model cost function varies from the existing one. That is the global minimum is reset every time a new data record is available. This calls for the updating of the network weights using the new batch of data records available.

Owing to the characteristics of the electrical data available, the error function typically has a highly nonlinear relationship with the network parameters such as weights. This makes it possible for the multiple local minima to exist i.e. several weight combinations will give a small error. For the successful application of the forecasting model using deep neural networks, it is required that the global minimum or the lowest local minima are determined. However, with streams of new data available, these minima are changed. To find a solution that updates with time and data available, it is necessary to compare the minima and update the weights periodically by running batch online learning continually at regular intervals.

In incremental machine learning, the models learn 1 observation at a time or 'n' observations at a time. The value 'n' depends on the number of hours after which the incremental machine learning algorithm is invoked.

The incremental machine learning provides the benefits of a smooth transition between successive models and consistency of data [191]. The new models are developed incrementally using the models developed on historical old data and new data points. There is no requirement to develop the models from scratch when new batches of data are available. Also, the consistency is maintained i.e. the performance of the incrementally trained model is unchanged for records comprised in the initial data instances.

Usually, the incremental machine learning based on transfer learning suffers which phenomenon called catastrophic forgetting. This calls for the tradeoff between two characteristics called rigidity (rigid with old tasks and so perform better with old data instances) and plasticity (flexible with new tasks and so perform better with new data instances). However, in this methodology, the incremental machine learning is performed on the clustered models which are then incrementally trained on new datasets from individual transformers. This avoids catastrophic forgetting for a significant duration of time initially, however in long run would lead to low accuracies.

3.3.2 Related work

There have been few works that have capitalized on the incremental learning approach to develop short-term energy forecasting models.

Polikar et al. introduced incremental learning initially in the neural networks for classification tasks [192]. For every mini-batch of data, they generated a different hypothesis, and the result of classification is obtained by the aggregation of all the classes obtained by the ensemble of classifiers on the different mini-batches of data. This algorithm is termed Learn++. Learn++ proved to present new classes from new batches of data.

To tackle the issues of changing patterns in a predicted variable, Sanchez et al., in their work [193], introduced a forgetting function to enable the adaptability of the models to the changing patterns. For applications where the patterns of predicted variables change over time, the results of predictions are based on incremental learning with higher weights given to the knowledge from new data and low weights to the knowledge from an old hypothesis. The proposed forgetting function methodology was applied to a two-layer feedforward neural network.

Zribi et al. proposed an incremental learning methodology for neural networks wherein they start the development of a network from a simple structure to adding a neuron to the hidden layers until the error of the predictions reaches a threshold while incrementing the data [194].

In [195], Qiu et al. proposed an incremental approach while using an ensemble model of Discrete Wavelet Transform (DWT)- Empirical Mode Decomposition (EMD) based Random Vector Functional Link network (RVFL) network for short-term load forecasting. However, the RVFL network is based on randomly generated weights between input and hidden layers, rather than a systematic approach to parameter optimization leaving scope for accuracy improvement. Reference [196] proposed an online support vector regression (SVR) based on nested particle swarm optimization (NPSO) for parameter optimization.

In [197], Gabriela et al. proposed an incremental ensemble model utilizing the time-series characteristics such as seasonality, and concept drift of energy consumption data. The ensemble model was based on Multi-Linear Regression (MLR), Support Vector Machine (SVR), Seasonal decomposition of time series by loess (STL), Holt-Winters exponential smoothing (HW), Feed-forward neural networks (NNE), and autoregressive model (AR).

CHAPTER IV

MACHINE LEARNING AND DEEP LEARNING MODELS*[5,6,7]

In this chapter, the utilized and proposed machine learning and deep learning models are detailed. The research utilizes the existing machine learning and deep learning models for purposes of benchmarking the performance. Additionally, the research has proposed several deep learning and ensemble architectures.

4.1 Linear Regression (LR)

The hypothesis of LR is given by:

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + \cdots . + w_nx_n \quad , \tag{11}$$

where $x_1, x_2, \ldots \ldots .., x_n$ represent features, and $w_0, w_1, w_2, \ldots .., w_n$ represent model parameters.

The objective of the LR is to choose $w_0, w_1, w_2, \ldots .., w_n$ so that the values of $h_w(x)$ is as close to the actual values of the labels (y). This is achieved by the introduction of a constraint while determining $w_0, w_1, w_2, \ldots .., w_n$.

Conventionally, the objective constraint in LR is given by Equation **3** [198].

$$J(w_0, w_1, \ldots ., w_n) = \underset{w_0 w_1 \ldots w_n}{minimize} \frac{1}{2n} \sum_{k=1}^{n} \left(h_w\left(x^{(k)}\right) - y^{(k)}\right)^2 \ , \tag{12}$$

Here, $J(w_0, w_1, \ldots ., w_n)$ is the cost function in terms of model parameters.

---

[5] Reprinted with permission from "Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid with Clustering and Consumption Pattern Recognition." by Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Shady S. Refaat, Mahdi Houchati, Othmane Bouhali, and Santiago Bañales, 2021. IEEE Access 9, 54992-55008, Copyright 2021 by Dabeeruddin Syed.

[6] Reprinted with permission from "Averaging Ensembles Model for Forecasting of Short-term Load in Smart Grids." by Dabeeruddin Syed, Shady S. Refaat, Haitham Abu-Rub, Othmane Bouhali, Ameema Zainab, and Le Xie, 2019. IEEE International Conference on Big Data, pp. 2931-2938, Copyright 2019 by IEEE.

[7] Reprinted with permission from "Household-Level Energy Forecasting in Smart Buildings Using a Novel Hybrid Deep Learning Model." by Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, and Shady S. Refaat, 2021. IEEE Access 9, 33498–33511, Copyright 2021 by Dabeeruddin Syed.

This constraint is basically the sum of squared error and the aim is to minimize this error while determining the weights.

The LR has been used as one of the prediction models to act as a benchmark for training time as this model would have the lowest training time owing to the simplicity of the model but coarser accuracy.

4.2 Deep Neural Networks (DNN)

If the artificial neural networks have multiple hidden layers between the input layer and the output layer, then these are termed as DNNs [199]. DNNs have the capabilities of modeling linear or non-linear relationships between the data features. Also, the tendency to overfit can be reduced with the application of dropout where the neurons in random or systematic order are dropped.

The non-linear function representing the data is effectively determined in the neural networks using summation and product operations. If a neuron 'j' of layer 'l' (depicted in Figure 14 from a neural network is considered, then the input to the neuron is S_j^l, the weight at the neuron is w_{ij}^l. Let σ be the activation function, then x_j^l is the output from the neuron and this output acts as input to the neurons in the next layer. Here, i represents the neuron number in the previous layer and d^l represents the number of neurons in the layer 'l'.

The input to the neuron S_j^l is given as

$$S_j^l = \sum_{i=1}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}, \qquad (13)$$

And, the output from the neuron j in layer l is given as follows.

$$x_j^l = \sigma\left(S_j^l\right) , \tag{14}$$

In the matrix form, the equation for the input to neuron in layer l is given as:

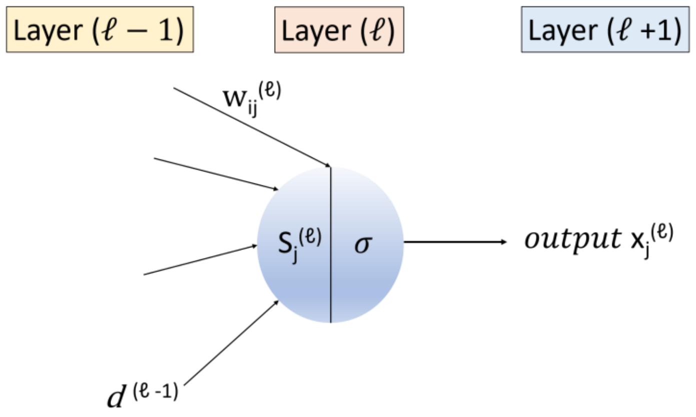


Figure 14 Neuron 'j' of layer 'l' in a neural network

$$\underline{S}^l = (W^l)^T \underline{x}^{l-1} , \tag{15}$$

This equation is used in the forward propagation calculations. The value $\underline{x}$, which is input, is available initially. It is used with pre-initialized weights $W^{(1)}$ to calculate the input $\underline{S}^{(1)}$ to the neurons in the hidden layer 1. This input when applied with activation function yields the output $\underline{x}^{(1)}$ from the neuron in hidden layer 1.

Graphically, the forward propagation can be represented as the calculations involving the following:

$$\underline{x} \equiv \underline{x}^{(0)} \xrightarrow{W^{(1)}} \underline{S}^{(1)} \xrightarrow{\sigma} \underline{x}^{(1)} \xrightarrow{W^{(2)}} \underline{S}^{(2)} \xrightarrow{\sigma} \underline{x}^{(2)} \ldots\ldots\ldots\ldots \xrightarrow{W^{(L)}} \underline{S}^{(L)} \xrightarrow{\sigma} \underline{x}^{(L)} \equiv \mathrm{h}(\underline{\mathrm{x}}) , \tag{16}$$

The algorithm for the forward propagation of the neural network is given in Algorithm 3. The aim of the forward propagation is to calculate the inputs and outputs in different layers of the network using the weights, bias, and activation functions.

The backpropagation is utilized to determine the gradient of error in the direction of the last hidden layer to the first hidden layer while minimizing the gradient of the error with respect to the weights of neurons.

Algorithm 3. Forward Propagation

```
Input: $\underline{x}^{(0)} = \underline{x}$. Initialization of augmented vector
Output:
1:   for $l = 1, 2, 3, ..., L$ do
2:      compute $\underline{S}^{l} = (W^{l})^{T}\underline{x}^{l-1}$
3:      compute $\underline{x}^{(l)} = \begin{bmatrix} 1 \\ \sigma(\underline{S}^{l}) \end{bmatrix}$
4:   end for
5:   compute $h(\underline{x}) = \underline{x}^{(L)}$
```

The error associated with the predictions is given by Equation 8 [200]. The subsequent Equations 9, 10, and 11 are the calculation of error gradient with respect to the weight of the corresponding layer.

$$E = \frac{1}{N}\sum_{n=1}^{N}\frac{\delta e_n}{\delta W}\left(h\left(\underline{x}_n - y_n\right)^2\right), \tag{17}$$

$$\frac{\delta E}{\delta W^{l}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\delta e_n}{\delta W^{l}} , \tag{18}$$

where $\frac{\partial e_n}{\partial W^{l}}$ is given by $\frac{\partial e_n}{\partial w_{ij}^{l}}$ that is equal to

$$\frac{\partial e_n}{\partial w_{ij}^{l}} = \frac{\partial e_n}{\partial s_j^{l}}\frac{\partial s_j^{l}}{\partial w_{ij}^{l}} , \tag{19}$$

$$\frac{\partial e_n}{\partial w_{ij}^l} = \frac{\partial e_n}{\partial s_j^l} x_i^{l-1} \quad , \tag{20}$$

This brings the partial derivative of the error with respect to neuron weights to the following equation:

$$\frac{\partial e_n}{\partial w_{ij}^l} \equiv \frac{\partial e}{\partial W^l} = \delta_i^l x_i^{l-1} \, , \tag{21}$$

In the backpropagation, the error gradient $\delta_i^{(L)}$ is determined first (L represents the last layer in the neural network) and by way of backpropagation, the error in the previous layers is calculated as the following:

$$\begin{aligned} \delta_i^{l-1} &= \frac{\partial e(w)}{\partial s_i^{l-1}} \\ &= \sum_{j=1}^{d^l} \frac{\partial e(w)}{\partial s_j^l} \frac{\partial s_j^l}{\partial x_i^{l-1}} \frac{\partial x_i^{l-1}}{\partial s_i^{l-1}} \\ &= \sum_{j=1}^{d^l} \delta_j^{(1)} w_{ij}^l \frac{\partial x_i^{l-1}}{\partial s_i^{l-1}} \, , \end{aligned} \tag{22}$$

The above equation is the representation of the error gradient of a layer in terms of the error gradient of the next layer.

All the steps of forming a DNN are provided in Algorithm 4.

4.3 Long Short Term Memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN) that predicts the output based on not only the current state of the hidden units but also on the previous states witnessed so far, with the help of storing information in memory blocks. LSTMs are sequential

Algorithm 4. Neural Network

```
Input: Initialize all weights $w_{ij}^{(l)}$
Output:
1:  for $t = 0, 1, 2, 3, \ldots,$ do
2:      pick $n \in 1, 2, 3, \ldots, N$
3:      Forward propagation: compute all $x_j^l$
4:      Backward propagation: compute all $\delta_j^l$
5:      update the weight: $w_j^{(l)} \overset{w}{\leftarrow} w_j^{(l)} - \eta\, x_i^{l-1}\, \delta_j^l$
6:      iterate to the next step until it is time to stop
7:  end for
8:  return the final weights $w_{ij}^{(l)}$
```

models and hence, capture the temporal dependencies. These models are suitable for processing time-series data such as load forecasting data. In a standard RNN, there are two inputs at a time step t to a neuron: input of time step t (x_t) and output obtained at time step t-1 (h_{t-1}). Output at a time step is obtained by the weighted sum of x_t and h_{t-1} which is then followed by using activation functions such as Rectified Linear Unit (ReLU), and hyperbolic tangent (tanh) on the weighted sum.

LSTM places a mini neural network inside each neuron and therefore complicates the process of training. However, it helps to improve reliance and handles the long-term dependencies well by eliminating the issues of gradient vanishing and gradient explosion that usually exist with the use of standard RNN. The main idea of LSTM is to have two outputs and gates. One of the outputs goes to the output layer and the next time step. Besides, the other output goes to the next time step only. Gates are the multiply operations performed and there are several gates in the LSTM. The LSTM network determines the weights, and these weights are used to dot-product the inputs.

An LSTM layer followed by a fully connected neural network is depicted in Figure 16.

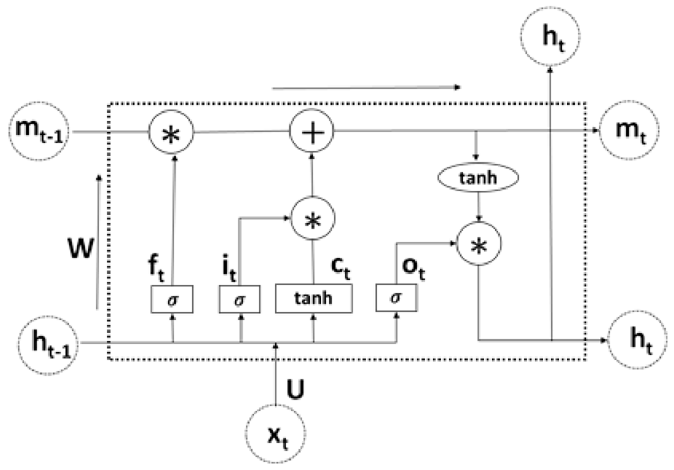


Figure 15 Mathematical operations in an LSTM unit at one time step

To understand the working of LSTM to handle long-term dependency, consider an LSTM unit at a particular time step as shown in Figure 16. Unlike standard RNNs, LSTMs have complex mathematical operations, additional three gates termed as forget (f_t), input (i_t) and output (o_t) gates, and additional layer called candidate layer [173], [201]. The gates utilize the sigmoid activation function and candidate layer operates using tanh

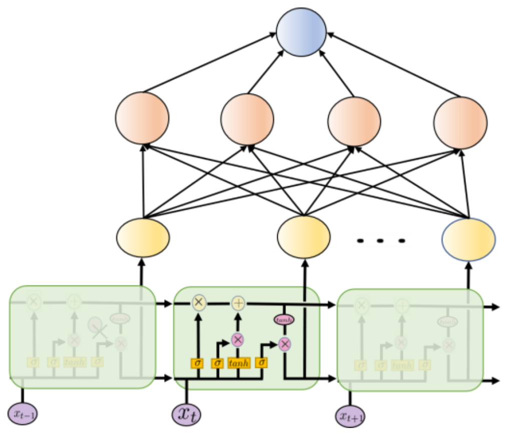


Figure 16 LSTM with Fully Connected NN

activation. Tanh activation is additionally adopted to calculate hidden cell state at next time step from prior memory cell state.

From Figure 15, it can be observed that the inputs to an LSTM neuron at time step t are the current input X_t, hidden state from prior time step h_{t-1}, and memory cell state from prior time step m_{t-1}. The outputs from the LSTM cell are hidden state at current time step h_t, and memory state at current time step m_t. There are multiple operations taking place inside LSTM cell than those that occur in RNN cell. The merits of these operations are that the gradients are preserved through the network and these allow for long-term dependency information to pass.

The forget gate looks similar to the input gate, however, its function is different, and it is to determine the usefulness of the previous hidden state memory cell whilst computing on current input.

$$f_t = \sigma\left(h_{t-1} \cdot W_f + x_t \cdot U_f + b_f\right) \tag{23}$$

The input gate performs a significant function. It determines if the current input requires to be preserved based on the prior hidden state value.

$$i_t = \sigma(h_{t-1} \cdot W_i + x_t \cdot U_i + b_i) \tag{24}$$

The candidate layer takes the current input x_t and prior hidden state value h_{t-1} and creates a new memory c_t.

$$c_t = tanh(h_{t-1} \cdot W_c + x_t \cdot U_c + b_c) \tag{25}$$

The output gate differentiates the hidden state memory and determines how much of the information in the memory should be present in the hidden state at time t as h_t is dependent on o_t.

$$o_t = \sigma(h_{t-1} \cdot W_o + x_t \cdot U_o + b_o) \quad (26)$$

As shown in (27), the final memory state m_t is generated by two product sums: i) taking forget gate output f_t and prior memory state value m_{t-1}, and ii) input gate output i_t and new memory c_t.

$$m_t = f_t \cdot m_{t-1} + i_t \cdot c_t \quad (27)$$

The final hidden state value is dependent on the output gate as formulated in the following:

$$h_t = o_t * tanh(m_t) \quad (28)$$

where b_f, b_i, b_c are time step independent biases of forget gate, input gate, and candidate layer respectively. W_f, U_f, W_i, U_i, and W_c, U_c are time step independent weights of forget gate, input gate, and candidate layer respectively.

4.4 Proposed Averaging Regression Ensembles Model (AREM)

Also, an ensemble model that provides better performance in terms of accuracy is proposed. The proposed Averaging Regression Ensembles Model is based on three ensembles model consisting of ensemble extreme learning machines model, linear regression, and long short-term memory recurrent neural network model. For the ensemble of extreme learning ensembles model, the weights are randomly initialized, and the number of neurons used in every model is different. Different learning algorithms were used, however, the better performing learning algorithms such as LBFGS and Adam have been finalized for the ensemble of extreme learning machines model, and learning algorithms LBFGS and Adam have been finalized for LSTM recurrent neural network model. The ensemble model, called as Averaging Regression Ensembles Model (AREM) model, is depicted in Figure 17 [154].

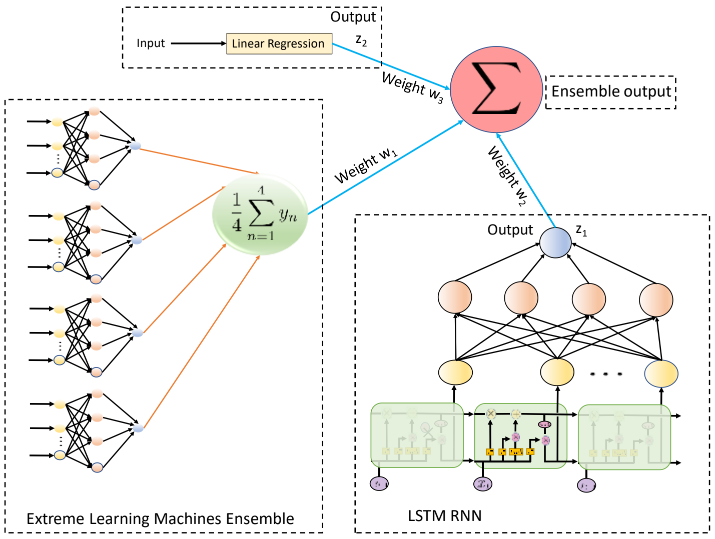


Figure 17 Architecture of Proposed AREM Model © 2019 IEEE

4.5 Proposed Bidirectional LSTM Architecture (Bi-LSTM)

Bi-directional LSTM is a development over uni-directional LSTM models. The bi-directional LSTMs process the inputs in two directions - in the forward pass, from past inputs to future inputs, and in the backward pass, from future inputs to past inputs. The combination of hidden states from the forwarding pass and backward pass preserves the information from both past inputs and future inputs through two different hidden layers. The output from these hidden layers is passed to the single identical output layer. This allows the bidirectional LSTMs to preserve the context and data patterns better from both past and future inputs without delay. It has been proven that bi-directional LSTMs perform better predictions and classifications than uni-directional LSTMs in diverse fields such as speech recognition [202]. However, the advantages of bi-directional LSTMs have not been much explored in the field of energy consumption forecasting in smart grids.

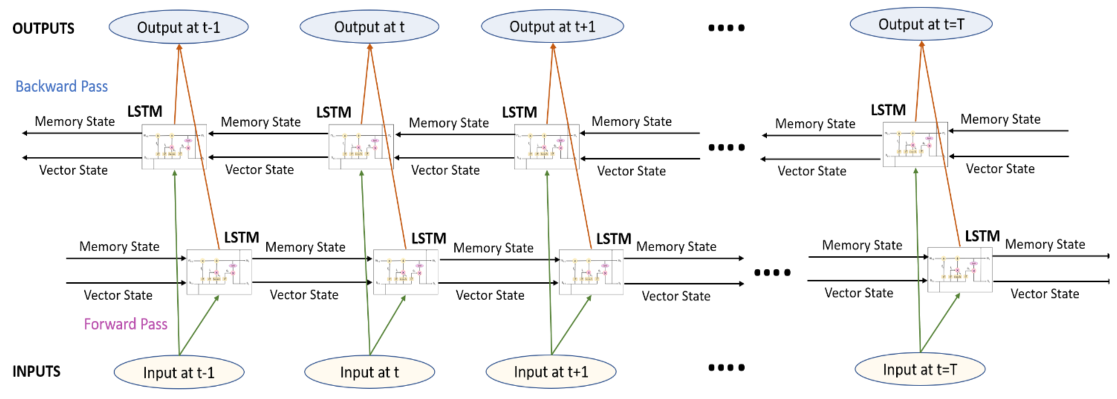


Figure 18 The architecture of unfolded bi-directional LSTM model © 2019 IEEE

The architecture of the unfolded bi-directional LSTM model, comprising of forwarding LSTM units and backward LSTM units, is depicted in Figure 18 [173]. The forward pass output ($\vec{h}$) is successfully determined using inputs in the positive sequence of time from $T - k$ to $T - 1$. Whereas the backward pass output ($\overleftarrow{h}$) is successfully determined using inputs in the negative sequence of time from $T + k$ to $T + 1$. There are no hidden-to-hidden layer connections between the forward LSTM units and the backward LSTM units. The calculations of outputs of the forward pass and backward pass utilize the traditional LSTM functions. The final output vector of the bi-directional LSTM layer is represented as $Z_T = [z_{T-k}, z_{T-k+1}, \dots., z_{T-1}]$. Each element in the final output vector is given by the following:

$$z_t = \sigma(\vec{h}_t, \overleftarrow{h}_t) \tag{29}$$

where σ represents a function used to integrate the outputs from forwarding pass and backward pass. The σ function can be a summation, averaging, concatenating, or multiplication function. And the forward pass output ($\vec{h}$) and backward pass output ($\overleftarrow{h}$) are given as follows.

$$\vec{h} = \mathrm{H}\left(W_{y\vec{h}}\mathrm{y}_t + W_{\vec{h}\vec{h}}\vec{h}_{t+1} + b_{\vec{h}}\right) \tag{30}$$

$$\overleftarrow{h} = \mathrm{H}\left(W_{y\overleftarrow{h}}\mathrm{y}_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right) \tag{31}$$

where H represents the hidden layer function, y_t represents the input sequence.

4.6 Proposed Clockwork RNN (CWRNN)

Clockwork RNNs are modifications to the vanilla RNNs such that the hidden layers in clockwork RNNs are partitioned into different modules. Each of the modules then processes the inputs to them at different temporal granularities and different preset clock rates. The advantages of such architecture are the reduction in the number of trainable parameters and an increase in accuracy when compared to the regular RNN and LSTM structures.

Similar to RNNs, clockwork RNNs also have the input to hidden layers, hidden to hidden layers, and hidden to output layer connections. The only difference is that the hidden layer in clockwork RNNs is partitioned into k partitions of size m, each of it will have the clock period $T_n = \{T_1, T_2, \ldots, T_k\}$.

The different partitions have internal connections together, but there are no recurrent connections from i to j if the time period $T_i < T_j$. That is, when the neural network is built, the partitions are arranged in ascending order and the recurrent connections exist from the partitions with higher clock periods to partitions with lower clock periods only.

The architecture of the clockwork RNN (CWRNN) is illustrated in Figure 19. The clock period of a partition i can be determined as the following:

$$T_i = 2^{i-1}\,, \tag{32}$$

where i is the number of partition of a hidden layer.

The input and hidden weight matrices are partitioned into k block rows, as follows:

$$W = \begin{bmatrix} W_1 \\ . \\ . \\ . \\ W_k \end{bmatrix} \; and \; U = \begin{bmatrix} U_1 \\ . \\ . \\ . \\ U_k \end{bmatrix}, \tag{33}$$

Here, W is an upper triangular matrix, and each W_i value is partitioned into block columns:

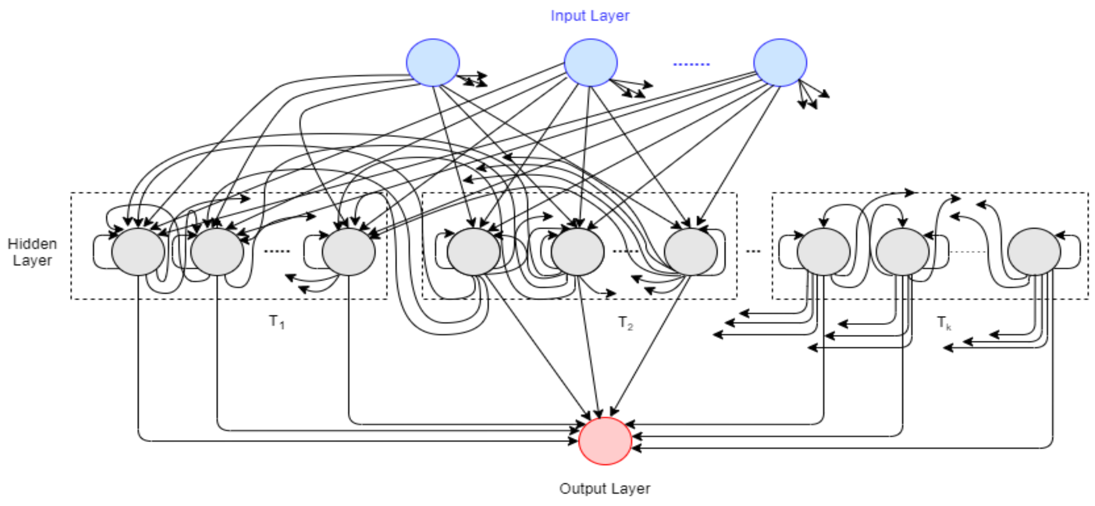


Figure 19 The architecture of clockwork RNN

$$\{0_1, \dots, 0_{i-1}, W_{i,i}, \dots, W_{i,k}\} \tag{34}$$

During the forward pass, only the block rows of the hidden weight matrix and the input weight matrix correspond to the executed modules. That is,

$$W_i = \begin{cases} W_i, & for\ (t\ mod\ T_i) = 0 \\ 0, & otherwise \end{cases} \tag{35}$$

where i represents the missing record number.

The modules with lower clock rates learn and maintain long-term information from the input and the modules with the higher clock rates learn local information. Each hidden layer is partitioned into k modules of size m, which means there are a total of $n = mk$ neurons. Since neurons are only connected to those that have a similar or larger period, the number of parameters within the hidden-to-hidden matrix is as follows:

$$N_H = \sum_{i=1}^{k} \sum_{k=1}^{m} m(k - i + 1) = \frac{n^2}{2} + \frac{nm}{2} \tag{36}$$

In vanilla RNNs, there are n^2 parameters. If the ratio of N_H to n^2 is considered, it is around $1/2$. Therefore, Clockwork RNN requires approximately 50% of the parameters.

CHAPTER V

PROPOSED METHODOLOGY*[8,9]

This chapter presents the multi-stage hybrid methodology proposed in this thesis for short-term load forecasting in large electrical networks. The main aim of the work is to train and build a deep learning hybrid model that provides high accuracy for electrical energy forecasts. The hourly energy forecasts one-day ahead are the goal of the work.

A detailed description of the various steps involved in the proposed methodology is described in subsequent sections starting with the description of acquisition data sets used for the case studies.

5.1 Data Acquisition

The load forecasting data available for analysis is the energy consumption data at the distribution transformers' level in Spanish Electrical Network. The data contain the hourly energy consumption data for 100,000 distribution transformers. The features in this dataset are timestamp, and energy consumption values. Feature engineering is performed to use the 24 energy lag values as features in the dataset to incorporate time dependency in the machine learning models to be developed.

The energy consumption data for 100,000 transformers would be around 4.8 billion rows for a five-year historical time. This corresponds to about 5.7 Terabytes of memory for data processing. The methodologies to be developed are required to use the

---

[8] Reprinted with permission from "Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid with Clustering and Consumption Pattern Recognition." by Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Shady S. Refaat, Mahdi Houchati, Othmane Bouhali, and Santiago Bañales, 2021. IEEE Access 9, 54992-55008, Copyright 2021 by Dabeeruddin Syed.

[9] Reprinted with permission from "Short-term Power Forecasting Model Based on Dimensionality Reduction and Deep Learning Techniques for Smart Grid." by Dabeeruddin Syed, Shady S. Refaat, Haitham Abu-Rub, and Othmane Bouhali, 2020. IEEE Kansas Power and Energy Conference (KPEC), pp. 1-6, Copyright 2020 by IEEE.

concepts of big data, parallel computing, and clustering techniques to reduce the computation time for initial training of models and incremental online update of trained models dynamically. This additionally calls for the concepts of transfer learning and incremental machine learning to be incorporated in the proposed methodology.

The methodologies have been proposed and these methodologies are scaled to be accurate, efficient, and with low latency as the size and rate of data increases. The data is used in the multiples of 10, 1000, and 100000 transformers to develop different strategies and algorithms.

5.1.1 Dataset 1

The load forecasting data available for analysis is the energy consumption data at the distribution transformers' level for Spain. The data contain the hourly energy consumption data for 10 distribution transformers. The weather data for the location of these 10 distribution transformers is scraped online using an Application Programming Interface (API) named Darksky [203]. The data is available for 33 months from 01 January 2017 to 28 September 2019. The weather data is merged with the energy consumption data. The data contain missing values for the weather features. The missing values should be either filled, extrapolated, or deleted [204]. The missing values for numerical features are filled with mean or median values. Otherwise, backward or forward filling methods are utilized to fill the missing values. Mode imputation is applied for categorical or ordinal features. In this work, the forward fill method has been used to fill missing values for numerical weather features. The dataset 1 consists of features such as datetime, wind speed, maximum temperature, minimum temperature, humidity, summertime, and other weather features in addition to energy consumption values and lag hour values. Figure 20 presents the standard deviation and mean of the energy consumption values for different transformers in dataset 1. The boxes represent the deviation of the energy consumption values, the horizontal blue line inside the blue box represents the median of the consumption, and the black circles represent the outliers. The higher width of the blue boxes represents that the energy consumption values for those transformers are highly varying.

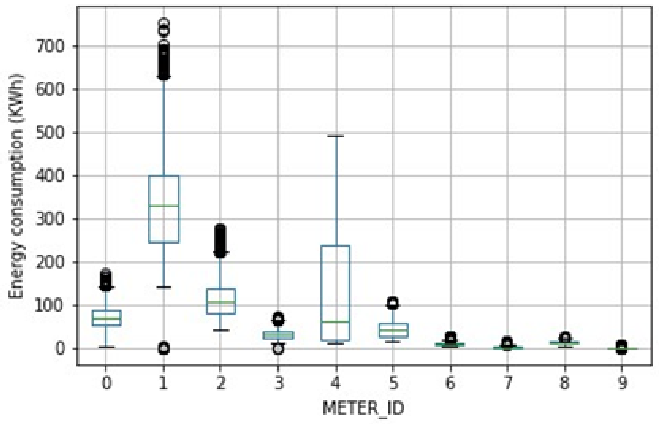


Figure 20 The standard deviation of energy consumption for different transformers in Dataset 1

5.1.2 Dataset 2

The load forecasting data available for analysis is the energy consumption data at the distribution transformers level for Spain. The data contains the hourly energy consumption data for 1000 distribution transformers. However, the location of these 1000 transformers is not available currently. The data is available for the same 33 months as dataset 1. The difference for dataset 2 is the weather information for all 1000 different locations for every hour is not available. Hence, the dataset 2 consists of features including energy consumption values, lag hour values, and season. Figure 21 presents the standard deviation and mean of the energy consumption values for a subset of transformers in Dataset 2. The high width of the boxes in Figure 21 depicts that the energy consumption values for transformers 21, 127, and 562 are highly varying. Also, it indicates that many records have zero values for consumption. The range and mean of energy consumption values in dataset 1 and dataset 2 are mentioned in Table 10.

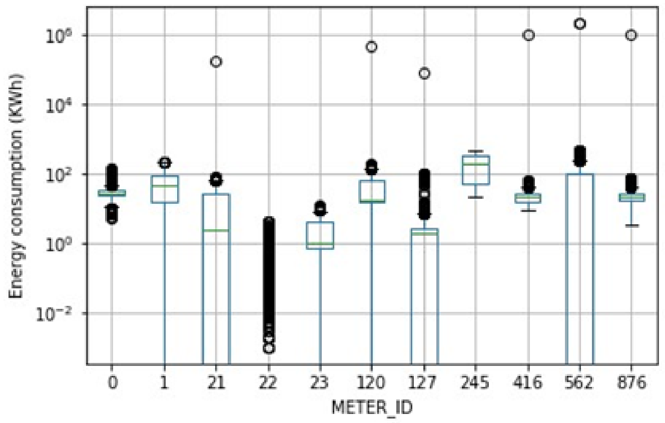


Figure 21 The standard deviation of energy consumption for different transformers in a sample of Dataset 2

Table 10 Descriptive statistics of datasets.

	Mean (KWh)	**Std (KWh)**	**Min (KWh)**	**Max (KWh)**
Dataset 1	75.25	111.87	0	754.60
Dataset 2	45.41	1346.71	0	2147483.64

5.1.3 Dataset 3

The dataset 3 is energy consumption data from 105,148 transformers in the distribution network spread across the country of Spain. The size of the dataset is around 250 GB and it contains 2.16 billion records of energy consumption.

5.2 Proposed Methodology

There are multiple implementation layers to the load forecasting application in smart grids. These layers are depicted in Figure 22 and are listed below:

- Feature Engineering Layer
- Clustering Layer
- Transfer Learning Layer
- Machine Learning Models
- Incremental Training Layer or Real-time Machine Learning

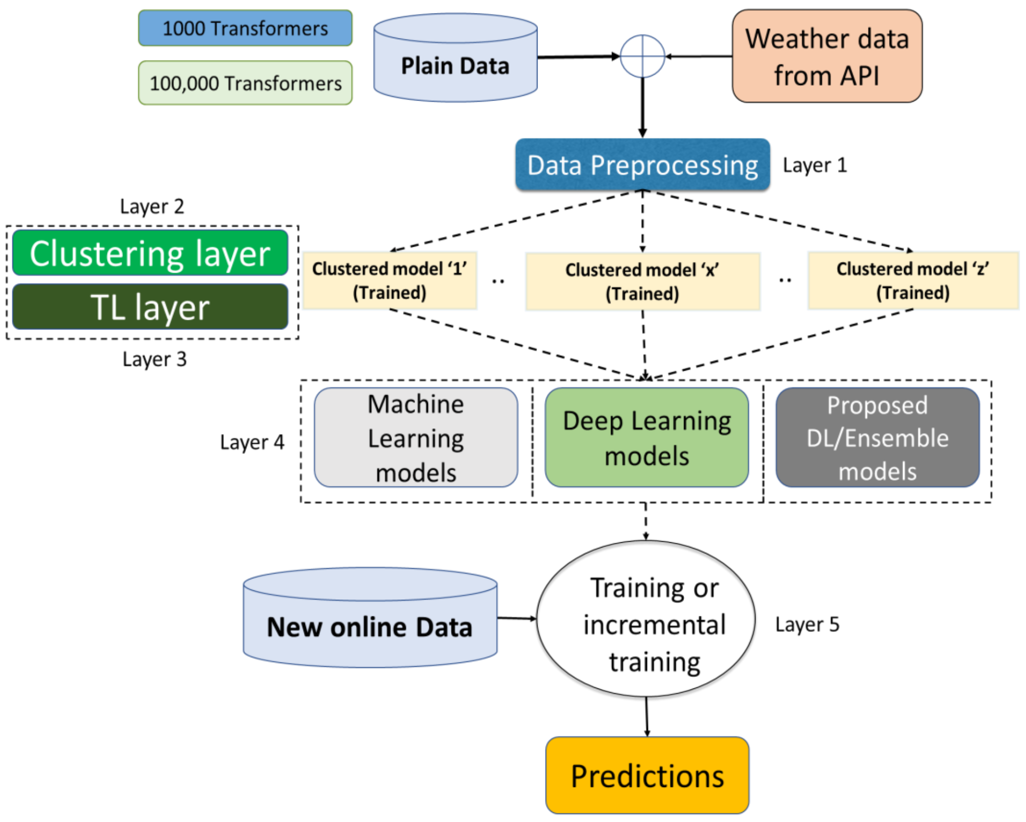


Figure 22 Different layers in the proposed methodology

The core layer is the machine learning models layer where the training and testing of the machine learning models happen on the load forecasting historical data. All the other layers are added on top of the machine learning models layer to enhance the performance of the forecasting. The enhancement may be in terms of accuracy of predictions or to reduce the number of prediction models to be developed for 100,000 distribution nodes or to reduce the training time of the models.

5.3 Feature Engineering Layer

This layer employs different feature extraction or dimensionality reduction techniques for the applications of short-term power forecasting using smart meters data. The number and data type of input features are crucial to the performance of power forecasting models. The input features in the dataset are weather parameters and lag hour values of energy. The performance of the machine learning models decreases with the increase in the number of input features. That is, the machine learning models tend to overfit, and the forecasting accuracy is reduced. The performance of the feature extraction or dimensionality reduction techniques has been evaluated in the context of the forecasting applications with models involving Artificial Neural Networks (ANN), Long Short Term Memory (LSTM), and Linear Regression (LR). While linear Principal Component Analysis (PCA) is a preferred dimensionality reduction technique for faster training times, kernel PCA, Non-negative Matrix Factorization (NMF), Independent Component Analysis (ICA), and Uniform Manifold Approximation and Projection (UMAP) yield better accuracies [152].

The feature extraction techniques evaluated in this work are described in the following:

5.3.1 Principal Component Analysis (PCA)

PCA transforms the features in a dataset into new features which are termed as Principal Components. The principal component is a linear combination of the original variables of the dataset. Principal components are ranked so that the first principal component is the one that represents the maximum variance in the data. The subsequent principal components represent the remaining variance in the dataset. However, these are not correlated to the first principal component of the data. Singular Value Decomposition (SVD) decomposes the original matrix into its components based on the concept of eigenvalues and eigenvectors, and this is used to remove the redundant features.

5.3.2 Independent Component Analysis (ICA)

ICA is set up on information theory and it differs from PCA in the sense that PCA finds the uncorrelated factors whereas ICA finds the independent factors. The uncorrelated factors are those which do not have any linear relationship between them. However, independence is an absolute characteristic.

5.3.3 Non-negative Matrix Factorization (NMF)

NMF is a multivariate analytic technique used for dimensionality reduction (DR). The NMF breaks down a non-negative data matrix into two non-negative matrices. One of the component matrices is termed as the basis vectors. These basis vectors are projected onto a lower-dimensional space to perform DR. The peculiarity of NMF over PCA and

SVD is that it has non-negativity constraints. NMF is an iterative method and has merits in algorithms using large matrices.

5.3.4 Manifold Isometric Mapping (ISOMAP)

It is a manifold projection-based DR technique. For any data, a manifold close to the data is located, and the projection of data on that manifold is calculated. Finally, for the representation, the manifold is unfolded to determine the representation of the original data onto a lower-dimensional space. ISOMAP is a method that focuses to retrieve full low-dimensional projection of a non-linear manifold which is presumed to be smooth.

5.3.5 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE recognizes the patterns in non-linear ways. It uses local approaches (mapping neighboring points on the manifold to neighboring points in the low dimensional data setting) and global approaches (preserving geometry at all scales, i.e. mapping neighboring points on a manifold to neighboring points and mapping far away points to faraway points in low-dimensional data setting) to map the data points into lower-dimensional data representation. It computes the probability similarity of points in high dimensional representation and low dimensional representation. Basically, it computes the Euclidean distances between points in high or low dimensional expanse and converts these distances to conditional probabilities to represent similarities. However, there are a few drawbacks such as loss of large-scale information, slower execution times, and inability to represent larger datasets for the t-SNE method. t-SNE performs efficiently when the dataset is not huge and there is non-linear dependency among the data features. Usually, GPU-accelerated implementations of t-SNE (such as Barnes-Hut approach, and RAPIDs) are employed to reduce the high processing times. However, the set of experiments in this work has been conducted only on the CPU implementations of the DR techniques.

5.3.6 Uniform Manifold Approximation and Projection (UMAP)

UMAP is a state-of-the-art DR technique that preserves much of local and global structure as compared to t-SNE. It uses the k-Nearest neighbor concept. It determines the span between the points in high-dimensional data representation and projects onto lower-dimensional data representation and employs Stochastic Gradient Descent (SGD) to reduce the distance in the lower dimensional setting. It has the following advantages such as the ability to handle large datasets, faster computation time when compared to t-SNE, and preservation of local and global structures of data.

5.4 Clustering Layer

This layer provides the gain in training time and performance in terms of accuracy when the clustering-based deep learning modeling is employed for load forecasting. For 100,000 distribution transformers, it is initially presumed that 100,000 machine learning models are required to make predictions at each transformer. However, by employing clustering methodology, the number of models to be developed can be reduced from 100,000 models to a lesser number 'k'.

The different forecasting models are generated for different clusters of load profiles. For clustering, k-Medoid based algorithm is employed. The clustering of the distribution transformers, based on the similarity in the energy consumption, improves the accuracy of the proposed methodology and reduces the number of models required for a large number of distribution transformers, consequently reducing the training time.

In current works, a deep neural network consisting of six layers and utilizing Adam optimization learning model using the TensorFlow framework has been employed. The case study has been successfully performed on real data of energy consumption at the distribution level for 1000 transformers in Spain Electricity Network. The results reveal that the proposed model has superior performance when compared to the state-of-the-art, and other classical load forecasting methodologies. The clustering-based approach improves accuracy by 0.005 to 2.9% and saves around 44% of training time using single-core processing compared to non-clustering models.

On 1000 transformers, the number of models to be developed has been reduced from 1000 to 93 models only. The number 'k' i.e. the number of clusters has been determined by utilization of the elbow curve which analyzes the within-cluster error to the number of clusters.

The clustering approach currently successfully tested on 1000 transformers, is to be tested on 100,000 transformers to prove the scalability of the proposed approach.

5.5 Transfer Learning Layer

This layer is applied on the clustered models with an aim to reduce the training time. The method of knowledge transfer from model 1 to other models enables the subsequent models to reach the convergence faster.

Also, if transfer learning is used between the transformers with similar energy consumption patterns, not only the training time but also the accuracy of the later models is improved.

Desired results have been obtained when transfer learning is utilized between transformers within the same clusters. However, there is an issue of local minima convergence when transfer learning is applied between different clusters of transformers. The aim of this layer is to employ a reliable transfer learning algorithm to use the knowledge from existing load forecasting machine learning models to innovatively develop highly accurate transfer learning models for cases of newly installed distribution nodes where the availability of load data is not sufficiently large. The work investigates how negative learning is avoided by transferring knowledge between similar and different distribution units. The overall results indicate that the knowledge transfer from developed models improves the accuracy of newer models, reduces the time of convergence to local minima, and reduces training time for deep learning models compared to that of models without transfer learning.

5.6 Machine Learning and Deep Learning models Layer

The modeling layer is the layer in which the offline training of the machine learning and deep learning models takes place. The modeling can be performed on encrypted data (if there is a security layer) or non-encrypted data (if there is no security layer). The primary work of this thesis focuses on the big data analytics and modeling on data without a security layer. However, our work that focused on the addition of security layer, where the homomorphic encryption of the data takes place before model training and the decryption of the predictions takes place after testing, is presented in Appendix A.

In this layer, the classical machine learning models such as Linear Regression, and Decision Trees, or the deep learning models such as Long Short-Term Memory (LSTM), and Deep Neural Networks (DNN) can be employed. The LR has been used as one of the prediction models to act as a benchmark for training time as this model would have the lowest training time owing to the simplicity of the model but coarser accuracy.

5.7 Modeling the Learning Process

5.7.1 Proposed Model Cost Function PULSE

5.7.1.1 Overview

Quadratic cost function such as Mean Squared Error (MSE) has been a widely used objective function whilst training deep neural networks for energy forecasting in Smart Grids. In this work, Penalizing Underestimation Logarithmic Square Error (PULSE), a novel objective function is proposed with an aim to reduce the tendency of deep learning models to underestimate the target variable.

Load forecasting is the foremost and crucial step in power system planning [205]. The system operators utilize the information of load forecasts to make decisions on generation resource management, economic dispatch, and scheduled maintenance. Inevitably, perfectly accurate demand forecasting directly influences the costing and reliability of the power systems. However, peak demand forecasting is crucial to prevent blackouts or loss of energy. Although the accuracy of demand forecasting is highly important, the underestimation of demand is more harmful than the overestimation of demand. This is the motivation of the proposed PULSE cost function. Excess load demand and shortage of supply can lead to unintentional brownouts and blackouts that have severe negative effect on the life of energy consumers [149].

The application of neural networks for forecasting load demand is to provide a black box of product-sum operations followed by activations to generate the non-linear mapping instead of formulating numerous mathematical derivations to explore the correlation and rules between input features and the output target variable.

The weight contributions of previous layer outputs onto the next layer inputs are determined during the learning process that involves the forward propagation and error backpropagation method. Conventionally, the error backpropagation is based on the error cost function that needs to be minimized using optimization algorithms. The cost function can be customized to penalize the underestimation of the target variable during the training process so that the networks are more prone to overestimate than to underestimate.

Currently, there are primarily five types of loss functions that evaluate the error in regression problems during the backpropagation method in neural networks: i) quadratic (L2) loss function such as mean squared error [206], ii) linear (L1) loss function such as mean absolute error [207], iii) Huber loss function [208], [209], iv) logarithmic cost function (logcosh) [209], and v) quantile loss function [210].

In [210], Ben et al. proposed a scheme to add a quantile constraint to any loss function of regression neural networks. Although quantile constraints do not yield any gradients, the authors formulated a method and an algorithm to minimize a generalized loss function. Their results presented that the cost function converges proving the feasibility of their proposed solution. The quadratic cost function has been adopted extensively in the deep learning methodologies for load demand forecasting owing to the least sum of squares postulate [174], [211]. In [174], Syed et al. utilized MSE as a loss function to develop clustering-based deep learning models on big data for load forecasting. The demerit of the quadratic cost function is the low rate of gradient descent and this increases training time especially in the case of big data processing. The problem is evident in artificial neural networks and is more pronounced in deep neural networks (DNNs) such as dense networks, convolutional neural networks, and recurrent neural networks.

In [212], Khosravi et al. proposed a weight decay cost function for decreasing the length of load forecasting intervals without affecting the coverage probability. Their results suggested that their proposed cost function overperformed the delta technique that is fundamentally utilized for constructing the load forecasting intervals instead of point-based forecasts.

To the best of my knowledge, there was no research work with a discussion on the elimination of underestimation of demand forecasting, and with proposed methodologies to reduce or eliminate the underestimation tendency of deep neural networks optimizing the quadratic, linear or quantile cost functions. The comprehensive results provide insights and investigate the models that are optimal keeping accuracy in consideration and compare these models with the proposed methodology that eliminates the tendency of underestimation. In this research, A custom cost function called Penalizing Underestimation Logarithmic Square Error (PULSE) is proposed with an aim of providing a tradeoff between high accuracy and removal of the underestimation tendency of deep learning models. A real case study based on a Portuguese load demand dataset of consecutive three years is used to investigate the efficiency of the proposed methodology that eliminates the tendency of underprediction of load demand.

5.7.1.2 PULSE Cost Function

Conventionally, the loss function (ℓ) utilized to minimize the error of predictions from machine learning models is mean absolute error, mean percentage error or mean squared error. The proposed PULSE cost function is based on the custom proposed loss function which is formulated as (37).

$$\ell = \left[log(y+1) - log(y_{pred}+1)\right]^2 + (\beta) * log(\psi(y_{pred} - y) + 1) \tag{37}$$

where y is the true value of the independent variable, y_{pred} is the predicted value of the independent variable, β is the hyperparameter (called penalty coefficient) in this loss function, and $\psi(.)$ is the proposed modified approximation function. β lies between 0 and 1 for normalized data. $\psi(.)$, the approximation function is given by the following equation:

$$\psi(x) = \begin{cases} -x, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{38}$$

The proposed approximation function ψ(.) can be reformulated as given in (39). It can aptly be called a negative ReLU function due to its analogous similarity to the ReLU

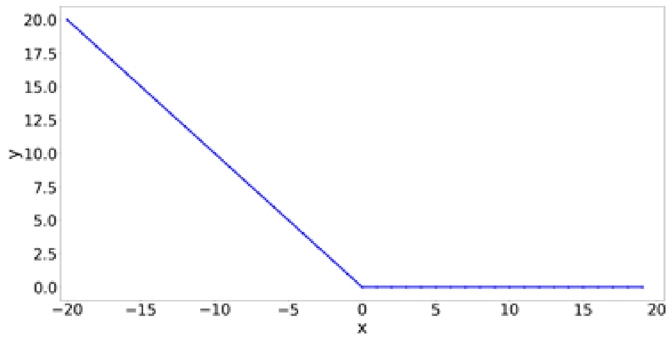


Figure 23 Proposed approximation function

activation function and is illustrated in Figure 23.

$$\psi(x) = -\,min(0, x)\ \forall\ x \tag{39}$$

The PULSE cost function (J) is therefore modified as given in (40).

$$\mathrm{J} = \frac{1}{2m}\sum_{i=1}^{m}\left[log(\mathrm{y_i}+1) - log\left(\mathrm{y_{i_{pred}}} + 1\right)\right]^2 + (\beta) * \frac{1}{2m}\sum_{i=1}^{m}\left[log\left(\psi\left(\mathrm{y_{i_{pred}}} - \mathrm{y_i}\right) + 1\right)\right]^2 \tag{40}$$

This can be rewritten as the following:

$$\mathrm{J} = \frac{1}{2m}\sum_{i=1}^{m}\left[log\left(\frac{\mathrm{y_i}+1}{\mathrm{y_{i_{pred}}} + 1}\right)\right]^2 + (\beta) * \frac{1}{2m}\sum_{i=1}^{m}\left[log\left(\psi\left(\mathrm{y_{i_{pred}}} - \mathrm{y_i}\right) + 1\right)\right]^2 \tag{41}$$

Figure 24 illustrates the backward propagation of deep neural networks. The backward propagation relies on the cost function to optimize the accuracy of prediction models updating weights in each epoch of an iteration.

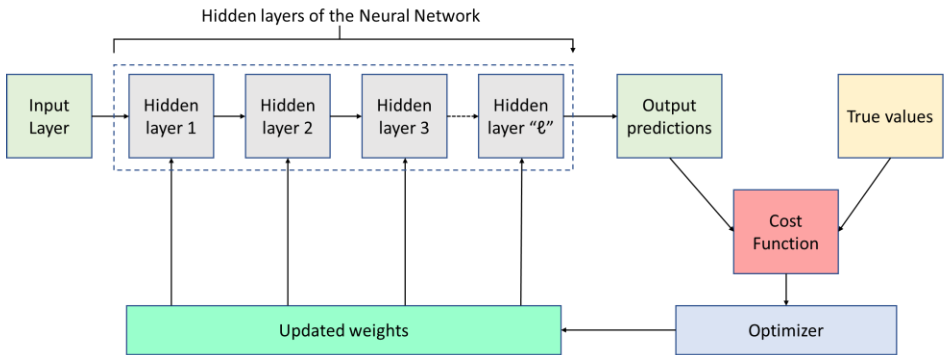


Figure 24 Backward propagation of deep neural networks

5.7.1.3 Case study for performance evaluation of PULSE

5.7.1.3.1 Data description

In this case study, an open-source real-world electricity load diagrams dataset is utilized for performance validation of the proposed cost function and methodology. The data contain electricity consumption information of 370 customers and cover 140256 electricity consumption records for each customer [213]. In total, the records amount to around 51.89 million records for all customers and before processing, the dataset occupies around 694.33 MB memory on the compute node.

5.7.1.3.2 Data processing

The dataset does not contain any missing values. Therefore, imputation methods were not adopted in preprocessing step [41]. The energy consumption values were recorded in kW every 15 mins. The values have been aggregated to hourly consumption values expressed in kWh.

Especially with neural network models, it is crucial to scale and normalize the features to a similar range [214]. Without feature scaling, convex optimization methods such as gradient descent take a long time to converge. Besides, normalization is performed so that each feature has approximately zero mean. The application of both feature scaling and mean normalization is formulated as (42).

$$x_j^{(i)\prime} = \frac{x_j^{(i)} - avg(x_j)}{max(x_j) - min(x_j)} \tag{42}$$

where $x_j^{(i)\prime}$ is the scaled and normalized value of a feature in row i, $x_j^{(i)}$ is the original value of a feature in row i, j represents a feature number, $avg(x_j)$ refers to the average value of the feature x_j. Similarly, $max(x_j)\ and\ min(x_j)$ refer to the maximum and minimum of the feature respectively.

5.7.1.3.3 Sliding Window Method

The data are fundamentally time series data of load demand. Additionally, the datetime extracted features referring to the day, month, year, and day of the month are considered. Furthermore, the sliding time window approach is deployed to integrate the lag values of load demand as feedback to the input layer of network models. This is based on the fact that the load demand in the prior time may have a crucial influence on load demand at the current time. The fixed-length sliding time window enables the model networks to analyze the demand history to extract time-varying features. Suppose $y(t);\ t = 1, 2, 3, 4, \ldots, n$ be the load demand time series where t is the index of time and n denotes the total number of time instances. The sliding window method assigns $\boldsymbol{\tau}$ lagged load demand values to each instance of time series at entry t. A $\boldsymbol{\tau}$x1 feature vector denoted by $[y(t-\tau), \ldots y(t-4), y(t-3), y(t-2), y(t-1)]$ is assigned to a typical instance of time series $y(t)$. The addition of extracted datetime features and the deployment of the sliding window method transforms the load demand time series data into a supervised machine learning problem.

5.7.1.3.4 Model development – LSTMs

For forecasting time-dependent target variables, recurrent neural networks (RNN) have been developed. These networks have memory cells that can retain information of the prior captured states of the input to make a forecast for a future time step. Long Short-Term Memory (LSTM) is a special type of RNNs that can handle long-term dependencies and additionally address the problem of exploding and vanishing gradients using the forget and memory cells that decide on which information to forgo and which information to carry forward or retain respectively. The explicit handling of temporal dependence between records while learning a mapping function from input variables to target variables is offered by RNNs such as LSTMs, unlike DNNs and Convolutional Neural Networks (CNNs). The theoretical perspective of LSTMs is illustrated in Chapter IV.

5.7.1.3.5 Simulation Setup

The simulations were performed for short-term hourly load demand forecasting. To ensure a fair assessment of the impact of formulating cost function i.e., the use of proposed PULSE cost function, the same built stacked LSTM network is employed along with sliding window method, regularization components, and batch normalization. The comparative investigation is performed for a problem of point-based forecasting. Additionally, the results of simulations are compared against the models developed in the literature. It is significant to note that the aim of this methodology is not just the supremacy of accuracy but to balance the tradeoff between accuracy and the avoidance of underestimation tendency. The configuration and specifications of the machine utilized for simulations are Intel® Xeon® CPU E5-2670 @ 2.60 Hz, 16 cores, 32 virtual processors, and 88.0 GB RAM. Python programming was employed, Keras API was utilized to build the deep learning models, and TensorFlow served as a backend for Keras.

5.7.1.3.6 Convergence performance of PULSE

The neural network model learning occurs by minimizing a cost function. Various optimization algorithms such as Gradient Descent, Normal Equations, and Adam can be utilized to determine a local or global minimum of a function [149]. The optimization algorithm enables the model to learn the gradients in the direction of minimization of error i.e., ultimately minimizing the defined cost function. A deterministic method of proper

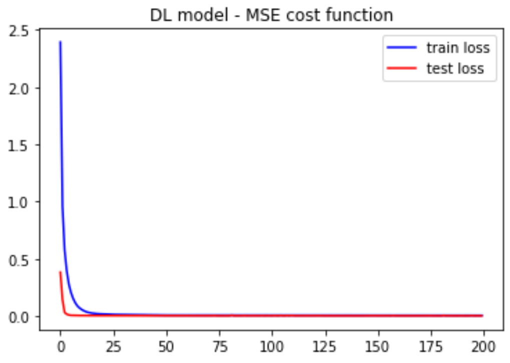


Figure 25 Convergence of widely used cost function (MSE)

learning using cost function is to plot a graph between the cost function value or error against the number of epochs of training. As the number of epochs iterates, it is ideal that the model converges after a particular epoch number yielding little or zero changes to the loss value going further. The variation of the cost function for a widely utilized MSE and the proposed cost function are illustrated in Figure 25 and Figure 26 respectively. The horizontal axes refer to the number of epochs during the learning process, and the vertical axes refer to the value of the cost function. As shown in Figure 26, the cost function value decreases logarithmically during the initial epochs and after 30 epochs, the cost function has converged to a minimum. From Figure 25 and Figure 26, it is evident that the models converged reasonably quickly, and both train and test performance remained similar in

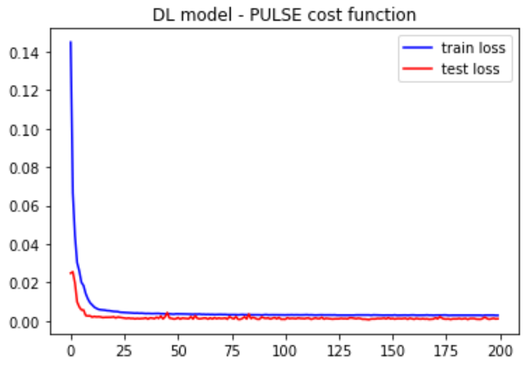


Figure 26 Convergence of proposed PULSE cost function

either of the cases. The performance and convergence behavior suggest that the proposed cost function is a good match for the learning process.

5.7.1.3.7 Simulation Results

The internal architecture and the optimized hyperparameters of the adopted deep learning model are presented in Table 11. Random search method is utilized for the optimization of hyperparameters. It has been empirically proven to work faster than the grid search method to discover precise values for the significant hyperparameters [215].

Table 11 Optimized Hyperparameters.

Parameter	**Utilized Model**
Number of layers	5 Stacked LSTM layers + 5 Dropout Layers + 5 Batch Normalization layers + 1 Dense layer
Number of neurons	{260, 210, 160, 50, 5, 1}
Dropout Rate	{0.10, 0.20, 0.30, 0.30, 0.30}
Number of Epochs	200
Batch size	128

Figure 27 illustrates the multiple plots of load demand predictions against the actual load demand for a subset of energy customers. The independent axes represent the time indices in hours and the dependent axes represent the inverse scaled hourly load demand in kWh. The results illustrate that the predicted load demand is never underestimated using the proposed PULSE loss function. The accuracy of the MSE loss function may or may not be greater than the PULSE loss function but it is evident that the model with MSE loss function is prone to the tendency of underestimation which can be

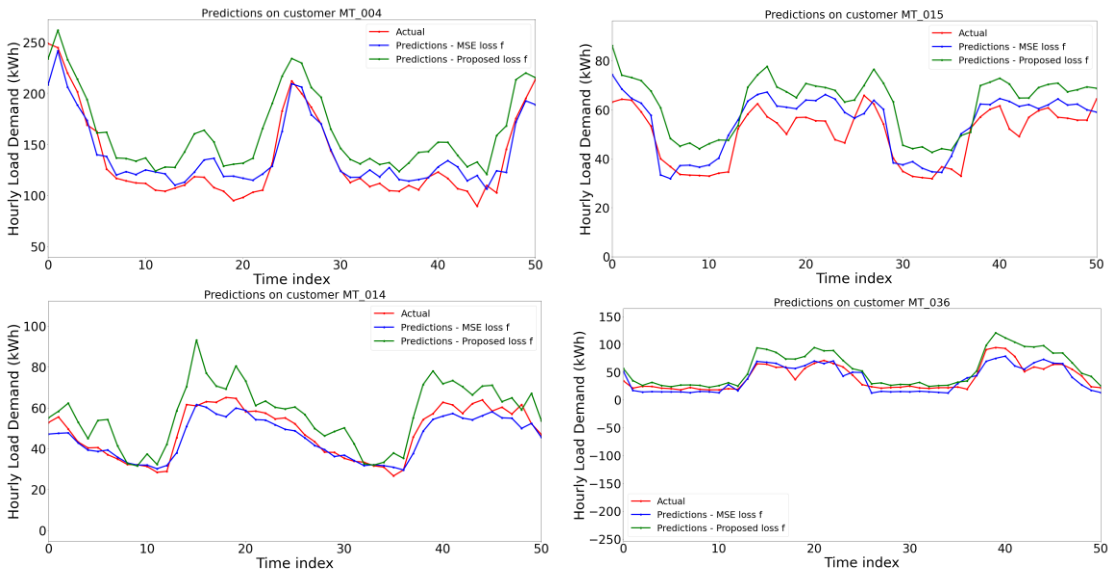


Figure 27 Load Demand Forecasting Results using MSE and PULSE cost functions on Deep Learning Models

clearly prevented using the proposed PULSE function. The results cannot be quantified using the performance graphs of a subset of consumers. Hence, the evaluation metrics are utilized, and the results are tabulated as shown in Table 12. The PULSE function based deep learning model yields an nRMSE = 7.106x10^{-2}, RMSE = 4.610x10^{-2} kWh, MAE= 3.267x10^{-2} kWh on the normalized and scaled load demand data, whilst the MAPE value is 10.857 % which indicates a superior accuracy of the model. The MSE function based deep learning model yields an nRMSE = 6.671x10^{-2}, RMSE = 4.555x10^{-2} kWh, MAE = 3.271x10^{-2} kWh, whilst the MAPE value is 35.066% which indicates a high accuracy of the model. The aim of the PULSE function is achieved as the model based on it does not underestimate over the testing dataset, and it definitely reduces the tendency of the deep learning models to underestimate the target variable.

Table 12 Evaluation Results.

Results on scaled load demand data	**MSE cost function**	**Proposed PULSE cost function**

nRMSE (10^{-2})	6.671	7.106
RMSE (10^{-2} kWh)	4.378	4.490
MAE (10^{-2} kWh)	3.231	3.267
MAPE (%)	35.066	10.857
Train time per model (s)	1326.209	1470.010

The performance of the deep learning models based on PULSE cost function is compared against the state-of-the-art models developed on the same dataset from literature and is presented in Table 13. As the results suggest, the proposed methodology performs better than ELM, RNN, LR, kSR, kNNR, GPR, and GRNN in terms of average RMSE error. There is RELM that has lower average RMSE when compared to the proposed methodology. It is crucial to note that the proposed methodology has an added advantage of avoiding underestimation tendency of deep learning models whilst predicting load demand.

Table 13 Performance evaluation against the state-of-the-art models from literature for arbitrary-ordered data.

Model	RMSE (10^{-2} kWh)
Extreme Learning Machine (ELM) [216]	6.47
Recurrent Neural Network (RNN) [216]	4.68
Linear Regressor (LR) [216]	6.46
k-smooth regressor (kSR) [216]	6.49
k-nearest neighborhood regressor (kNNR) [216]	7.32

Gaussian process regressor (GPR) [216]	6.45
Generalized regression neural network (GRNN) [216]	6.45
Recurrent extreme learning machine (RELM) [216]	3.94
DL Model built in this work	4.378
Proposed PULSE cost function-based model	4.490

5.7.1.3.8 Insights

With the proposed novel cost function PULSE, the deep learning models can optimize the convergence of error to a minimum whilst penalizing the tendency to underestimate the target variable. The investigated model consists of stacked long short-term memory networks that employ sliding window method for better handling of long-term dependencies, batch normalization, and dropout techniques to eliminate overfitting of the investigated model based on the proposed cost function. The simulated results indicate that the proposed PULSE cost function provides competitive accuracy and avoids underestimation of load demand with a superior training speed. This cost function can be applied to short-term, medium-term, and long-term load forecasting and its characteristic estimation behavior can enable the utilities to manage the generation resources properly to avoid any supply shortage.

5.7.2 Model Optimizer

Model optimizer is an algorithm that minimizes the cost function in the process of learning optimum values of weights and biases. Among the widely used model optimizers, the optimizers such as LBFGS and Adam has been utilized during model development.

Additionally, the decoupled weight regularized invariant of Adam has been proposed to break off from the local optima during transfer learning and the detailed explanation is provided in Section 6.2.2.3.

5.7.2.1 Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (LBFGS)

LBFGS is a limited memory quasi-Newton optimization solver [217]. For an optimization problem with n variables, BFGS needs $n\,x\,n$ matrix to approximate the hessian while LBFGS only needs a small number of vectors of length n to approximate the hessian. It does so by maintaining the knowledge of the historical states of previous m updates of current position and its gradient $g = \nabla f(x)$. The historical states are maintained as last m updates of the position difference and gradient difference. Each of these variables is a vector of length n. These $2m$ variables and the original gradient will be used in finding a new direction. The original description of LBFGS is given in [218].

5.7.2.2 Adaptive Moment Estimation (Adam) Optimizer

Adam algorithm puts together gradient descent of RMSProp and gradient descent with Momentum for optimization. Adam algorithm has been proven to be very effective for different neural networks of a very wide variety of architecture. The Adam Optimization steps are illustrated in Figure 28 and are detailed as follows.

1) Hyperparameters in this algorithm are learning rate, decay rates ρ_1 and ρ_2, and δ. Usually, the learning rate ϵ is the only term that is to be tuned. Whereas the rates ρ_1, ρ_2, and δ are not required to be tuned and their default values are used. Usually, $\delta = 10^{-8}$, $\rho_1 = 0.9$ and $\rho_2 = 0.999$. ρ_1 is the weighted average, and ρ_2 is the weighted average of the squares.

2) First step is to initiate the moment variables s and r to 0. At this point, time step is 0.

3) Until the stopping criteria is met, the following steps are performed.

4) A sample data point is selected from the training data.

5) Initially, the weights θ are calculated using current mini-batch gradient descent.

6) Momentum exponential weighted average is updated as $s \leftarrow \rho_1 s + (1 - \rho_1)\hat{g}$. This is momentum like update with hyperparameter ρ_1.

7) Similarly, the RMS prop update is calculated as $r \leftarrow \rho_2 r + (1 - \rho_2)\hat{g} \odot \hat{g}$. $\hat{g} \odot \hat{g}$ is the element wise squaring of the gradients $\hat{g}$. This is RMSprop like update with hyperparameter ρ_2.

8) In the typical implementation of Adam, the bias correction is performed as $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}, \hat{r} \leftarrow \frac{r}{1-\rho_2^t}$. t is the iteration number.

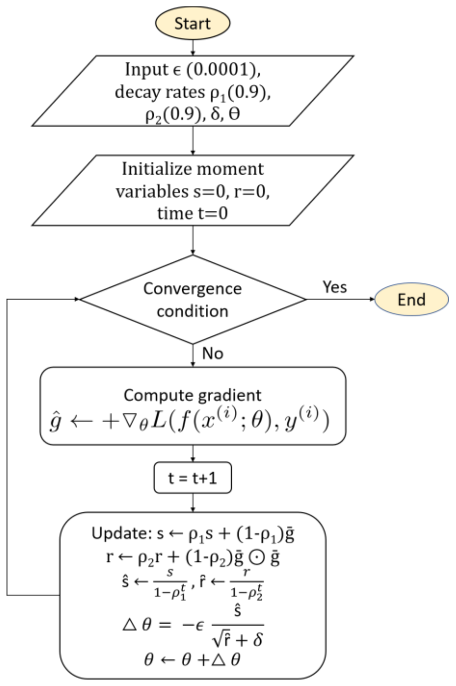

Start
Input ϵ (0.0001), decay rates ρ1(0.9), ρ2(0.9), δ, θ
Initialize moment variables s=0, r=0, time t=0
Convergence condition
Yes
End
No
Compute gradient $\hat{g} \leftarrow +\nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$
t = t+1
Update: $s \leftarrow \rho_1 s + (1-\rho_1)\bar{g}$
$r \leftarrow \rho_2 r + (1-\rho_2)\bar{g} \odot \bar{g}$
$\hat{s} \leftarrow \frac{s}{1-\rho_1^t}, \hat{r} \leftarrow \frac{r}{1-\rho_2^t}$
$\triangle\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$
$\theta \leftarrow \theta + \triangle\theta$

Figure 28 Adam Optimization

9) Finally, the θ is updated as the formula $\theta \leftarrow \theta + -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$. Here the numerator in $\nabla\theta$ is due to the moment operation and the denominator is due to the gradient descent of the RMSprop or bias correction. This brings the end to iteration.
10) Make note of the points that the values of $\rho 1, \rho 2, \delta$ can be default values and only the learning rate ϵ is to be tuned for better performance.

5.7.3 Batch Normalization

Traditionally, the inputs to the neural networks are scaled and normalized to a range between 0 and 1. This type of normalization is generally efficient for the machine learning model performance. In the current research, batch normalization is additionally adopted to support dropout layer in removing overfitting of the neural network and deep learning models. In batch normalization, not just the inputs to the input layer but also the inputs to all the hidden layers are normalized. Batch normalization utilizes the mean and the standard deviation of the batch of data and not of entire data unlike the normalization of the input layer. The merits of batch normalization include the improvement of the accuracy performance of deep learning models and the reduction of overfitting. It is recommended to utilize batch normalization in every layer and alongside dropout.

5.7.4 Dropout

Artificial Neural Networks and Deep Learning Networks are prone to overfitting which corresponds to having high variance. Overfitting causes the models to have high training accuracy but perform with poor accuracy on testing data. The most common technique to eliminate overfitting in neural networks is called Dropout. In each iteration, dropout removes some neurons randomly on the defined layers as shown in Figure 29.

Dropout essentially trains different neural networks. That is, in each iteration different network is trained. At testing time, the combination of the different networks is generated by averaging the results. It is well known as a generalization that averaging multiple machine learning models reduces overfitting.

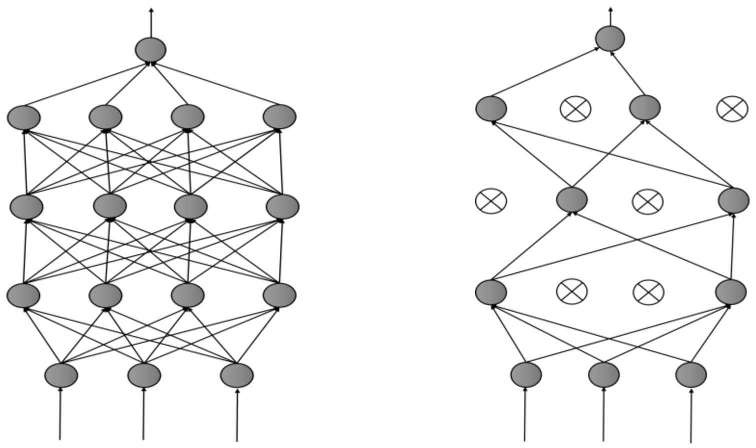


Figure 29 Standard Fully connected Neural Networks and Neural Networks with Dropout

5.8 Incremental Learning Layer

This layer enables the proposed framework to perform online machine learning. This allows the deep learning models to use the data points that arrive at the current time and in the future to incrementally retrain the already trained machine learning models. The online training of trained models using new data points is termed incremental learning. Incremental learning is analogous to transfer learning. The historical data in incremental learning is similar to the dataset of application 1 in transfer learning and the newer data points in incremental learning are similar to the dataset of application 2 in transfer learning.

Let $L_{k,t}$ be the load of transformer k at time duration t. The load matrix for the transformers is represented by $L_{K\times T}$ (Equation (43)). The size of the matrix increases with the increase in the number of transformers or with the reduction in the aggregation level of energy consumption values. For the 1000 transformers dataset that is used in this work, the size of the load matrix is 1000x1001 where 1001 is the number of days between 01 January 2017 to 28 September 2019.

$$L_{K\times T} = \begin{bmatrix} L_{1,1} & L_{1,2} & \cdots & \cdots & L_{1,t-1} & L_{1,t} \\ L_{2,1} & L_{2,2} & \cdots & \cdots & L_{2,t-1} & L_{2,t-1} \\ \vdots & \vdots & \cdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots & \vdots \\ L_{k-1,1} & L_{k-1,2} & \cdots & \cdots & L_{k-1,t-1} & L_{k-1,t} \\ L_{k,1} & L_{k,2} & \cdots & \cdots & L_{k,t-1} & L_{k,t} \end{bmatrix}, \tag{43}$$

The similarity between any two transformers k, m at any given time p is determined based on pairwise Minkowski similarity as given by Equation (44).

$$D_{k,m,p} = \left(\sum_{t=1}^{24} \left|L_{k,t} - L_{m,t}\right|^{q}\right)^{\frac{1}{q}}, \tag{44}$$

where L_k, L_m represent the row vectors of load values for transformers k, m respectively. The optimized value of q in Equation (44) was determined to be equal to unity.

Finally, the obtained distance matrix is passed as an argument to the clustering function to obtain the clusters of transformers with similar energy consumption patterns. The adoption of Minkowski similarity enhanced the performance of clustering.

Algorithm 5 presents the sequence of steps performed for incremental learning.

Algorithm 5 Incremental Learning Framework

```
Input Data $S_{t-1}$ with n rows: $\{x_i,y_i\}$ where i represents the row number (i=1, 2,
3, ..., n).

Initialize k-medoid clustering algorithm to generate k number of clustered
models which provides distribution function $W^{t-1}$.

A hypothesis function $h^{t-1}$ is generated at time t-1 along with the distribution
fuction $W^{t-1}$. And, $W^{t-1}$ = [$w_1$, $w_2$, . . . ., $w_m$]; m represents the number of
layers in the neural networks.

while new data $S'_{t+k}$ is available with n' instances. do
    Hypothesis h' is updated using new data $S'_{t+k}$
    $W^{t+k}$ = [$w_1$, $w_2$, . . . ., $w_m$] is updated using new data $S'_{t+k}$ as the
following:
    for t = t, t+1, t+2, . . . . , t+k do
            Forward propagation: compute all $x_j^l$
            Backward propagation: compute all $\delta_j^l$
            update the weight: $w_j^{(l)} \leftarrow w_j^{(l)}$ – η $x_i^{l-1}$ $\delta_j^l$
            iterate to the next step until it is time to stop
    end for
    return the final weights $w_{ij}^{(l)}$ as $W^{t+k}$
end while
```

CHAPTER VI

PERFORMANCE ANALYSIS*[10]

6.1 Performance Evaluation Metrics

The metrics of evaluation used for accuracy are Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Training time and testing time are used to evaluate the performance in terms of execution time.

1) Root Mean Square Error (RMSE): RMSE is the square root of the sum of squares of the difference between actual and predicted energy consumption. RMSE is an effective performance metric for comparing forecasting errors of different models for a single attribute which is the case in this research work. However, it is not a recommended measure to compare performance between attributes as RMSE is scale dependent. RMSE is given by (45) [219].

$$RMSE = \sqrt{\frac{1}{N}\sum_{r=1}^{N}(E_r' - E_r)^2}, \tag{45}$$

2) Mean Absolute Percentage Error (MAPE): MAPE represents the ratio of the absolute difference between the actual and predicted value to the actual value at every record of energy consumption. It is necessary to make sure that the actual value is not zero while calculating MAPE. MAPE is given by (46) [219].

$$MAPE(\%) = \frac{100}{N}\sum_{r=1}^{N}\left|\frac{E_r' - E_r}{E_r}\right|, \tag{46}$$

[10] Reprinted with permission from "Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid with Clustering and Consumption Pattern Recognition." by Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Shady S. Refaat, Mahdi Houchati, Othmane Bouhali, and Santiago Bañales, 2021. IEEE Access 9, 54992-55008, Copyright 2021 by Dabeeruddin Syed.

For low prediction values, the MAPE value cannot exceed 100%. However, for high prediction values, there is no maximum control limit to the value of MAPE.

3) nRMSE:

$$nRMSE = \frac{\sqrt{\frac{1}{N}\sum_{r=1}^{N}(E'_r - E_r)^2}}{E_{max} - E_{min}} \tag{47}$$

4) MAE:

$$\text{MAE} = \frac{\sum_{\text{r}=1}^{\text{N}}|E'_r - E_r|}{\text{N}} \tag{48}$$

where E_r is the actual energy consumption at instant $t = r$, E'_r is the predicted value of energy consumption at instant $t = r$, and N denotes the total number of testing samples.

6.2 Experimental Results

The hardware specifications of the supercomputers utilized in the case study are described in Table 14.

Table 14 Hardware Specification of the experimental setup.

Hardware Specification	**Value**
Nodes	1-16
Interconnect	Onmi-Path
CPU Architecture	Intel Broadwell x86_64 CPU operating at 2.4 GHz
CPU cores	8 per node
Memory	16 GB RAM per node
Job scheduler	Slurm

6.2.1 Clustering

Figure 30 represents the sequence of steps performed for developing the clustering-based short-term load forecasting model. As shown in the figure, the proposed methodology is carried out in four main stages:

A. Data acquisition and pre-processing stage
B. Clustering stage
C. Training stage
D. Testing stage

6.2.1.1 Elbow Curve

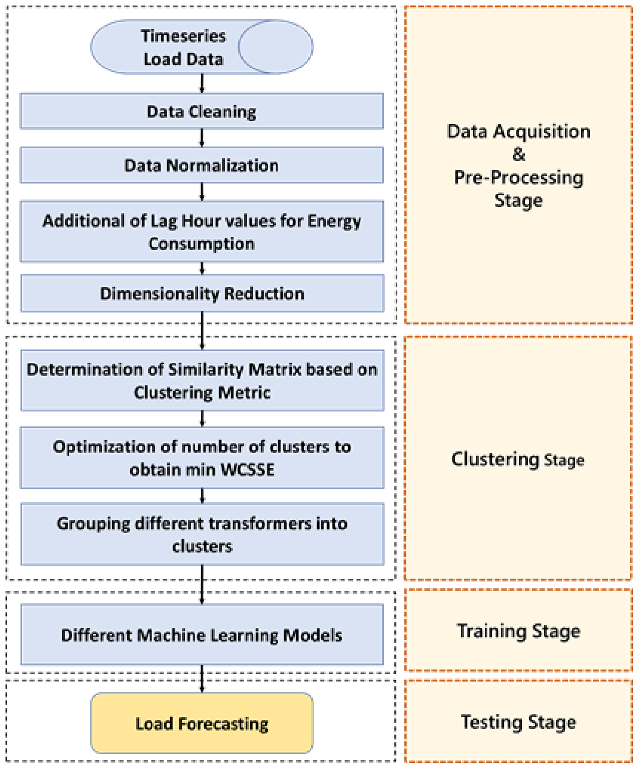

Timeseries Load Data
Data Cleaning
Data Normalization
Additional of Lag Hour values for Energy Consumption
Dimensionality Reduction
Data Acquisition & Pre-Processing Stage
Determination of Similarity Matrix based on Clustering Metric
Optimization of number of clusters to obtain min WCSSE
Grouping different transformers into clusters
Clustering Stage
Different Machine Learning Models
Training Stage
Load Forecasting
Testing Stage

Figure 30 Clustering Layer Steps

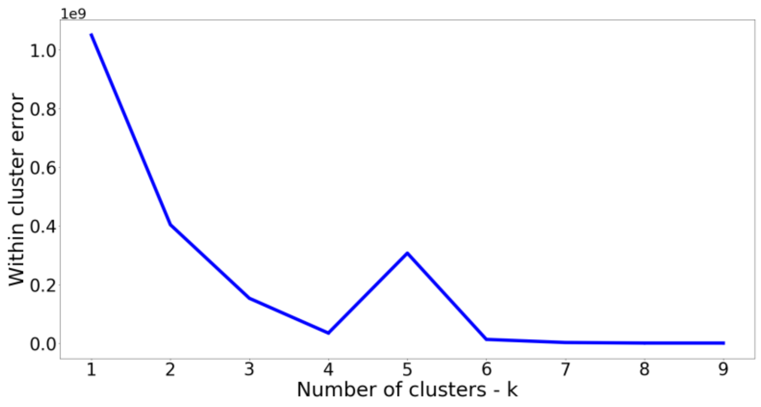


Figure 31 Elbow Curve for 10 transformers dataset

In this work, the direct method of the elbow curve is utilized. The elbow curve calculates the within-cluster sum of square errors (WCSSE) and determines the k-value such that WCSSE is minimized. The aim of the selection of k is to determine a low value of k such that the sum of square error for that value of k is the minimum and if any more clusters are added, the clustering is not improved much. This is to provide a trade-off between the number of clusters and the accuracy. The elbow method is selected over other methods of determining the k-value for clustering because of its simple complexity. As per existing research [220], the execution time is the lowest for the elbow method when compared to other methods owing to its low complexity of utilizing the sum of the square distance between cluster points and representative centers.

To determine the optimal number of clusters, the elbow curves are obtained for dataset 1 and dataset 2 as illustrated in Figure 31 and Figure 32. The independent axes in the figures indicate the number of clusters and the dependent axes in the figures represent the WCSSE for the corresponding number of clusters (k) value. As per Figure 31, the sharp decline in the WCSSE is observed for $k = 3$. Hence, the optimum number of clusters is selected as 3 for 10 transformers dataset. The elbow in Figure 32 suggests that the optimal number of clusters is k=93 for the 1000 transformers dataset. Hence, the

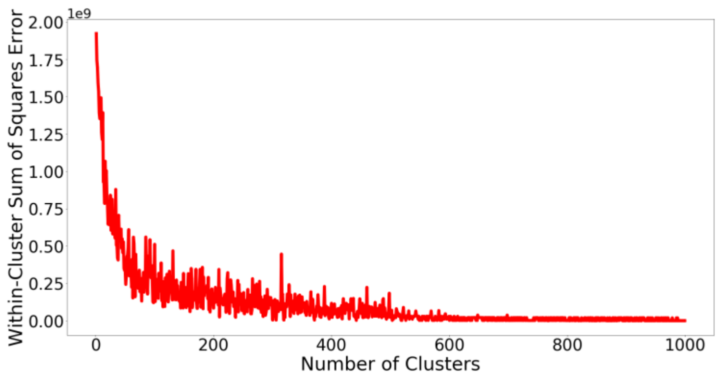


Figure 32 Elbow Curve for 1000 transformers dataset

clusters are determined, and the deep learning models are developed with the number of clusters k=93.

In this layer, the K-Medoid clustering is utilized to cluster similar transformers together. The similarity indicates that transformers have similar patterns of aggregated daily consumption and hourly consumption. The aim of the work is to evaluate the performance of individual models for 1000 transformers against the clustered models. Individual models mean that 1000 transformers have separately trained models using their individual data i.e. each transformer has a separate trained model built on its data. The clustered models indicate that the 1000 transformers are clustered into 'k' different groups and each of these 'k' clusters have one separate model trained on the data of the transformers within the clusters. The employment of a clustering technique reduces the required number of forecasting models from 1000 to k for 1000 transformers. As described in the previous subsection, the value of 'k' (number of clusters) is optimized to minimize the within-cluster sum of square errors.

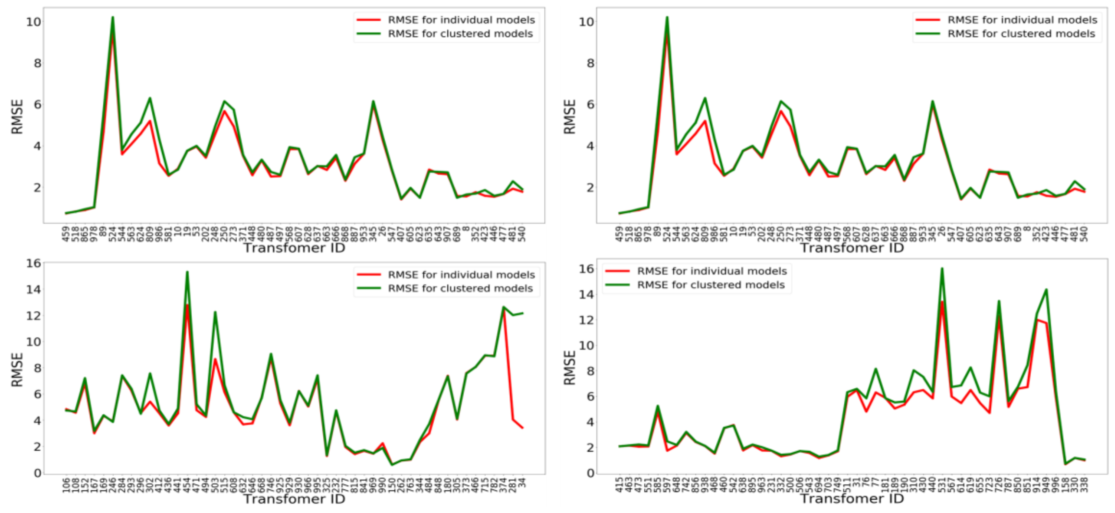


Figure 33 Performance Evaluation (in RMSE) of Clustered Models v/s Individual Models for different Transformers

The performance of clustered and individual forecasting models for distribution transformers is evaluated in terms of RMSE, MAPE, training time, and testing time.

The RMSE and MAPE values for individual models and clustered models using Deep Neural Networks (DNNs) are determined and these are depicted in Figure 33 and Figure 34.

Figure 33 indicates the results of the DNN models for load forecasting. Each of the subfigures indicates a representative subset of 1000 transformers. As observed from the RMSE lines, mostly the individual models represent the lower boundary of the two lines. The RMSE values range between 0 and 30 kWh. These values are very low considering the range (0 to 2,147,484 kWh) of energy consumption in the dataset. At a few points, the clustered models over perform the individual models for the respective transformers. The MAPE values for the individual models range between 4 to 16 percent and the MAPE for clustered models range between 5 to 19 percent. These MAPE values

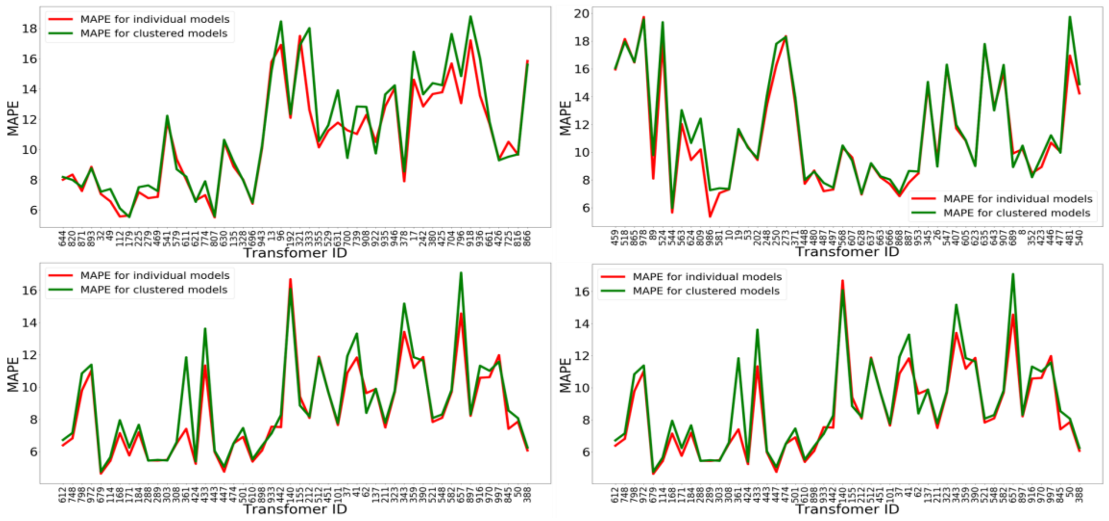


Figure 34 Performance Evaluation (in MAPE) of Clustered Models v/s Individual Models for different Transformers

indicate that the clustered models are very comparable to the individual models. A few transformers exhibit high statistical variance in the energy consumption, i.e., they have either zero consumption values, or very high energy consumption values, or actual energy values range between 0 and 1. The MAPE values for such transformers are around 20-32%. These transformers have been found to be alternate backup transformers that are used only during the periods of faults, preventive, or predictive maintenance of main transformers.

Table 15 presents the results of clustering and individual models on 10 transformers dataset when the machine learning models used are LR, LSTM, and DNNs. When accuracy is considered, the best performing model is the DNN model. In the clustering-based algorithm, the models are trained on a cluster whilst the testing is performed on each transformer within the cluster. If the clustering and individual models are compared, the individual models have slightly better accuracy when compared to clustered models. However, the accuracy of clustered models is highly competitive. If the gain on training time is considered, then the clustered models are highly preferable to individual models. When the training times for different machine learning models are considered, LR is the best owing to its simplicity. The DNN models have 10 folds of lesser training times compared to LSTM models. As a trade-off between accuracy and training time, it can be concluded that the clustering based DNN models perform better. A similar pattern is also recognized in Table 16. It depicts the results of clustering and individual models on a 1000 transformers dataset when LR, LSTM, and DNNs are used for training and testing.

Table 15 Results on 10 transformers dataset.

Model	**Mean RMSE (kWh)**	**Mean MAPE (%)**	**Training Time (s)**	**Testing Time (s)**
LR non-clustered	12.27	28.82	0.0656	0.0076
LR + clustering	13.25	32.23	0.0525	0.0076
ARIMA non-clustered	12.6305	30.2368	1.7182	1.8768
ARIMA + Clustering	14.2755	31.6923	1.0611	1.0047
LSTM non-clustered	2.2087	19.0902	421.00	3.7731
LSTM + clustering	3.1301	21.6020	118.83	0.2520
DNN non-clustered	2.3769	14.6451	14.63	0.0710
DNN + clustering	2.6874	15.9380	10.76	0.1070

Table 16 Results on 1000 transformers dataset.

Model	**Mean RMSE (kWh)**	**Mean MAPE (%)**	**Training Time (s)**	**Testing Time (s)**

LR non-clustered	54.0449	20.3235	14.06	0.89
LR + clustering	62.3503	20.8479	17.35	1.07
ARIMA non-clustered	59.4144	37.7886	513.5092	14.7560
ARIMA + Clustering	67.8725	39.4502	317.1253	8.6468
LSTM non-clustered	22.52	7.27	113432 = 31 hr	378.86 = 6 min
LSTM + clustering	37.06	11.10	29106 = 8 hr	110.41 = 1.8 min
DNN non-clustered	19.82	7.18	8409 = 2.33 hr	6.08
DNN + clustering	21.25	7.22	4644 = 1.29 hr	4.57

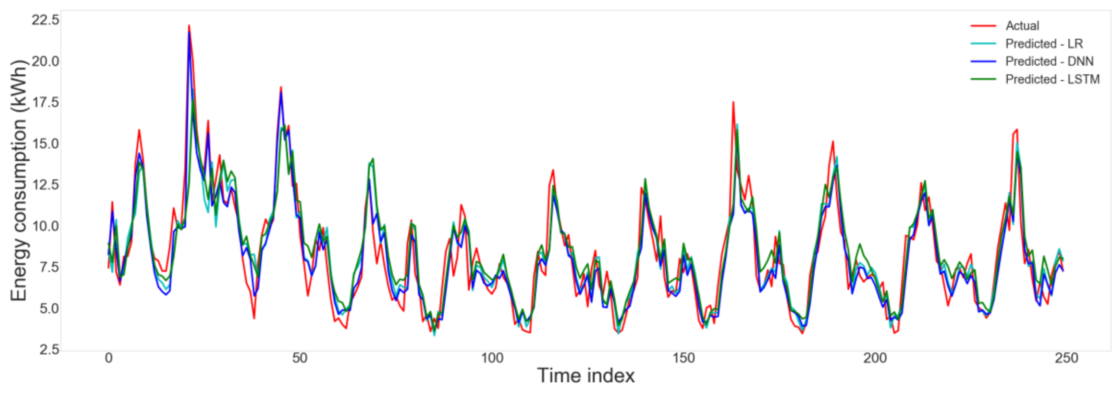


Figure 35 Forecast comparison of a trained clustered STLF model using different machine learning algorithms

The comparison of a trained clustered STLF model using different machine learning algorithms is illustrated in Figure 35. The independent axis represents the time points, and the dependent axis represents the energy consumption in kWh. The results in the figure denote that the proposed k-Medoids methodology has generated accurate clusters, and the clustered model predicts energy consumption values close to the actual values of consumption for all machine learning algorithms in general. Figure 35 also indicates that the DNN forecasts follow the consumption peaks better than LSTM and LR models. LSTM and LR at many time points forecast peaks after the peaks have occurred.

Figure 36 illustrates the error bars that depict the standard deviation of predictions using DNN and LSTM-based clustering models for STLF. The shaded region around the blue line depicting predicted energy values using clustered DNN model represents the error region or the deviation of model predictions. The experiments were repeated a

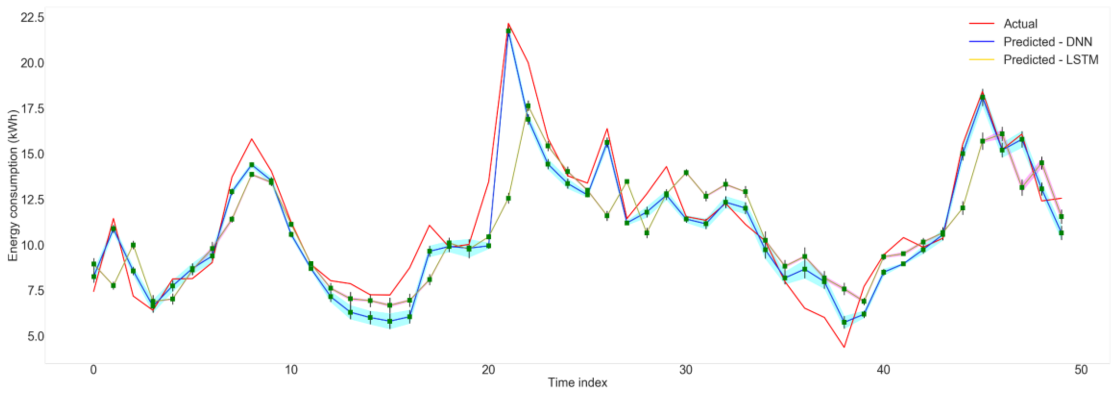


Figure 36 Error bars for forecasts using DNN and LSTM-based clustered STLF models

reasonable number of times i.e., 20 times to obtain the mean prediction and standard deviation of the predictions. LR-based clustering models had zero variance for predictions and hence, are not plotted. LSTM-based clustering models have variance tending to zero and additionally, DNN-based models have very low variance as shown in Figure 36. The sources of randomness are kept at the minimum whilst training the proposed models and the trained models can be saved using deep learning serialization for future testing in industrial applications. The standard deviation of the error metrics for retraining of forecasting models under similar initialization conditions will be negligible.

6.2.2 Transfer Learning

6.2.2.1 Transfer Learning Results

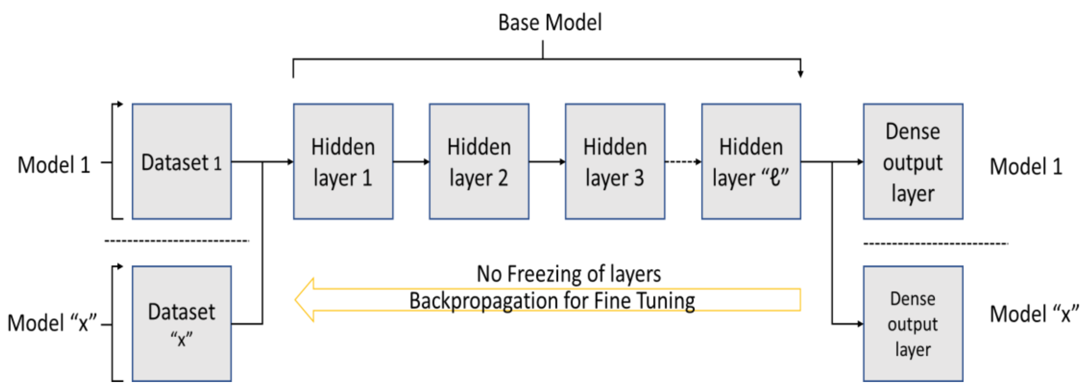


Figure 37 Homogeneous Transfer Learning through Fine Tuning

The proposed solution uses homogeneous inductive transfer learning by fine-tuning through all layers for target tasks. The homogeneous transfer learning is illustrated in Figure 37. As shown in Figure 37, dataset 1 is employed to train model 1 from scratch i.e., the weights of hidden layers in the base model are optimized. During the development of model x, the base layers from model 1 are utilized without freezing and the fine-tuning is performed through all layers.

The integrated methodology of the construction of power forecasting models is depicted in Figure 38. The data of thousand distribution nodes are passed through the clustering stage to form the group of similar distribution nodes into clusters. The optimal number of clusters is determined to be 93 clusters [174]. Similar distribution nodes are formed into clusters.

The hyperparameter k in the k-Medoids algorithm cannot be learned directly and hence, the elbow curve method is employed to select the optimal value of k which yields the least within-cluster error.

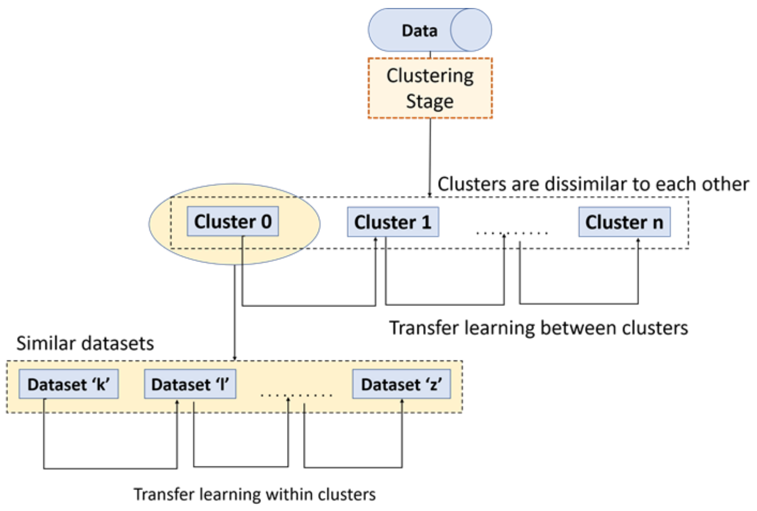

Data
Clustering Stage
Clusters are dissimilar to each other
Cluster 0
Cluster 1
Cluster n
Transfer learning between clusters
Similar datasets
Dataset 'k'
Dataset 'l'
Dataset 'z'
Transfer learning within clusters

Figure 38 Clustering-based methodology with TL

In the next stage of methodology, a forecasting model one each for a cluster is developed using transfer learning. That is, a forecasting model (model 0) is firstly trained from scratch on the source dataset (cluster 0). Secondly, the model is re-trained on target datasets (cluster 1, cluster 2, . . ., cluster n) through fine-tuning all the layers in the neural network. For convenience, the clustered models formed using TL are denoted as Clus-Tr-DNN and the clustered models formed without TL framework are denoted as Clus-DNN where DNN indicates the inherent deep learning neural network model. The accuracies of Clus-Tr-DNN are compared with the accuracies of the Clus-DNN.

The next stage of the proposed methodology involves the creation of models within clusters. These are individual models developed for each dataset. Already, the datasets which are similar in energy consumption patterns have been clustered together in the previous stage. Now, the knowledge transfer is performed only between the distribution datasets within the same clusters to eliminate any negative transfer of knowledge. In the first subset of experiments, TL is used to construct the subsequent models within a cluster using knowledge transfer from the source domain within the same cluster. For convenience, these models are denoted as Ind-Tr-DNN. To develop source domains from cluster 1 onwards, we utilize weight regularization optimizer to transfer knowledge from source domain within cluster 0. The use of weight regularization eliminates negative learning when knowledge transfer occurs between clusters. In another subset of experiments, the individual models are developed without the use of any TL. For convenience, these models are denoted as Ind-DNN.

Extensive experiments were performed to evaluate the performance of transfer learning-based methodology. The utilized datasets are the energy consumption data at ten and thousand distribution nodes in the Spanish electrical network.

In one set of experiments, individual models are developed using the individual datasets and in another set of experiments, the clustering approach is applied to group the similar distribution nodes into clusters based on the similarity metric of daily energy consumption.

The clustering technique employed is the k-Medoid clustering technique to eliminate the sensitivity to outliers in data analytics. According to the within-cluster error elbow curve, the optimal number of clusters is determined as 3 for 10 distribution nodes data and as 93 for 1000 distribution nodes dataset [174].

The initial cluster (cluster 0) is trained using the conventional way without any TL. The other clusters are trained with the help of TL from cluster 0 and the fine-tuning is performed using the corresponding dataset of the cluster. The knowledge from the training of cluster 0 is used for training cluster 1, cluster 2, and so on.

6.2.2.1.1 Results on Dataset 1 – ten distribution nodes dataset

The performance of traditional learning and TL between dissimilar clusters on clustered models for ten distribution nodes dataset is depicted in Table 17.

Table 17 Testing of clustered models on cluster data with and without TL framework applied between clusters.

	RMSE (kWh)		**Improvement (%)**
	Clus-DNN	**Clus-TR-DNN**	
Cluster 0	4.35	-	-
Cluster 1	12.86	9.92	+22.86 %
Cluster 2	17.10	23.68	- 34.47 %

The RMSE of cluster 1 shows significant improvement after the transfer of knowledge. However, the performance of the model for cluster 2 shows a negative transfer of learning indicating that the model converged to a local minimum rather than a global optimization point. The negative learning can be explained because the TL is performed between the dissimilar distribution nodes belonging to different clusters. A few potential solutions that can be considered to avoid convergence to local minima are the following [221], [222]: i) considering cyclic learning rate, ii) using Stochastic Gradient Descent (SGD) with warm restarts, iii) considering high values for learning rate, iv) using meta-heuristic algorithms such as Grey-Wolf Algorithm, Ant Colony Optimization, and Harmony Search, and v) variants of optimizers such as Vanilla Gradient Descent, AdamW, QHAdam, YellowFin, AggMo, QHM, and Demon. The negative TL can be removed when the transfer of knowledge happens between the distribution nodes that are similar. This is observed in subsequent tables. Moreover, the improvement with TL is more pronounced when the data for target tasks are not sufficiently large.

The ten distribution nodes are clustered into 3 clusters. With the k-Medoid clustering algorithm, it was determined that the three clusters of distribution nodes are: {0, 1, 2, 6}, {5} and {3, 4, 7, 8, 9}. One clustered model based on deep neural networks was developed for each cluster. So, the three clustered models have been developed and these have been tested on the individual datasets of the distribution nodes and the results of the performance with and without the use of TL are depicted in Table 18.

Table 18 Testing of clustered models on individual distribution node datasets with and without TL framework applied between clusters.

	RMSE (kWh)		**Improvement (%)**
	Clus-DNN	**Clus-Tr-DNN**	
t/f 0	27.95	-	-
t/f 1	31.39	-	-
t/f 2	29.91	-	-
t/f 6	27.50	-	-
t/f 5	21.15	22.29	5.39
t/f 3	18.49	12.97	29.85
t/f 4	21.75	20.92	3.81
t/f 7	16.52	20.42	-23.62
t/f 8	16.19	19.32	-19.33
t/f 9	17.47	20.52	-17.45

The first column in the table represents the distribution node number or transformer number (t/f). The similar distribution nodes are grouped into the same clusters however, any two clusters are assumed to be dissimilar. With the transfer of knowledge between dissimilar clusters, it is possible that the transfer is either positive or a little on the negative side. However, the gain in the execution or training time is always positive. The gain in time is displayed in Table 19. From Table 19, it is clear that the time to train the models with TL is much less than the time to train the models without TL.

Table 19 Cluster training times after testing of clustered models with TL applied between clusters.

	Training time (s)		**Improvement (%)**
	Clus-DNN	**Clus-Tr-DNN**	
Cluster 0	92	-	-
Cluster 1	49	40	9
Cluster 2	79	50	36.7

The negative transfer of knowledge is inherently eliminated when the TL is employed between similar distribution nodes. The k-medoid clustering algorithm based on similarity metric of energy consumption clustered the 10 distribution nodes into the clusters {0, 1, 2, 6}, {5} and {3, 4, 7, 8, 9}. The negative TL is eliminated when the knowledge from the model of dataset 0 is transferred to develop models on dataset 1, 2, and 6. The knowledge from the model of dataset 3 is transferred to develop models on datasets 4, 7, 8, and 9. The use of the clustering-based methodology eliminated any negative TL within a cluster and the results are described in Table 20. The negative transfer learning between dissimilar clusters is eliminated by the weight regularization technique proposed in Section 6.2.2.3.

Table 20 Testing of individual models on individual distribution node datasets with TL applied within clusters.

	RMSE of individual models without TL (kWh)	**RMSE of individual models after transfer of knowledge (kWh)**	**improvement (%)**
t/f 0	13.60	-	-
t/f 1	10.32	7.90	23.44
t/f 2	7.35	5.03	31.56
t/f 6	6.42	1.09	83.02
t/f 5	18.68	-	-
t/f 3	2.18	-	-
t/f 4	2.30	1.26	45.21
t/f 7	14.91	10.34	30.65
t/f 8	3.81	2.22	41.73
t/f 9	3.70	2.52	31.89

6.2.2.1.2 Results on Dataset 2 – thousand distribution nodes dataset

The performance of TL with respect to training time has also been verified with a second case study on 1000 distribution nodes which according to elbow curve and k-Medoid clustering were grouped into 93 clusters and the models were developed using deep neural networks. As shown in Figure 39, the time to train the clustered models using

TL is always less when compared to the time taken to train the clustered models without TL. This confirms that the TL allows for faster convergence of models. The performance of TL in coalition with the clustering layer on the thousand distribution nodes dataset is depicted in Table 21. It takes 3.23 mins to develop 93 clustered models using TL when compared to 2.20 hours of training time without TL. However, the MAPE varies from 7.22% to 14.37% when TL is employed between dissimilar clusters.

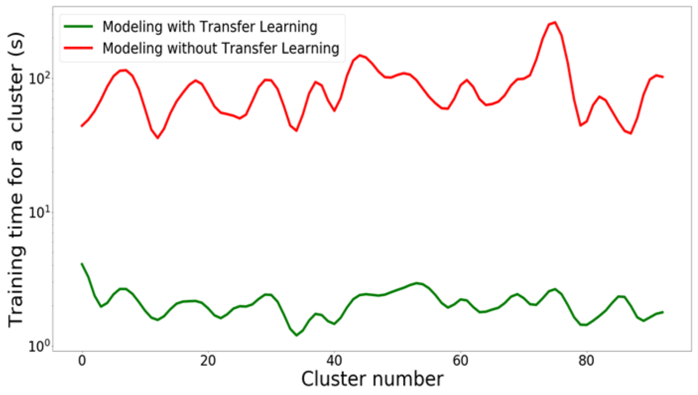


Figure 39 Cluster training times for 1000 distribution nodes with TL applied between clusters

6.2.2.1.3 Weight Regularization to eliminate negative learning between dissimilar datasets

For transfer learning between dissimilar clusters, an improved Adam optimizer was proposed to eliminate any negative learning and to breakout from local convergence. The first optimization step involves the use of cyclical learning rate in which learning rate is initialized to a larger value and is scheduled to decrease subsequently to prevent the avoidance of global minima. The proposed optimizer invariant is utilized with decoupled weight regularization and cyclical learning rate (Adamw) to eliminate negative learning. The weight update rule in the general Adam optimizer is given by the following:

$$w(t) = w(t-1) - \alpha \nabla f, \tag{49}$$

Here ∇f is the gradient, and α is the learning rate.

The general Adam optimizer is characterized by large step size when gradient change is less, smaller step size when gradient change is rapid and the adaptability in step size is performed by maintaining moving averages (called moments) of gradient over the steps.

The implemented optimizer invariant employs decoupled weight regularization. This allows for weight regularization without the coupling of hyperparameters such as learning rate and weight decay factor.

The weight update rule in the proposed optimizer invariant is given by the following:

$$w(t) = (weightdecayfactor)w(t-1) - \alpha \nabla f, \tag{50}$$

Here ∇f is the gradient, and α is the learning rate.

The weight decay factor is introduced as a coefficient to the weight at past instant and lies between 0 and 1. This forces the weights learnt to be small and so, the model generalizes better. For convenience, the models using weight regularization are denoted by Clus-Tr-WR-DNN.

6.2.2.2 Results of TL after weight regularization

6.2.2.2.1 Results on Dataset 1 – ten distribution nodes dataset

Figure 40 and Figure 41 indicate the performance of TL after weight regularization on 10 distribution nodes dataset. The results, obtained after the testing of clustered models

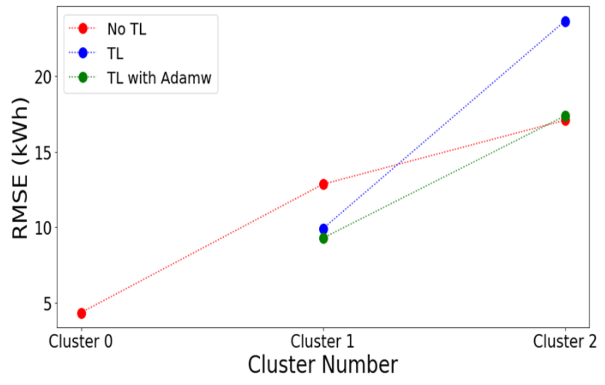


Figure 40 TL between clusters – testing on cluster data

is performed on cluster data, are illustrated in Figure 40. The graph of TL with weight decay regularization is at the lower bound of error when compared to the model development without TL for both cluster 1 and cluster 2. At no point, the error is high in case of model development after TL. This indicates that the negative learning has been eliminated by the use of weight regularization in the optimizer.

The results, obtained after the testing of clustered models on individual transformers' data, are illustrated in Figure 41. The graph of TL with weight decay regularization is at the lower bound of error when compared to the model development without TL for all the transformers including t/f 1, t/f 8, t/f 3, t/f 6, t/f 7, t/f 9. At no point,

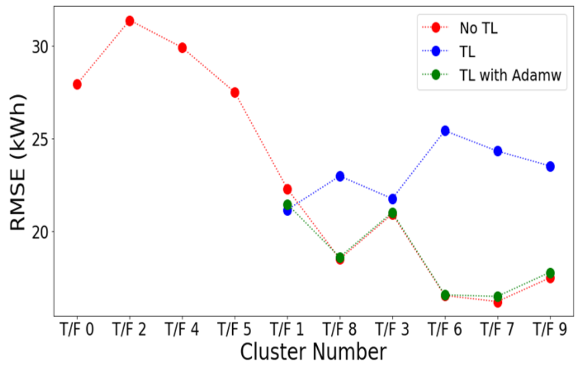


Figure 41 TL between clusters – testing on individual transformer data

the error is high in case of model development after TL. This corroborates that the negative learning has been eliminated by the use of weight regularization in the optimizer.

6.2.2.2.2 Results on Dataset 2 – thousand distribution nodes dataset

The performance of TL after weight regularization on 1000 distribution nodes dataset is presented in Table 21. To analyze the performance of the proposed weight regularization TL modeling (Clus-Tr-WR-DNN), several state-of-the-art benchmark models, including Linear Regression (LR), Autoregressive Integrated Moving Averages (ARIMA), and deep long-short term memory (LSTM) are selected as comparative methods as shown in Table 21. Weight regularization utilized during objective function optimization in the proposed model eliminates negative knowledge transfer. The proposed Clus-Tr-WR-DNN has a higher overall development time of 20.17 mins whilst maintaining average MAPE error to a minimum of 7.20% when compared to Clustering-based TL modeling that has 3.23 mins as development time and average MAPE of 31.96%.

Table 21 Performance of TL on thousand distribution nodes dataset (Dataset 2).

Model	**Train time (min)**	**Average MAPE (%)**	**Average RMSE (kWh)**
Ind-LR [174]	0.23	20.32	54.04
Clus-LR [174]	0.28	20.84	62.35
Ind-ARIMA [174]	8.55	37.78	59.41
Clus-ARIMA [174]	5.28	39.45	67.87

Ind-LSTM [174]	1890	7.27	22.52
Clus-LSTM [174]	485	11.11	37.06
Ind-DNN	140.15	7.18	19.82
Clus-DNN	77	7.22	21.25
Clus-Tr-DNN	**3.23**	14.37	31.96
Clus-Tr-WR-DNN (Proposed)	20.17	**7.20**	**22.10**

6.2.2.3 Results of TL on targets with smaller datasets

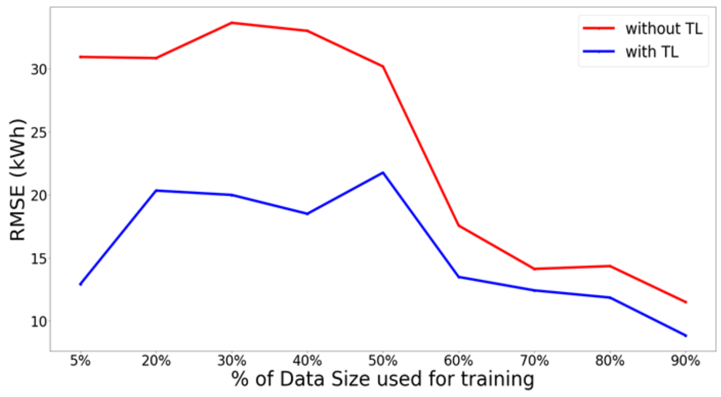


Figure 42 TL results when the data availability is low

Besides, the effect of TL has been analyzed with smaller datasets. As observed in Figure 42, for smaller datasets, the model developed from scratch has low accuracy when compared to the model with knowledge transferred from a similar distribution point. As the size of the dataset increases, the accuracy of both the models, with and without TL, increases, and when a threshold size is reached, these models will have very close accuracy values. The results of the performance of TL, when the data availability is low, is verified on the available dataset (Table 22). It has been found that the model with TL performs 58% better than the model without TL when the data size for the second model is 5% of the original dataset. In all the cases of data availability, the TL model outperforms the conventional model by 13-43%.

Table 22 Performance of TL when the data availability is low.

Data size	**RMSE without TL**	**RMSE with TL**	**Improvement (%)**
5%	33.9412	20.9255	38.34
20%	33.8498	20.3398	39.91
30%	33.6460	19.9956	40.57
40%	33.0107	18.5012	43.95
50%	30.1865	17.7643	41.15
60%	17.5656	13.4901	23.20
70%	14.3533	12.4246	13.43
80%	14.0287	11.8557	15.48
95%	11.5013	8.8494	23.05

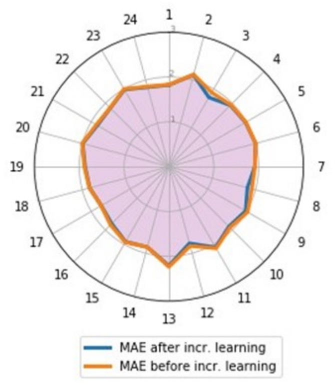


Figure 43 Radar plot for MAE for different horizons

6.2.3 Incremental Learning

With the incremental learning layer, the load forecasting can be performed in real-time with streams of data collected at an interval of 1 hour. From Figure 43 and Figure 44, it is evident that the mean absolute error of the predictions using incremental machine learning is at the minimum for the horizon value of 24 hours.

Figure 45 and Figure 46 depict the RMSE illustration for different horizons of short-term load forecasting on the dataset of 1000 transformers. Table 23 depicts the results of online machine learning for energy forecasting. It depicts the enhancement in the accuracy of the

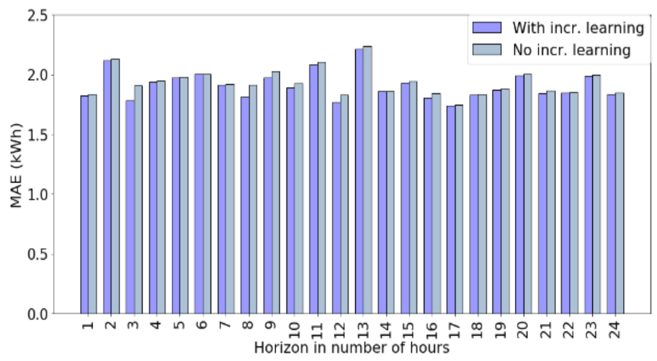


Figure 44 MAE for different horizons Incremental learning based STLF model

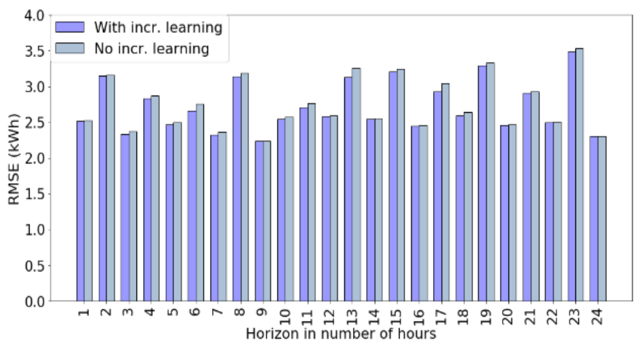


Figure 45 RMSE for different horizons Incremental learning based STLF model

incrementally trained models over the clustered models and also indicates that 7 minutes is required for every 6 hours to generate the incrementally trained models for 1000 distribution transformers. To capture the daily trends of energy consumption, the incremental algorithm has been invoked every 24 hours. The results of incremental learning with a 24-hour horizon indicate improvement in accuracy while also maintaining very low execution time for incremental stage.

The experiment is performed to illustrate the performance of incremental learning. There are three models developed here for every 24 hours. At time t=0, there is only one model developed which is from scratch using the historical data which is available. Now, this model is kept constant to predict the energy consumption values of all future days in

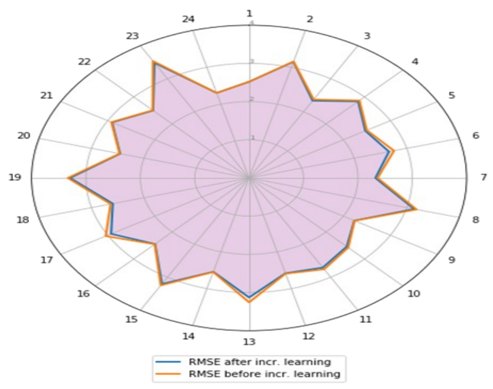


Figure 46 Radar plot for RMSE for different horizons

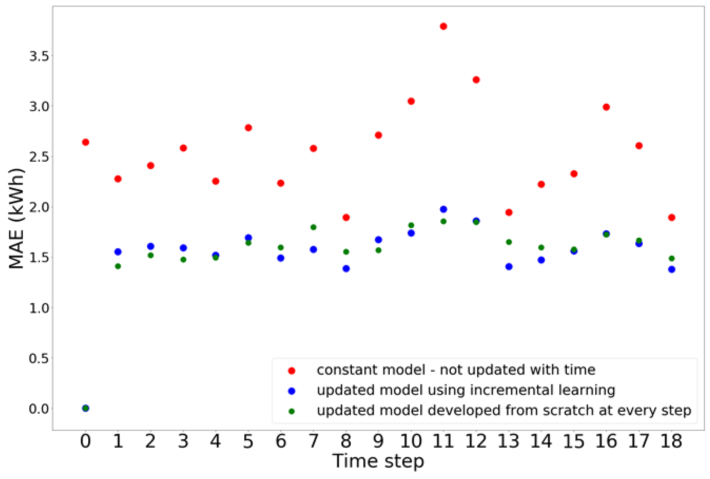


Figure 47 Improvement using incremental learning

case I which is represented by red dots in Figure 47. The second model is updated every 24 hours with the help of incremental learning over the model at the last time point. It is represented by blue dots in the figure. The third model is an updated model till the current time point; however, it is developed from scratch at every time point. From the graph, it is evident that incremental learning provides efficient performance in terms of accuracy and it is developed in a fraction of the time it takes for model development from scratch.

Table 23 Online machine learning results.

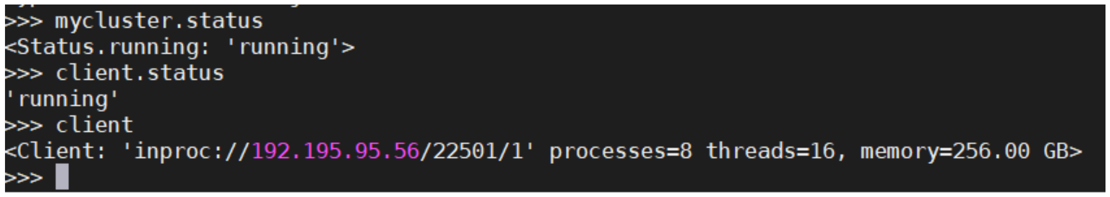
	DNN (train stage)	**DNN (incremental stage 6h horizon)**	**DNN (incremental stage 24h horizon)**
Avg. RMSE (kWh)	21.2596	20.1149	19.43
Train time (s)	1210.56 sec for training stage	Extra 400 secs every 6 hours.	Extra 403 secs every 24 hrs.
Avg. MAPE (%)	7.20	6.92	6.37

6.2.4 Dask Parallel Computation

Dask framework is a flexible parallel computation library for data analysis. It works on datasets in an out-of-memory fashion, uses multiple cores inherently unlike python. It is efficient in high-performance computations with high flexibility, scalability, high throughput features, maximizes the utilization of cores, and memory. On a single node, Dask performs automatic scaling to a cluster of cores and utilizes its ability to scale over a cluster of nodes, when available. The characteristic features favoring dask in multiprocessing are 1) possibility of data sharing between workers, 2) low latency performance, 3) support for complex scheduling, and 4) easy to setup.

In complex use cases where big data platforms such as spark do not provide a solution, for instance in transfer learning and incremental learning, dask is of much relevance. The big data processing engines are written in Scala, Java, and Python. However, dask is written purely in python. It can additionally interoperate with other python libraries such as scikit-learn, NumPy, pandas, and Keras.

Dask provides flexibility to choose threads or processes. Initially, a dask setup was used for multiprocessing and a cluster was set up as illustrated in Figure 48 and Table 24.



```
>>> mycluster.status
<Status.running: 'running'>
>>> client.status
'running'
>>> client
<Client: 'inproc://192.195.95.56/22501/1' processes=8 threads=16, memory=256.00 GB>
>>> 
```

Figure 48 Dask cluster setup

Table 24 Configuration of dask cluster.

Parameter	**Value**
Workers	8

Threads	2 per worker
Memory limit	32 GB per worker

Dask provides the following user interfaces:

- High-level: Arrays, Bags, Dataframes.
- Low-level: Task schedulers for computation graphs. (Directed Acyclic Graphs).

6.2.5 Multi-core processing in Python

Fundamentally, Python has a bottleneck in the form of Global Interpreter Locker (GIL) that limits the capacity of multiple threads or programming processes. GIL exists on the compiler of python denoted by CPython and it causes a considerable penalty to the speed of multi-threaded python programs. Generally, multi-threaded python programs are 50% slower than single-threaded operations owing to the significant CPU wait time. The problem of GIL in python can be overcome by the use of multiple processes instead of using multiple threads. The multiprocessing python library can be employed to imitate the multiple threads library interface and to solve the GIL problem.

In this research, the method of pool mapping is employed for multicore processing. To map to the execution pool, a user-defined function is created which performs the analysis of the data. The analyses include clustering into groups, performing transfer learning, and incremental learning. So, the program is executed in two steps:

Firstly, there is the main program that reads the data and stores the data in memory. There is a user-defined function that has the definition to predict and generate RMSE. Secondly, the main program calls this user-defined pool function multiple times, and each of these calls is handled by a separate core. Hence, the multiple cores handle jobs simultaneously. Each core is mapped to the same subprogram but with a different cluster number. The application is built to support the functionality of multicore processing using the pool mapping method.

The results of multi-core processing in the 1000 transformers dataset are depicted in Table 25. The scalability of the results is verified up to the size of 10,000 transformers.

Table 25 Results of multi-core processing.

Metric	**Clustering + Training**		
	1 core	8 cores	
	Single-core processing	map method	map async
Avg RMSE (KWh)	21.2596	21.2596	21.2596
MAPE	7.57	7.57	7.57
Train Time	4688.82	1054.188	903.589
Testing Time	4.57	2.690	2.286

6.2.5.1 Metrics of multi-core processing

The performance of multi-core processing is evaluated in the following measures:

- *CPU Efficiency:* CPU Efficiency is calculated as the ratio of the actual core time from all cores divided by the number of cores requested divided by the run time.
- *Memory Efficiency:* Memory Efficiency is calculated as the ratio of the high-water mark of memory used by all tasks divided by the memory requested for the job.
- *Parallel Efficiency:* Parallel efficiency, which compares the performance of the full system or a specific subset of the processors to the performance of one processor

Two of the reasons for low CPU Efficiency are I/O bottleneck and CPU-bound bottlenecks. I/O bottlenecks are bottlenecks where a computer processor spends more time waiting on various inputs and outputs than it does on processing the information. CPU-bound bottlenecks exist in applications with a large amount of data to process. The CPU-bound bottlenecks are limited by the computational speed of CPUs. If the CPUs are of high computational power, then the speed of processing is fast.

6.2.5.2 Ways to improve CPU Efficiency

The different ways to improve CPU efficiency while performing multi-core processing for load forecasting include the following:

- Efficient Inter-process communication
- Different methods involving the utilization of following parallel computing procedures:
    1) Pipes (package: os)
    2) Files (package:py-filelock; filelocks)
    3) Message Queue (package:activeMQ, redis.)
    4) Shared Memory (package:mmap; shared memory registers)
    5) Sockets
    6) Signals
    7) Remote procedure call (RPC)

It is important to analyze how multi-cores perform actions. A core is an individual CPU unit that has all independent components and architecture to execute information. The following are the steps in a cycle through which each core goes through:

- Fetch: It involves fetching instructions from the program memory. It is dictated by a program counter (PC), that identifies the location of the next step to the process.
- Decode: The core converts the fetched instruction into a series of signals that will trigger other components of the CPU.
- Execute: Finally, the execute step is performed. This is where the fetched and decoded instruction, is executed and the results are stored in a CPU register.

The results of eight-core processing depict that the parallel load forecasting can be performed using multiple cores with a parallel efficiency of 4 to 5. That is, when the number of cores used to perform parallel load forecasting is 8, the training time is reduced to $\frac{1}{4}$ th to $\frac{1}{5}$ th of the training time it takes when one core is utilized for processing.

6.2.6 Results of proposed parallel computing methodology on big data

To prove the scalability of the approach of parallel computing and multiprocessing using python framework, the modeling has been performed on datasets with increasing size in stages. The results of the proposed methodology on datasets collected from 10 diverse sources and 1000 diverse sources have been illustrated in the previous subsections. In this subsection, the performance of proposed methodology on datasets that can be considered big data is illustrated.

The optimization of number of clusters developed on dataset of 10,000 diverse sources is shown in Figure 49. The figure represents the elbow curve that depicts the within cluster sum of square errors on the dependent axis and the number of clusters is shown on the independent axis. The dip in the error is observed at x=890 indicating that 890 clusters would be optimal to develop non-overlapping clusters on the dataset. This is in resonance to the results of elbow curve on 10 and 1000 transformers dataset that imply

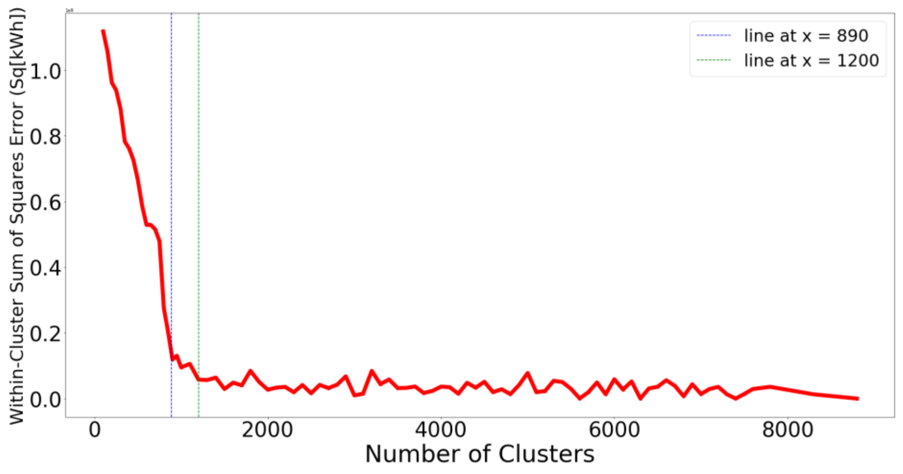


Figure 49 Elbow curve on 10,000 transformers dataset

that the factor of decrease in the number of models is 10 when applying clustering to the distribution transformers dataset.

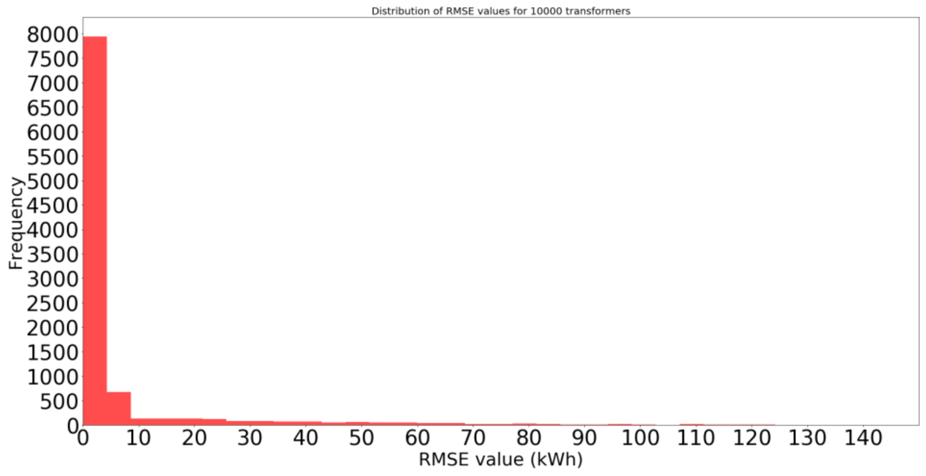


Figure 50 Distribution of RMSE values for 10,000 transformer models

Hence, the forecasting models were developed for 10,000 distribution transformers and the statistical distribution of RMSE of the models is depicted in Figure 50. As per Figure 50, nearly 8000 transformer forecasting models have RMSE value less than 5 kWh indicating high success of the forecasting models. The statistical distribution of RMSE against MAPE is represented in Figure 51.

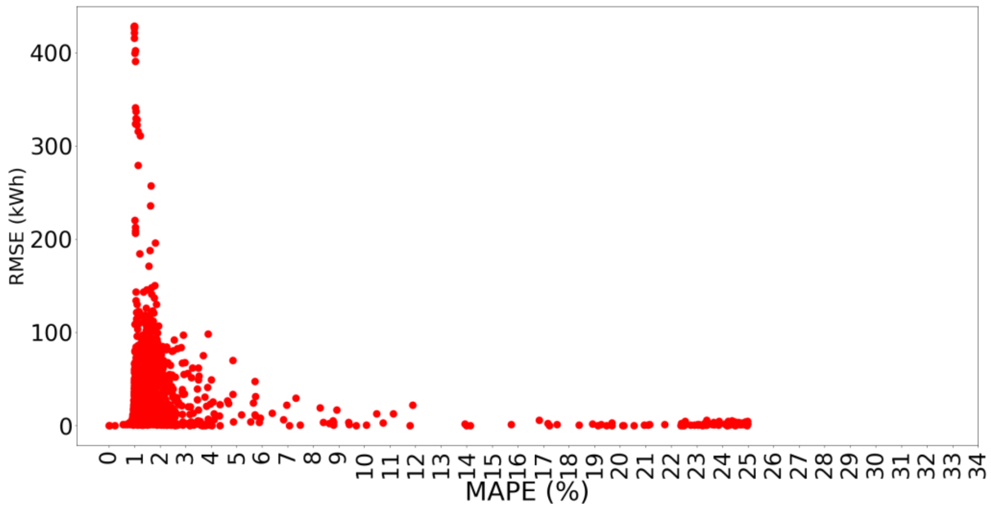


Figure 51 Distribution of RMSE against MAPE for 10,000 transformer

The results of development of models for 10,000 transformers is depicted in Table 26. Additionally, it takes 48 minutes to cluster the 10,000 transformers into similar clusters and then takes around 3.09 hours to develop the forecasting models for the clusters. With the usage of dask dataframe to read or filter and pandas dataframe to process or sort data, the peak memory usage was reduced from 97.05 GB to 42.75 GB during the development of individual models on the dataset of size 64 GB.

Dataset 3 described in Table 27 is an example of big data as it contains around 2.2 billion records, and the processing of the dataset requires out of memory and parallel computing. The results of parallel multi-core processing on a non-Spark platform is mentioned in Table 27.

Table 26 Results on 10,000 transformers.

Data	**Field**	**Cores**	**Reading**	**k-optimization**	**Clustered models**	**Individual models**
10,000 transformers	Avg. RMSE (kWh)	8	-	-	8.94	7.65
	Training time (s)	8	5.88 (dask)	6480 = 1.8 hr	3.09 hr	50664.5 = 14 h
	Test time (s)	8	-	-	45.68	2802.5 = 46.7 min

Table 27 Results of parallel computing on big data (dataset 3).

Parameter	**Parameter value**
No. of transformer models trained	105,148
Resources	8 nodes. 48 cores each node.
Execution time	144 hours (5.9 days)
Mean RMSE	5.13 kWh
Mean nRMSE	0.0619
Mean MAPE	11.634

Mean MAE	3.282

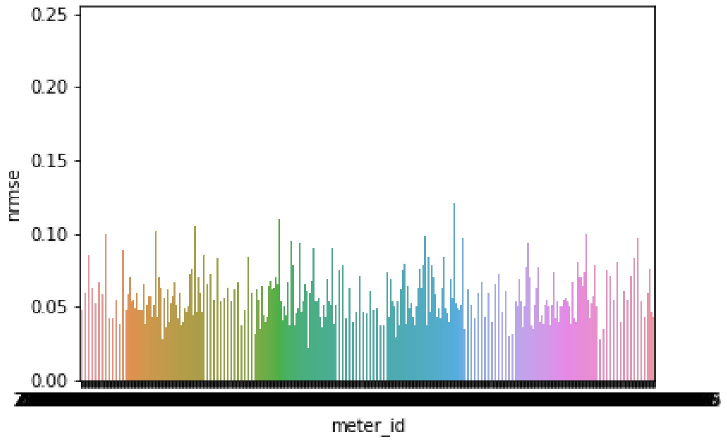


Figure 52 nRMSE results of the developed models for big data

The accuracy performance can be observed in Figure 52. The independent axis denotes the transformer number/model number, and the dependent axis represents the nRMSE value for a particular transformer. As shown in Figure 52, the nRMSE value for

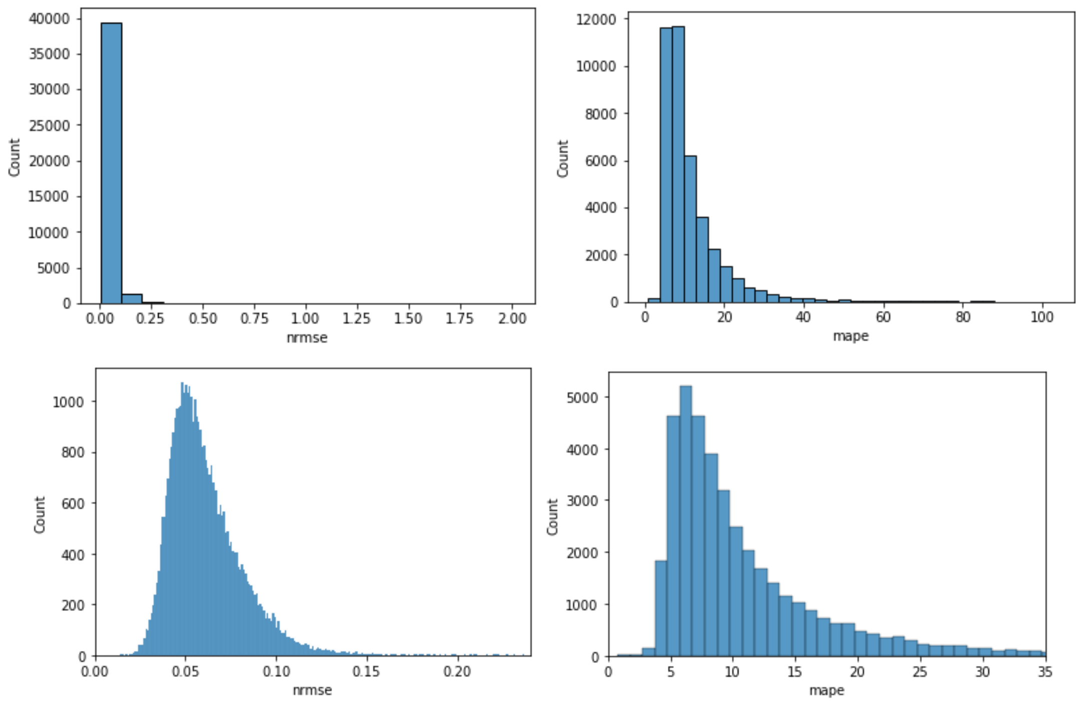


Figure 53 Frequency plots of nRMSE and MAPE for the developed models

the transformers do not exceed the value of 0.12 and this low value indicates the highly accurate prediction performance of the developed models.

The frequency plots of the performance of the developed and trained models are illustrated in Figure 53.

The performance of parallel computing in terms of execution time is tabulated in Table 28.

Table 28 Time performance of parallel computing on multiple datasets.

No. of transformers	**Model training time (hr)**	**Datasize (GB)**	**No. of nodes used**	**Cores used**	**RAM**
1,000	2.33	2.85	1	1	8 GB
10,000	14	23.78	1	8	64 GB
105,148	144	242.78	8	48	128 GB/node

As observed in Table 28, the dataset size is increased linearly by 10 and 100 times. And, with the help of parallel computing and multicore processing, the development of models occurs under similar scaled time.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

The use of the proposed hybrid framework with deep learning can be utilized to develop parallel and real-time forecasting system for the smart grids and to change positively the way the electrical grids save energy. In this work, various feature selection methods, clustering techniques, transfer learning, incremental learning, and multi-core processing have been employed and the integration of diverse data from multiple sources has been performed. The outstanding results of the proposed methodology demonstrate the reduction in the number of trained models by a factor of 10, the reduction in training time by a factor of 2, and the improvement in accuracy owing to real-time analytics approach and incremental learning of trained networks. Additionally, when the multi-core analysis is performed, the execution time is reduced by a factor of at least $\frac{k}{2}$, where k is the number of cores employed. According to the results, it is evident that the parallel computing of load forecasting provides satisfactory performance in terms of accuracy and computation times. The proposed PULSE cost function-based DL models eliminate the tendency to underestimate. The performance of the proposed multi-stage hybrid framework is evaluated on data collected from 100,000 diverse data sources indicating that the framework is highly scalable.

7.2 Future Work

In the future, the work can be further extended towards the optimization of the number of nodes, number of cores, and provided memory. This optimization will help in the computation resource management for parallel computing. Current steps in progress are the partitioning of the recurrent network connections in neural networks to reduce the number of trainable parameters and to reduce the computational time for training and building networks. Additionally, the impact of forecasting horizon on the proposed PULSE function will be studied.

REFERENCES

[1] Y. Zhang, R. Yu, M. Nekovee, Y. Liu, S. Xie, and S. Gjessing, "Cognitive machine-to-machine communications: Visions and potentials for the smart grid," *IEEE Netw.*, vol. 26, no. 3, pp. 6–13, May 2012, doi: 10.1109/MNET.2012.6201210.

[2] T. H. Dang-Ha, R. Olsson, and H. Wang, "The role of big data on smart grid transition," in *Proceedings - 2015 IEEE International Conference on Smart City, SmartCity 2015, Held Jointly with 8th IEEE International Conference on Social Computing and Networking, SocialCom 2015, 5th IEEE International Conference on Sustainable Computing and Communic*, Dec. 2015, pp. 33–39, doi: 10.1109/SmartCity.2015.43.

[3] M. K. Saggi and S. Jain, "A survey towards an integration of big data analytics to big insights for value-creation," *Inf. Process. Manag.*, vol. 54, no. 5, pp. 758–790, Sep. 2018, doi: 10.1016/j.ipm.2018.01.010.

[4] D. B. Rawat and C. Bajracharya, "Cyber security for smart grid systems: Status, challenges and perspectives," in *Conference Proceedings - IEEE SOUTHEASTCON*, Apr. 2015, vol. 2015-June, no. June, pp. 1–6, doi: 10.1109/SECON.2015.7132891.

[5] J. N. Bharothu, M. Sridhar, and R. S. Rao, "A literature survey report on Smart Grid technologies," in *2014 International Conference on Smart Electric Grid, ISEG 2014*, Sep. 2015, pp. 1–8, doi: 10.1109/ISEG.2014.7005601.

[6] J. Joy, E. A. Jasmin, and V. John, "Challenges of Smart Grid," *ISSN Int. J. Adv. Res. Electr. Electron. Instrum. Eng.*, vol. 2, no. 3, pp. 2320–3765, 2013, [Online]. Available: www.ijareeie.com.

[7] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on smart grid communication infrastructures: Motivations, requirements and challenges," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1. pp. 5–20, 2013, doi: 10.1109/SURV.2012.021312.00034.

[8] Expansión, "Iberdrola bets on 'Big Data' to manage its electricity networks." Accessed on: Oct. 2, 2021., [Online]. Available: http://www.expansion.com/empresas/energia/2015/04/23/.

[9] P. Siano, "Demand response and smart grids - A survey," *Renew. Sustain. Energy Rev.*, vol. 30, pp. 461–478, Feb. 2014, doi: 10.1016/j.rser.2013.10.022.

[10] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*, vol. 7. McGraw-hill New York, 1994.

[11] M. L. Tuballa and M. L. Abundo, "A review of the development of Smart Grid technologies," *Renewable and Sustainable Energy Reviews*, vol. 59. Elsevier Ltd,

pp. 710–725, Jun. 2016, doi: 10.1016/j.rser.2016.01.011.

[12] S. Li, L. Goel, and P. Wang, "An ensemble approach for short-term load forecasting by extreme learning machine," *Appl. Energy*, vol. 170, pp. 22–29, May 2016, doi: 10.1016/j.apenergy.2016.02.114.

[13] M. N. Q. Macedo, J. J. M. Galo, L. A. L. De Almeida, and A. C. De, "Demand side management using artificial neural networks in a smart grid environment," *Renewable and Sustainable Energy Reviews*, vol. 41. Elsevier Ltd, pp. 128–133, Jan. 2015, doi: 10.1016/j.rser.2014.08.035.

[14] G. M. U. Din and A. K. Marnerides, "Short term power load forecasting using deep neural networks," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 594–598.

[15] K. G. Boroojeni, M. H. Amini, S. Bahrami, S. S. Iyengar, A. I. Sarwat, and O. Karabasoglu, "A novel multi-time-scale modeling for electric power demand forecasting: From short-term to medium-term horizon," *Electr. Power Syst. Res.*, vol. 142, pp. 58–73, Jan. 2017, doi: 10.1016/j.epsr.2016.08.031.

[16] S. R. Khuntia, J. L. Rueda, and M. A. M. M. van der Meijden, "Forecasting the load of electrical power systems in mid- and long-term horizons: A review," *IET Gener. Transm. Distrib.*, vol. 10, no. 16, pp. 3971–3977, Dec. 2016, doi: 10.1049/iet-gtd.2016.0340.

[17] C. Xia, J. Wang, and K. McMenemy, "Short, medium and long term load forecasting model and virtual load forecaster based on radial basis function neural networks," *Int. J. Electr. Power Energy Syst.*, vol. 32, no. 7, pp. 743–750, Sep. 2010, doi: 10.1016/j.ijepes.2010.01.009.

[18] L. Hernández *et al.*, "Artificial Neural Network for Short-Term Load Forecasting in Distribution Systems," *Energies*, vol. 7, no. 3, pp. 1576–1598, Mar. 2014, doi: 10.3390/en7031576.

[19] F. Mwasilu, J. J. Justo, E. K. Kim, T. D. Do, and J. W. Jung, "Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration," *Renew. Sustain. Energy Rev.*, vol. 34, pp. 501–516, 2014, doi: 10.1016/j.rser.2014.03.031.

[20] Accenture, "Unlocking the Value of Analytics." Accessed on: Jul. 20, 2020., pp. 1–20, 2014, [Online]. Available: https://www.accenture.com/.

[21] D. Syed, A. Zainab, S. S. Refaat, H. Abu-Rub, and O. Bouhali, "Smart Grid Big Data Analytics: Survey of Technologies, Techniques, and Applications," *IEEE Access*, vol. 9, pp. 59564--59585, Nov. 2020, doi: 10.1109/access.2020.3041178.

[22] Y. Simmhan *et al.*, "Cloud-based software platform for big data analytics in smart grids," *Comput. Sci. Eng.*, vol. 15, no. 4, pp. 38–47, 2013, doi:

10.1109/MCSE.2013.39.

[23] R. Mahmud, R. Vallakati, A. Mukherjee, P. Ranganathan, and A. Nejadpak, “A survey on smart grid metering infrastructures: Threats and solutions,” in *IEEE International Conference on Electro Information Technology*, May 2015, vol. 2015-June, pp. 386–391, doi: 10.1109/EIT.2015.7293374.

[24] Y. J. Kim, M. Thottan, V. Kolesnikov, and W. Lee, “A secure decentralized data-centric information infrastructure for smart grid,” *IEEE Commun. Mag.*, vol. 48, no. 11, pp. 58–65, 2010, doi: 10.1109/MCOM.2010.5621968.

[25] Ausgrid, “Ausgrid Average Electricity Use.” Accessed: Sep. 26, 2020., [Online]. Available: https://data.gov.au/data/organization/ausgrid.

[26] Irish Social Science Data Archive, “Electricity Smart Meter Data.” Accessed: Sep. 26, 2020, [Online]. Available: http://www.ucd.ie/issda.

[27] Cornell University, “Smart meter data.” Accessed: Jul. 26, 2019, [Online]. Available: http://buildingdashboard.net/cornell/#/cornell.

[28] École Polytechnique Fédérale de Lausanne (Switzerland), “Smart Grid Data.” Accessed: Sep. 26, 2020, [Online]. Available: http://nanotera-stg2.epfl.ch/data/.

[29] Electric Reliability Council of Texas, “Grid Information Load Data.” Accessed: Sep. 26, 2020, [Online]. Available: http://www.ercot.com/%0Agridinfo/load.

[30] North American SynchroPhasor Initiative, “PMU Data.” Accessed: Mar. 1, 2020, [Online]. Available: https://www.naspi.net/PmuRegistry/#.

[31] Pecan Street Inc., “Pecan Street Dataport.” Accessed: Mar. 1, 2020, [Online]. Available: https://dataport.pecanstreet.org/.

[32] Pennsylvania-New Jersey-Maryland Interconnection, “PJM data.” Accessed: Sep. 26, 2020, [Online]. Available: http://www.pjm.com/.

[33] IEEE, “Data Sets IEEE Intelligent Data Mining and Analysis (IDMA).” Accessed: Jul. 26, 2019, [Online]. Available: https://site.ieee.org/psaceidma/%0Adata-sets/.

[34] UC Berkeley Campus, “Smart grid and building consumption data.” Accessed: Mar. 1, 2020, [Online]. Available: https://us.pulseenergy.com/UniCalBerkeley/.

[35] I. Lee, “Big data: Dimensions, evolution, impacts, and challenges,” *Bus. Horiz.*, vol. 60, no. 3, pp. 293–303, May 2017, doi: 10.1016/j.bushor.2017.01.004.

[36] X. W. Chen and X. Lin, “Big data deep learning: Challenges and perspectives,” *IEEE Access*, vol. 2, pp. 514–525, 2014, doi: 10.1109/ACCESS.2014.2325029.

[37] A. Mohamed, S. S. Refaat, and H. Abu-Rub, “A Review on Big Data Management and Decision-Making in Smart Grid,” *Power Electron. Drives*, vol. 4, no. 1, pp. 1–

13, 2019, doi: 10.2478/pead-2019-0011.

[38] Y. Wang, Q. Chen, T. Hong, and C. Kang, “Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges,” *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3125–3148, May 2019, doi: 10.1109/TSG.2018.2818167.

[39] V. Rajaraman, “Big data analytics,” *Resonance*, vol. 21, no. 8, pp. 695–716, Aug. 2016, doi: 10.1007/s12045-016-0376-7.

[40] Y. Zhang, T. Huang, and E. F. Bompard, “Big data analytics in smart grids: a review,” *Energy Informatics*, vol. 1, no. 1, p. 8, Aug. 2018, doi: 10.1186/s42162-018-0007-5.

[41] J. Sessa and D. Syed, “Techniques to deal with missing data,” in *International Conference on Electronic Devices, Systems, and Applications, Sarawak, Malaysia*, 2017, pp. 1–4, doi: 10.1109/ICEDSA.2016.7818486.

[42] I. Nusrat and S. B. Jang, “A comparison of regularization techniques in deep neural networks,” *Symmetry (Basel).*, vol. 10, no. 11, p. 648, Nov. 2018, doi: 10.3390/sym10110648.

[43] K. Wadhwa *et al.*, “BYTE - Big data roadmap and cross disciplinarY community for addressing socieTal Externalities,” in *11th European Semantic Web Conference - EU projects networking session*, 2014, pp. 6–7, [Online]. Available: http://2014.eswc-conferences.org/sites/default/files/eswc2014euprojects_submission_18.pdf.

[44] N. Tatbul, “Streaming data integration: Challenges and opportunities,” in *Proceedings - International Conference on Data Engineering*, 2010, pp. 155–158, doi: 10.1109/ICDEW.2010.5452751.

[45] R. M. A. Velásquez and J. V. M. Lara, “Principal Components Analysis and Adaptive Decision System Based on Fuzzy Logic for Power Transformer,” *Fuzzy Inf. Eng.*, vol. 9, no. 4, pp. 493–514, 2017, doi: 10.1016/j.fiae.2017.12.005.

[46] J. C. Palomares-Salas, J. J. G. De La Rosa, A. Agüera-Pérez, and J. M. Sierra-Fernandez, “Smart grids power quality analysis based in classification techniques and higher-order statistics: Proposal for photovoltaic systems,” *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2015-June, no. June, pp. 2955–2959, 2015, doi: 10.1109/ICIT.2015.7125534.

[47] E. De Santis, L. Livi, A. Sadeghian, and A. Rizzi, “Modeling and recognition of smart grid faults by a combined approach of dissimilarity learning and one-class classification,” *Neurocomputing*, vol. 170, pp. 368–383, Dec. 2015, doi: 10.1016/j.neucom.2015.05.112.

[48] A. E. Lazzaretti, D. M. J. Tax, H. Vieira Neto, and V. H. Ferreira, “Novelty detection and multi-class classification in power distribution voltage waveforms,”

Expert Syst. Appl., vol. 45, pp. 322–330, Mar. 2016, doi: 10.1016/j.eswa.2015.09.048.

[49] A. R. Khan, A. Mahmood, A. Safdar, Z. A. Khan, and N. A. Khan, “Load forecasting, dynamic pricing and DSM in smart grid: A review,” *Renew. Sustain. Energy Rev.*, vol. 54, pp. 1311–1322, 2016, doi: 10.1016/j.rser.2015.10.117.

[50] Y. Weng, R. Negi, C. Faloutsos, and M. D. Ilić, “Robust data-driven state estimation for smart grid,” *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1956–1967, Jul. 2016, doi: 10.1109/tsg.2015.2512925.

[51] Y. Cai and M. Y. Chow, “Exploratory analysis of massive data for distribution fault diagnosis in smart grids,” 2009. doi: 10.1109/PES.2009.5275689.

[52] 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. Informatics*, vol. 14, no. 4, pp. 1606–1615, 2018, doi: 10.1109/TII.2017.2785963.

[53] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, “Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network,” *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019, doi: 10.1109/TSG.2017.2753802.

[54] M. Q. Raza and A. Khosravi, “A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings,” *Renew. Sustain. Energy Rev.*, vol. 50, pp. 1352–1372, Oct. 2015, doi: 10.1016/j.rser.2015.04.065.

[55] K. Chahine *et al.*, “Electric load disaggregation in smart metering using a novel feature extraction method and supervised classification,” *Energy Procedia*, vol. 6, pp. 627–632, 2011, doi: 10.1016/j.egypro.2011.05.072.

[56] A. I. Saleh, A. H. Rabie, and K. M. Abo-Al-Ez, “A data mining based load forecasting strategy for smart electrical grids,” *Adv. Eng. Informatics*, vol. 30, no. 3, pp. 422–448, Aug. 2016, doi: 10.1016/j.aei.2016.05.005.

[57] A. Jindal, A. Dua, K. Kaur, M. Singh, N. Kumar, and S. Mishra, “Decision Tree and SVM-Based Data Analytics for Theft Detection in Smart Grid,” *IEEE Trans. Ind. Informatics*, vol. 12, no. 3, pp. 1005–1016, 2016, doi: 10.1109/TII.2016.2543145.

[58] M. W. Ahmad, M. Mourshed, and Y. Rezgui, “Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption,” *Energy Build.*, vol. 147, pp. 77–89, 2017, doi: 10.1016/j.enbuild.2017.04.038.

[59] S. Yang, C. Shen, and others, “A review of electric load classification in smart grid environment,” *Renew. Sustain. Energy Rev.*, vol. 24, pp. 103–110, Aug. 2013, doi:

10.1016/j.rser.2013.03.023.

[60] T. Warren Liao, “Clustering of time series data - A survey,” *Pattern Recognit.*, vol. 38, no. 11, pp. 1857–1874, Nov. 2005, doi: 10.1016/j.patcog.2005.01.025.

[61] X. Yang, P. Zhao, X. Zhang, J. Lin, and W. Yu, “Toward a Gaussian-mixture model-based detection scheme against data integrity attacks in the smart grid,” *IEEE Internet Things J.*, vol. 4, no. 1, pp. 147–161, Nov. 2016, doi: 10.1109/JIOT.2016.2631520.

[62] I. Khan, J. Z. Huang, and K. Ivanov, “Incremental density-based ensemble clustering over evolving data streams,” *Neurocomputing*, vol. 191, pp. 34–43, 2016, doi: 10.1016/j.neucom.2016.01.009.

[63] X. Yuan, “An improved Apriori algorithm for mining association rules,” in *AIP Conference Proceedings*, 2017, vol. 1820, no. 1, p. 80005, doi: 10.1063/1.4977361.

[64] M. K. Najafabadi, M. N. ri Mahrin, S. Chuprat, and H. M. Sarkan, “Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data,” *Comput. Human Behav.*, vol. 67, pp. 113–128, 2017, doi: 10.1016/j.chb.2016.11.010.

[65] A. A. Munshi and Y. A. R. I. Mohamed, “Big data framework for analytics in smart grids,” *Electr. Power Syst. Res.*, vol. 151, pp. 369–380, 2017, doi: 10.1016/j.epsr.2017.06.006.

[66] A. El Khaouat and L. Benhlima, “Big data based management for smart grids,” in *Renewable and Sustainable Energy Conference (IRSEC), 2016 International*, 2016, pp. 1044–1047, doi: 10.1109/irsec.2016.7983902.

[67] D. Vohra, *Practical Hadoop Ecosystem*, First Edit. New York: Apress, 2016.

[68] A. B. M. Moniruzzaman and S. A. Hossain, “NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison,” *Int. J. Database Theory Appl.*, vol. 6, no. 4, pp. 1–13, 2013, [Online]. Available: http://arxiv.org/abs/1307.0191.

[69] R. Kumar, B. B. Parashar, S. Gupta, Y. Sharma, and N. Gupta, “Apache Hadoop , NoSQL and NewSQL Solutions of Big Data,” *Int. J. Adv. Found. Res. Sci. Eng.*, vol. 1, no. 6, pp. 28–36, 2014.

[70] G. Liu, W. Zhu, C. Saunders, F. Gao, and Y. Yu, “Real-time Complex Event Processing and Analytics for Smart Grid,” *Procedia Comput. Sci.*, vol. 61, pp. 113–119, 2015, doi: 10.1016/j.procs.2015.09.169.

[71] C. L. Stimmel, *Big data analytics strategies for the smart grid*, First Edit. Auerbach Publications, 2016.

[72] S. Ghemawat, H. Gobioff, and S. T. Leung, “The google file system,” *Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, Dec. 2003, doi: 10.1145/1165389.945450.

[73] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, “A survey of open source tools for machine learning with big data in the Hadoop ecosystem,” *J. Big Data*, vol. 2, no. 1, Nov. 2015, doi: 10.1186/s40537-015-0032-1.

[74] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, “A survey of open source tools for machine learning with big data in the Hadoop ecosystem,” *J. Big Data*, vol. 2, no. 1, p. 24, Nov. 2015, doi: 10.1186/s40537-015-0032-1.

[75] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop distributed file system,” in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, May 2010, pp. 1–10, doi: 10.1109/MSST.2010.5496972.

[76] D. Borthakur, “HDFS architecture guide,” *Hadoop Apache Proj. http//hadoop apache ...*, vol. 53, pp. 1–13, 2008, [Online]. Available: http://archive.cloudera.com/cdh/3/hadoop-0.20.2-cdh3u6/hdfs_design.pdf%5Cnpapers3://publication/uuid/BE03DF70-D0C1-441E-A65F-1888C84992D6.

[77] A. Bahga and V. Madisetti, *Big data science & analytics: A hands-on approach*, First Edit. VPT, 2016.

[78] G. Turkington, T. Deshpande, and S. Karanth, *Hadoop: Data Processing and Modelling*, First Edit. Birmingham: Packt Publishing Ltd, 2016.

[79] V. Kalavri and V. Vlassov, “MapReduce: Limitations, optimizations and open issues,” in *Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013*, Jul. 2013, pp. 1031–1038, doi: 10.1109/TrustCom.2013.126.

[80] Apache Software Foundation, “Apache Storm.” Accessed on: Oct. 2, 2020., [Online]. Available: http://storm.apache.org/.

[81] B. Peng, M. Hosseini, Z. Hong, R. Farivar, and R. Campbell, “R-storm: Resource-aware scheduling in storm,” in *Middleware 2015 - Proceedings of the 16th Annual Middleware Conference*, 2015, pp. 149–161, doi: 10.1145/2814576.2814808.

[82] Apache Software Foundation, “Apache ZooKeeper.” Accessed on: Oct. 2, 2020., [Online]. Available: http://zookeeper.apache.org/.

[83] B. Chambers and M. Zaharia, *Spark, the definitive guide: Big data processing made simple*. “ O’Reilly Media, Inc.,” 2017.

[84] Shyam R., B. Ganesh H.B., S. Kumar S., P. Poornachandran, and Soman K.P., “Apache Spark a Big Data Analytics Platform for Smart Grid,” *Procedia Technol.*,

vol. 21, pp. 171–178, 2015, doi: 10.1016/j.protcy.2015.10.085.

[85] S. Sakr, "Big Data Processing Stacks," *IT Prof.*, vol. 19, no. 1, pp. 34–41, Jan. 2017, doi: 10.1109/MITP.2017.6.

[86] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," 2010. Accessed: Dec. 09, 2020. [Online]. Available: https://www.usenix.org/event/hotcloud10/tech/full_papers/Zaharia.pdf.

[87] H. Zhiwei *et al.*, "Transient power quality assessment based on big data analysis," in *China International Conference on Electricity Distribution, CICED*, Sep. 2014, vol. 2014-Decem, pp. 1308–1312, doi: 10.1109/CICED.2014.6991919.

[88] M. Mayilvaganan and M. Sabitha, "A cloud-based architecture for Big-Data analytics in smart grid: A proposal," in *2013 IEEE International Conference on Computational Intelligence and Computing Research, IEEE ICCIC 2013*, 2013, pp. 1–4, doi: 10.1109/ICCIC.2013.6724168.

[89] A. A. Munshi and Y. A. R. I. Mohamed, "Data Lake Lambda Architecture for Smart Grids Big Data Analytics," *IEEE Access*, vol. 6, pp. 40463–40471, 2018, doi: 10.1109/ACCESS.2018.2858256.

[90] H. Hu, Y. Wen, T. S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014, doi: 10.1109/ACCESS.2014.2332453.

[91] E. Summary, "Challenges and Opportunities with Big Data: A community white paper developed by leading researchers across the United States," Washington, D.C., Mar. 2009.

[92] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 2032–2033, Aug. 2012, doi: 10.14778/2367502.2367572.

[93] D. Haak, "Achieving high performance in smart grid data management," Dublin, Ireland, Accenture, Tech. Rep. WSS141, 2010.

[94] Accenture, "Applying Smart Meter Analytics." Accessed: Jul. 14, 2020, [Online]. Available: https://www.accenture.com/nl-en/blogs/insights/.

[95] Iberdrola, "At the forefront of digital transformation." Accessed: Mar. 7, 2018, 2019, [Online]. Available: https://www.iberdrola.com/about-us/utility-of-the-future/digital-transformation.

[96] Wood Mackenzie Power & Renewables, "Big Data Is Boosting Power Production, Reducing Downtime Across Wind Fleets." Accessed: Jul. 28, 2020, [Online]. Available: https://www.greentechmedia.com/articles/.

[97] S. Johnson, "OpenWay Demand Response Maximizing Value and Efficiency in Energy Delivery," Liberty Lake, Washington, 2010.

[98] L. Hogg, "Business Intelligence for Enterprise Energy Management," Itron, Liberty Lake, WA, USA, Tech. Rep. 100710WP-02, 2007, 2007.

[99] S. Moore, "Key features of meter data management systems," Itron, Liberty Lake, WA, USA, Tech. Rep. 100910WP-01, 2008, 2008.

[100] International Business Machines Corporation, "E.ON and IBM Deliver Innovative Service Offerings to Customers with New Smart Energy Solutions." Accessed: Aug. 7, 2018.

[101] Wood Mackenzie Power & Renewables, "C3's Tom Siebel Opens Up About His Secretive Firm's Smart Grid Data Analytics." Accessed: Jul. 7, 2020, [Online]. Available: https://www.greentechmedia.com/articles/.

[102] Smart Energy International, "KEPCO pilots big data projects for AMI and customer service systems." Accessed on: Oct. 2, 2020., [Online]. Available: https://www.smart-energy.com/regional-news/asia/.

[103] Z. Chen, X. Pei, M. Yang, L. Peng, and P. Shi, "A novel protection scheme for inverter-interfaced microgrid (IIM) operated in islanded mode," *IEEE Trans. Power Electron.*, vol. 33, no. 9, pp. 7684–7697, 2017, doi: 10.1109/tpel.2017.2769559.

[104] A. Zainab, S. S. Refaat, D. Syed, A. Ghrayeb, and H. Abu-Rub, "Faulted Line Identification and Localization in Power System using Machine Learning Techniques," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, Dec. 2019, pp. 2975–2981, doi: 10.1109/BigData47090.2019.9006377.

[105] J. J. Q. Yu, Y. Hou, A. Y. S. Lam, and V. O. K. Li, "Intelligent fault detection scheme for microgrids with wavelet-based deep neural networks," *IEEE Trans. Smart Grid*, vol. 10, no. 2, pp. 1694–1703, Mar. 2019, doi: 10.1109/TSG.2017.2776310.

[106] B. Liu, S. Wu, M. Xie, and W. Kuo, "A condition-based maintenance policy for degrading systems with age- and state-dependent operating cost," *Eur. J. Oper. Res.*, vol. 263, no. 3, pp. 879–887, Dec. 2017, doi: 10.1016/j.ejor.2017.05.006.

[107] X. Wang, S. M. Strachan, S. D. J. McArthur, and J. D. Kirkwood, "Automatic analysis of Pole Mounted Auto-Recloser data for fault diagnosis and prognosis," in *2015 18th International Conference on Intelligent System Application to Power Systems, ISAP 2015*, Nov. 2015, doi: 10.1109/ISAP.2015.7325519.

[108] R. Rocchetta, L. Bellani, M. Compare, E. Zio, and E. Patelli, "A reinforcement learning framework for optimal operation and maintenance of power grids,"

Applied Energy, vol. 241. pp. 291–301, 2019, doi: 10.1016/j.apenergy.2019.03.027.

[109] G. Peng, S. Tang, Z. Lin, and Y. Zhang, “Applications of fuzzy multilayer support vector machines in fault diagnosis and forecast of electric power equipment,” *Proc. 2017 IEEE 2nd Adv. Inf. Technol. Electron. Autom. Control Conf. IAEAC 2017*, pp. 457–461, 2017, doi: 10.1109/IAEAC.2017.8054056.

[110] M. Rafiei, T. Niknam, J. Aghaei, M. Shafie-Khah, and J. P. S. Catalao, “Probabilistic load forecasting using an improved wavelet neural network trained by generalized extreme learning machine,” *IEEE Trans. Smart Grid*, vol. 9, no. 6, pp. 6961–6971, 2018, doi: 10.1109/TSG.2018.2807845.

[111] S. Zhang, Y. Wang, M. Liu, and Z. Bao, “Data-Based Line Trip Fault Prediction in Power Systems Using LSTM Networks and SVM,” *IEEE Access*, vol. 6, pp. 7675–7686, 2017, doi: 10.1109/ACCESS.2017.2785763.

[112] C. Hu, “Ensemble Feature Learning-Based Event Classification for Cyber-Physical Security of the Smart Grid,” 2019. [Online]. Available: https://spectrum.library.concordia.ca/985779/.

[113] D. P. Wadduwage, C. Q. Wu, and U. D. Annakkage, “Power system transient stability analysis via the concept of Lyapunov Exponents,” *Electr. Power Syst. Res.*, vol. 104, pp. 183–192, 2013, doi: 10.1016/j.epsr.2013.06.011.

[114] S. Zadkhast, J. Jatskevich, and E. Vaahedi, “A multi-decomposition approach for accelerated time-domain simulation of transient stability problems,” *IEEE Trans. Power Syst.*, vol. 30, no. 5, pp. 2301–2311, 2014, doi: 10.1109/tpwrs.2014.2361529.

[115] C. He, L. Guan, and W. Mo, “A method for transient stability assessment based on pattern recognition,” *2016 Int. Conf. Smart Grid Clean Energy Technol. ICSGCE 2016*, pp. 343–347, 2017, doi: 10.1109/ICSGCE.2016.7876081.

[116] J. Q. James, Y. Hou, A. Y. S. Lam, and V. O. K. Li, “Intelligent fault detection scheme for microgrids with wavelet-based deep neural networks,” *IEEE Trans. Smart Grid*, vol. 10, no. 2, pp. 1694–1703, Mar. 2017, doi: 10.1109/tsg.2017.2776310.

[117] L. Zhang, X. Hu, P. Li, F. Shi, and Z. Yu, “ELM model for power system transient stability assessment,” in *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, 2017, vol. 2017-Janua, pp. 5740–5744, doi: 10.1109/CAC.2017.8243808.

[118] G. N. Baltas, C. Perales-Gonzalez, P. Mazidi, F. Fernandez, and P. Rodriguez, “A Novel Ensemble Approach for Solving the Transient Stability Classification Problem,” *7th Int. IEEE Conf. Renew. Energy Res. Appl. ICRERA 2018*, pp. 1282–1286, 2018, doi: 10.1109/ICRERA.2018.8566815.

[119] M. Rahmatian, Y. C. Chen, A. Palizban, A. Moshref, and W. G. Dunford, “Transient stability assessment via decision trees and multivariate adaptive regression splines,” *Electr. Power Syst. Res.*, vol. 142, pp. 320–328, 2017, doi: 10.1016/j.epsr.2016.09.030.

[120] J. I. Aizpurua *et al.*, “Probabilistic Power Transformer Condition Monitoring in Smart Grids,” *ARWtr 2019 - Proc. 2019 6th Adv. Res. Work. Transform.*, pp. 42–47, 2019, doi: 10.23919/ARWtr.2019.8930193.

[121] W. L. Woon, Z. Aung, and A. El-Hag, “Intelligent monitoring of transformer insulation using convolutional neural networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11325 LNAI, pp. 127–136, doi: 10.1007/978-3-030-04303-2_10.

[122] H. F. C. Tan, W. L. Woo, A. Sharma, T. Logenthiran, and D. S. Kumar, “Study of Smart Condition Monitoring using Deep Neural Networks with Dropouts and Cross-Validation,” in *2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*, 2019, pp. 3965–3969, doi: 10.1109/isgt-asia.2019.8881423.

[123] M. Žarković and Z. Stojković, “Analysis of artificial intelligence expert systems for power transformer condition monitoring and diagnostics,” *Electr. Power Syst. Res.*, vol. 149, pp. 125–136, 2017, doi: 10.1016/j.epsr.2017.04.025.

[124] N. G. Chothani, M. B. Raichura, D. D. Patel, and K. D. Mistry, “Real-Time Monitoring Protection of Power Transformer to Enhance Smart Grid Reliability,” in *2018 IEEE Electrical Power and Energy Conference, EPEC 2018*, Dec. 2018, doi: 10.1109/EPEC.2018.8598427.

[125] N. Mahmood and S. Yadav, “An Enhanced MPPT Technique by Using Fuzzy Logic Controller,” *J. Multimed. Technol. Recent Adv.*, vol. 6, no. 2, pp. 1–10, 2019, [Online]. Available: http://computers.stmjournals.com/index.php.

[126] G. Gui, H. Pan, Z. Lin, Y. Li, and Z. Yuan, “Data-driven support vector machine with optimization techniques for structural health monitoring and damage detection,” *KSCE J. Civ. Eng.*, vol. 21, no. 2, pp. 523–534, Feb. 2017, doi: 10.1007/s12205-017-1518-5.

[127] W. H. Allen, A. Rubaai, and R. Chawla, “Fuzzy neural network-based health monitoring for HVAC system variable-air-volume unit,” *IEEE Trans. Ind. Appl.*, vol. 52, no. 3, pp. 2513–2524, 2015, doi: 10.1109/tia.2015.2511160.

[128] A. A. Abdoos, P. Khorshidian Mianaei, and M. Rayatpanah Ghadikolaei, “Combined VMD-SVM based feature selection method for classification of power quality events,” *Appl. Soft Comput. J.*, vol. 38, pp. 637–646, 2016, doi: 10.1016/j.asoc.2015.10.038.

[129] P. D. Achlerkar, S. R. Samantaray, and M. S. Manikandan, “Variational mode decomposition and decision tree based detection and classification of power quality disturbances in grid-connected distributed generation system,” *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3122–3132, 2016, doi: 10.1109/tsg.2016.2626469.

[130] Y. Luo, K. Li, Y. Li, D. Cai, C. Zhao, and Q. Meng, “Three-layer Bayesian network for classification of complex power quality disturbances,” *IEEE Trans. Ind. Informatics*, vol. 14, no. 9, pp. 3997–4006, 2017, doi: 10.1109/tii.2017.2785321.

[131] R. Zhu, X. Gong, S. Hu, and Y. Wang, “Power quality disturbances classification via fully-convolutional siamese network and k-nearest neighbor,” *Energies*, vol. 12, no. 24, 2019, doi: 10.3390/en12244732.

[132] Y. Chen, G. Fu, and X. Liu, “Air-Conditioning Load Forecasting for Prosumer Based on Meta Ensemble Learning,” *IEEE Access*, vol. 8, pp. 123673–123682, 2020, doi: 10.1109/ACCESS.2020.2994119.

[133] W. Wang, N. Yu, B. Foggo, J. Davis, and J. Li, “Phase identification in electric power distribution systems by clustering of smart meter data,” in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2016, pp. 259–265, doi: 10.1109/icmla.2016.0050.

[134] R. Jiang, R. Lu, Y. Wang, J. Luo, C. Shen, and X. Shen, “Energy-theft detection issues for advanced metering infrastructure in smart grid,” *Tsinghua Sci. Technol.*, vol. 19, no. 2, pp. 105–120, 2014, doi: 10.1109/TST.2014.6787363.

[135] S. S. S. R. Depuru, L. Wang, and V. Devabhaktuni, “Support vector machine based data classification for detection of electricity theft,” *2011 IEEE/PES Power Syst. Conf. Expo. PSCE 2011*, 2011, doi: 10.1109/PSCE.2011.5772466.

[136] H. Huang, S. Liu, and K. Davis, “Energy Theft Detection Via Artificial Neural Networks,” *Proc. - 2018 IEEE PES Innov. Smart Grid Technol. Conf. Eur. ISGT-Europe 2018*, 2018, doi: 10.1109/ISGTEurope.2018.8571877.

[137] D. Yao, M. Wen, X. Liang, Z. Fu, K. Zhang, and B. Yang, “Energy Theft Detection with Energy Privacy Preservation in the Smart Grid,” *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7659–7669, 2019, doi: 10.1109/JIOT.2019.2903312.

[138] M. Wen, D. Yao, B. Li, and R. Lu, “State Estimation Based Energy Theft Detection Scheme with Privacy Preservation in Smart Grid,” *IEEE Int. Conf. Commun.*, vol. 2018-May, 2018, doi: 10.1109/ICC.2018.8422731.

[139] S. C. Yip, W. N. Tan, C. K. Tan, M. T. Gan, and K. S. Wong, “An anomaly detection framework for identifying energy theft and defective meters in smart grids,” *Int. J. Electr. Power Energy Syst.*, vol. 101, pp. 189–203, 2018, doi: 10.1016/j.ijepes.2018.03.025.

[140] Aasim, S. N. Singh, and A. Mohapatra, “Repeated wavelet transform based

ARIMA model for very short-term wind speed forecasting," *Renew. Energy*, vol. 136, pp. 758–768, Jun. 2019, doi: 10.1016/j.renene.2019.01.031.

[141] A. Samalot, M. Astitha, J. Yang, and G. Galanis, "Combined Kalman filter and universal kriging to improve storm wind speed predictions for the northeastern United States," *Weather Forecast.*, vol. 34, no. 3, pp. 587–601, Jun. 2019, doi: 10.1175/WAF-D-18-0068.1.

[142] F. O. Hocaoglu and F. Serttas, "A novel hybrid (Mycielski-Markov) model for hourly solar radiation forecasting," *Renew. Energy*, vol. 108, pp. 635–643, 2017, doi: 10.1016/j.renene.2016.08.058.

[143] M. Abuella and B. Chowdhury, "Solar power probabilistic forecasting by using multiple linear regression analysis," *Conf. Proc. - IEEE SOUTHEASTCON*, vol. 2015-June, no. June, 2015, doi: 10.1109/SECON.2015.7132869.

[144] L. Cai, J. Gu, J. Ma, and Z. Jin, "Probabilistic wind power forecasting approach via instance-based transfer learning embedded gradient boosting decision trees," *Energies*, vol. 12, no. 1, 2019, doi: 10.3390/en12010159.

[145] R. Azimi, M. Ghayekhloo, and M. Ghofrani, "A hybrid method based on a new clustering technique and multilayer perceptron neural networks for hourly solar radiation forecasting," *Energy Convers. Manag.*, vol. 118, pp. 331–344, 2016, doi: 10.1016/j.enconman.2016.04.009.

[146] A. Zendehboudi, M. A. Baseer, and R. Saidur, "Application of support vector machine models for forecasting solar and wind energy resources: A review," *J. Clean. Prod.*, vol. 199, pp. 272–285, 2018, doi: 10.1016/j.jclepro.2018.07.164.

[147] H. Wang, Z. Lei, X. Zhang, B. Zhou, and J. Peng, "A review of deep learning for renewable energy forecasting," *Energy Convers. Manag.*, vol. 198, 2019, doi: 10.1016/j.enconman.2019.111799.

[148] M. Abdel-Nasser and K. Mahmoud, "Accurate photovoltaic power forecasting models using deep LSTM-RNN," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 2727–2740, Jul. 2019, doi: 10.1007/s00521-017-3225-z.

[149] D. Syed, S. S. Refaat, H. Abu-Rub, O. Bouhali, A. Zainab, and L. Xie, "Averaging Ensembles Model for Forecasting of Short-term Load in Smart Grids," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, Dec. 2019, pp. 2931–2938, doi: 10.1109/BigData47090.2019.9006183.

[150] Y. Wang, Q. Xia, and C. Kang, "Secondary forecasting based on deviation analysis for short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 26, no. 2, pp. 500–507, 2011, doi: 10.1109/TPWRS.2010.2052638.

[151] X. Zheng, X. Ran, and M. Cai, "Short-Term Load Forecasting of Power System based on Neural Network Intelligent Algorithm," *IEEE Access*, pp. 1–1, Sep. 2020,

doi: 10.1109/access.2020.3021064.

[152] D. Syed, S. S. Refaat, H. Abu-Rub, and O. Bouhali, "Short-term Power Forecasting Model Based on Dimensionality Reduction and Deep Learning Techniques for Smart Grid," in *2020 IEEE Kansas Power and Energy Conference, KPEC 2020*, Jul. 2020, doi: 10.1109/KPEC47870.2020.9167560.

[153] D. Syed, S. S. Refaat, and H. Abu-Rub, "Performance evaluation of distributed machine learning for load forecasting in smart grids," *Proc. 30th Int. Conf. Cybern. Informatics, K I 2020*, 2020, doi: 10.1109/KI48306.2020.9039797.

[154] D. Syed, S. S. Refaat, H. Abu-Rub, O. Bouhali, A. Zainab, and L. Xie, "Averaging Ensembles Model for Forecasting of Short-term Load in Smart Grids," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019, Los Angeles, CA, USA*, Dec. 2019, pp. 2931–2938, doi: 10.1109/BigData47090.2019.9006183.

[155] M. Sajjad *et al.*, "A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting," *IEEE Access*, vol. 8, pp. 143759–143768, 2020, doi: 10.1109/ACCESS.2020.3009537.

[156] X. Shao, C. Pu, Y. Zhang, and C. S. Kim, "Domain Fusion CNN-LSTM for Short-Term Power Consumption Forecasting," *IEEE Access*, vol. 8, pp. 188352–188362, 2020, doi: 10.1109/access.2020.3031958.

[157] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 1087–1088, Jan. 2018, doi: 10.1109/TPWRS.2017.2688178.

[158] X. Wang, W.-J. Lee, H. Huang, R. L. Szabados, D. Y. Wang, and P. Van Olinda, "Factors that impact the accuracy of clustering-based load forecasting," *IEEE Trans. Ind. Appl.*, vol. 52, no. 5, pp. 3625–3630, 2016, doi: 10.1109/TIA.2016.2558563.

[159] N. Huang, W. Wang, S. Wang, J. Wang, … G. C.-I., and U. 2020, "Incorporating Load Fluctuation in Feature Importance Profile Clustering for Day-Ahead Aggregated Residential Load Forecasting," *IEEE Access*, vol. 8, no. 1, pp. 25198--25209, 2020, doi: 10.1109/ACCESS.2020.2971033.

[160] Z. Han, M. Cheng, F. Chen, Y. Wang, and Z. Deng, "A spatial load forecasting method based on DBSCAN clustering and NAR neural network," *J. Phys. Conf. Ser.*, vol. 1449, no. 1, p. 12032, 2020, doi: 10.1088/1742-6596/1449/1/012032.

[161] H. H. H. Aly, "A proposed intelligent short-term load forecasting hybrid models of ANN, WNN and KF based on clustering techniques for smart grid," *Electr. Power Syst. Res.*, vol. 182, p. 106191, 2020, doi: 10.1016/j.epsr.2019.106191.

[162] Q. Zhang and J. Zhang, "Short-Term Load Forecasting Method Based on EWT and

IDBSCAN," *J. Electr. Eng. Technol.*, vol. 15, no. 2, pp. 635–644, Mar. 2020, doi: 10.1007/s42835-020-00358-0.

[163] B. Nepal, M. Yamaha, A. Yokoe, and T. Yamaji, "Electricity load forecasting using clustering and ARIMA model for energy management in buildings," *Japan Archit. Rev.*, vol. 3, no. 1, pp. 62–76, 2020, doi: 10.1002/2475-8876.12135.

[164] F. Fahiman, S. M. Erfani, and C. Leckie, *Robust and Accurate Short-Term Load Forecasting: A Cluster Oriented Ensemble Learning Approach*. IEEE, 2019.

[165] L. Wang, S. Mao, B. W.-2019 I. Conference, and U. 2019, "Short-Term Load Forecasting with LSTM Based Ensemble Learning," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, GA, USA*, 2019, doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00145.

[166] G. Sideratos, A. Ikonomopoulos, and N. D. Hatziargyriou, "A novel fuzzy-based ensemble model for load forecasting using hybrid deep neural networks," *Electr. Power Syst. Res.*, vol. 178, no. 1, p. 106025, 2020, doi: 10.1016/j.epsr.2019.106025.

[167] S. Dasgupta, A. Srivastava, J. Cordova, and R. Arghandeh, "Clustering household electrical load profiles using elastic shape analysis," in *2019 IEEE Milan PowerTech, Milano, Italy*, 2019, pp. 1–6, doi: 10.1109/PTC.2019.8810883.

[168] K. P. Sinaga and M.-S. Yang, "Unsupervised K-Means Clustering Algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.

[169] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, 2002, doi: 10.1109/tkde.2002.1033770.

[170] A. K. Srivastava, A. S. Pandey, and D. Singh, "Short-term load forecasting methods: A review," in *2016 International Conference on Emerging Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES)*, 2016, pp. 130–138.

[171] C.-N. Yu, P. Mirowski, and T. K. Ho, "A sparse coding approach to household electricity demand forecasting in smart grids," *IEEE Trans. Smart Grid*, vol. 8, no. 2, pp. 738–748, 2016.

[172] Z. A. Khan and D. Jayaweera, "Approach for forecasting smart customer demand with significant energy demand variability," in *2018 1st International Conference on Power, Energy and Smart Grid (ICPESG)*, 2018, pp. 1–5.

[173] D. Syed, H. Abu-Rub, A. Ghrayeb, and S. S. Refaat, "Household-level Energy Forecasting in Smart Buildings using a Novel Hybrid Deep Learning Model," *IEEE*

Access, 2021, doi: 10.1109/ACCESS.2021.3061370.

[174] D. Syed *et al.*, "Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid With Clustering and Consumption Pattern Recognition," *IEEE Access*, vol. 9, pp. 54992–55008, Apr. 2021, doi: 10.1109/access.2021.3071654.

[175] A. Ghasemi, H. Shayeghi, M. Moradzadeh, and M. Nooshyar, "A novel hybrid algorithm for electricity price and load forecasting in smart grids with demand-side management," *Appl. Energy*, vol. 177, pp. 40–59, 2016.

[176] Y. Yang, W. Hong, and S. Li, "Deep ensemble learning based probabilistic load forecasting in smart grids," *Energy*, vol. 189, p. 116324, 2019.

[177] Y. Lu, T. Zhang, Z. Zeng, and J. Loo, "An improved RBF neural network for short-term load forecast in smart grids," in *2016 IEEE International Conference on Communication Systems (ICCS)*, 2016, pp. 1–6.

[178] Z. Y. Dong, D. J. Hill, and Y. Xu, "Short-Term Residential Load Forecasting based on LSTM Recurrent Neural Network," *ieeexplore.ieee.org*, 2017, doi: 10.1109/TSG.2017.2753802.

[179] A. Zainab, A. Ghrayeb, D. Syed, H. Abu-Rub, S. S. Refaat, and O. Bouhali, "Big data management in smart grids: technologies and challenges," *IEEE Access*, pp. 1–1, 2021, doi: 10.1109/ACCESS.2021.3080433.

[180] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," *J. Mach. Learn. Res.*, vol. 11, no. Feb, pp. 625–660, 2010.

[181] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big data*, vol. 3, no. 1, p. 9, 2016.

[182] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowledge-Based Syst.*, vol. 80, pp. 14–23, 2015.

[183] M. Rohrbach, S. Ebert, and B. Schiele, "Transfer learning in a transductive setting," in *Advances in neural information processing systems*, 2013, pp. 46–54.

[184] Z. Deng, K.-S. Choi, Y. Jiang, and S. Wang, "Generalized hidden-mapping ridge regression, knowledge-leveraged inductive transfer learning for neural networks, fuzzy systems and kernel methods," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2585–2599, 2014.

[185] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 759–766.

[186] M. Ribeiro, K. Grolinger, H. F. ElYamany, W. A. Higashino, and M. A. M. Capretz, “Transfer learning with seasonal and trend adjustment for cross-building energy forecasting,” *Energy Build.*, vol. 165, pp. 352–363, 2018.

[187] A. Hooshmand and R. Sharma, “Energy Predictive Models with Limited Data using Transfer Learning,” in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, 2019, pp. 12–16.

[188] R. Ye and Q. Dai, “A novel transfer learning framework for time series forecasting,” *Knowledge-Based Syst.*, vol. 156, pp. 74–99, 2018.

[189] J. Lago, F. De Ridder, and B. De Schutter, “Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms,” *Appl. Energy*, vol. 221, pp. 386–405, 2018.

[190] A. S. Qureshi, A. Khan, A. Zameer, and A. Usman, “Wind power prediction using deep neural network based meta regression and transfer learning,” *Appl. Soft Comput.*, vol. 58, pp. 742–755, 2017.

[191] A. Gepperth and B. Hammer, “Incremental learning algorithms and applications,” in *2016 European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2016.

[192] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, “Learn++: An incremental learning algorithm for supervised neural networks,” *IEEE Trans. Syst. man, Cybern. part C (applications Rev.*, vol. 31, no. 4, pp. 497–508, 2001.

[193] B. Pérez-Sánchez, O. Fontenla-Romero, B. Guijarro-Berdiñas, and D. Mart\'\inez-Rego, “An online learning algorithm for adaptable topologies of neural networks,” *Expert Syst. Appl.*, vol. 40, no. 18, pp. 7294–7304, 2013.

[194] M. Zribi and Y. Boujelbene, “The neural networks with an incremental learning algorithm approach for mass classification in breast cancer,” *Biomed. Data Min.*, vol. 5, no. 118, p. 2, 2016.

[195] X. Qiu, P. N. Suganthan, and G. A. J. Amaratunga, “Ensemble incremental learning random vector functional link network for short-term electric load forecasting,” *Knowledge-Based Syst.*, vol. 145, pp. 182–196, 2018.

[196] Y. Yang, J. Che, Y. Li, Y. Zhao, and S. Zhu, “An incremental electric load forecasting model based on support vector regression,” *Energy*, vol. 113, pp. 796–808, 2016.

[197] G. Grmanová *et al.*, “Incremental ensemble learning for electricity load forecasting,” *Acta Polytech. Hungarica*, vol. 13, no. 2, pp. 97–117, 2016.

[198] C. M. Bishop, *Pattern recognition and machine learning*, First ed. Springer Science+Business Media, LLC, New York, NY, USA, 2006.

[199] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, First ed. MIT press, Cambridge, MA, USA, 2016.

[200] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," in *Advances in Neural Information Processing Systems, Montréal, Canada*, 2018, pp. 8157–8166, [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/54fe976ba170c19ebae453679b362263-Paper.pdf.

[201] A. Zainab, D. Syed, and D. Al-Thani, "Deployment of deep learning models to mobile devices for spam classification," *Proc. - 2019 IEEE 1st Int. Conf. Cogn. Mach. Intell. CogMI 2019*, pp. 112–117, Dec. 2019, doi: 10.1109/COGMI48466.2019.00024.

[202] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schluter, and H. Ney, "A comprehensive study of deep bidirectional LSTM RNNS for acoustic modeling in speech recognition," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Jun. 2017, pp. 2462–2466, doi: 10.1109/ICASSP.2017.7952599.

[203] D. S. API, "Weather conditions." https://darksky.net/dev (accessed Dec. 31, 2020).

[204] J. Sessa and D. Syed, "Techniques to deal with missing data," in *International Conference on Electronic Devices, Systems, and Applications, Sarawak, Malaysia*, 2017, pp. 1–4, doi: 10.1109/ICEDSA.2016.7818486.

[205] D. Syed, A. Zainab, S. S. Refaat, H. Abu-Rub, and O. Bouhali, "Smart Grid Big Data Analytics: Survey of Technologies, Techniques, and Applications," *IEEE Access*, vol. 9, pp. 59564--59585, Nov. 2020, doi: 10.1109/access.2020.3041178.

[206] N. Zhang, S. L. Shen, A. Zhou, and Y. S. Xu, "Investigation on Performance of Neural Networks Using Quadratic Relative Error Cost Function," *IEEE Access*, vol. 7, pp. 106642–106652, 2019, doi: 10.1109/ACCESS.2019.2930520.

[207] D. Syed, S. S. Refaat, and H. Abu-Rub, "Performance evaluation of distributed machine learning for load forecasting in smart grids," in *Proceedings of the 30th International Conference on Cybernetics and Informatics, K and I 2020*, Jan. 2020, doi: 10.1109/KI48306.2020.9039797.

[208] S. Balasundaram and C. Subhash Prasad, "Robust twin support vector regression based on Huber loss function," *Neural Comput. Appl.*, vol. 32, doi: 10.1007/s00521-019-04625-8.

[209] T. Moshagen, N. A. Adde, and A. N. Rajgopal, "Finding hidden-feature depending laws inside a data set and classifying it using Neural Network," Jan. 2021, Accessed: Jul. 16, 2021. [Online]. Available: https://arxiv.org/abs/2101.10427v1.

[210] D. Ben Or, M. Kolomenkin, and G. Shabat, "Generalized Quantile Loss for Deep

Neural Networks," Dec. 2020, Accessed: Jul. 16, 2021. [Online]. Available: https://arxiv.org/abs/2012.14348v1.

[211] A. S. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, and B. Venkatesh, "Improved short-term load forecasting using bagged neural networks," *Electr. Power Syst. Res.*, vol. 125, pp. 109–115, Aug. 2015, doi: 10.1016/J.EPSR.2015.03.027.

[212] A. Khosravi, S. Nahavandi, and D. Creighton, "Construction of optimal prediction intervals for load forecasting problems," *IEEE Trans. Power Syst.*, vol. 25, no. 3, pp. 1496–1503, Aug. 2010, doi: 10.1109/TPWRS.2010.2042309.

[213] "UCI Machine Learning Repository: ElectricityLoadDiagrams20112014 Data Set," 2015. https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014# (accessed Jul. 06, 2021).

[214] D. Syed, H. Abu-Rub, S. S. Refaat, and L. Xie, "Detection of Energy Theft in Smart Grids using Electricity Consumption Patterns," *Proc. - 2020 IEEE Int. Conf. Big Data, Big Data 2020*, pp. 4059–4064, Dec. 2020, doi: 10.1109/BIGDATA50022.2020.9378190 © 2020 IEEE. Reprinted, with permission, from.

[215] A. Zainab, A. Ghrayeb, M. Houchati, S. S. Refaat, and H. Abu-Rub, "Performance Evaluation of Tree-based Models for Big Data Load Forecasting using Randomized Hyperparameter Tuning," Mar. 2021, pp. 5332–5339, doi: 10.1109/bigdata50022.2020.9378423.

[216] Ö. F. Ertugrul, "Forecasting electricity load by a novel recurrent extreme learning machines approach," *Int. J. Electr. Power Energy Syst.*, vol. 78, pp. 429–435, Jun. 2016, doi: 10.1016/J.IJEPES.2015.12.006.

[217] A. S. Berahas, J. Nocedal, and M. Takáč, "A Multi-Batch L-BFGS Method for Machine Learning."

[218] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, no. 1–3, pp. 503–528, Aug. 1989, doi: 10.1007/BF01589116.

[219] A. Zainab *et al.*, "A Multiprocessing-Based Sensitivity Analysis of Machine Learning Algorithms for Load Forecasting of Electric Power Distribution System," *IEEE Access*, vol. 9, pp. 31684–31694, 2021, doi: 10.1109/ACCESS.2021.3059730.

[220] C. Yuan and H. Yang, "Research on K-value selection method of K-means clustering algorithm," *MDPI J — Multidiscip. Sci. J.*, vol. 2, no. 2, pp. 226–235, 2019, doi: 10.3390/j2020016.

[221] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proceedings*

- 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, May 2017, pp. 464–472, doi: 10.1109/WACV.2017.58.

[222] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[223] C. Efthymiou and G. Kalogridis, “Smart grid privacy via anonymization of smart metering data,” in *2010 first IEEE international conference on smart grid communications*, 2010, pp. 238–243.

[224] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[225] H. Li, “Deep learning for natural language processing: advantages and challenges,” *Natl. Sci. Rev.*, 2017.

[226] H. Gjoreski, J. Bizjak, M. Gjoreski, and M. Gams, “Comparing deep and classical machine learning methods for human activity recognition using wrist accelerometer,” in *Proceedings of the IJCAI 2016 Workshop on Deep Learning for Artificial Intelligence, New York, NY, USA*, 2016, vol. 10.

[227] W. Mackenzie, “Global smart meter total to double by 2024 with Asia in the lead.” Accessed: Oct. 02, 2020, 2020, [Online]. Available: https://www.woodmac.com/news/editorial/.

[228] C. Consulting, “Big Data BlackOut: Are Utilities Powering Up Their Data Analytics?” Accessed on: Oct. 2, 2020., [Online]. Available: https://www.capgemini.com/consulting-no/wp-content/.

[229] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, “Machine learning classification over encrypted data.,” in *NDSS*, 2015, vol. 4324, p. 4325.

[230] R. L. Rivest, L. Adleman, M. L. Dertouzos, and others, “On data banks and privacy homomorphisms,” *Found. Secur. Comput.*, vol. 4, no. 11, pp. 169–180, 1978.

[231] M. Ajtai and C. Dwork, “A public-key cryptosystem with worst-case/average-case equivalence,” in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 284–293.

[232] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[233] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in *International Conference on Machine Learning*, 2016, pp. 201–210.

[234] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, “Fast homomorphic evaluation

of deep discretized neural networks," in *Annual International Cryptology Conference*, 2018, pp. 483–512.

[235] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 267–284.

[236] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Artificial Intelligence and Statistics*, 2012, pp. 933–941.

[237] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with pate," *arXiv Prepr. arXiv1802.08908*, 2018.

[238] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[239] H. Takabi, E. Hesamifard, and M. Ghasemi, "Privacy preserving multi-party machine learning with homomorphic encryption," in *29th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.

[240] R. Gilad-Bachrach, T. W. Finley, M. Bilenko, and P. Xie, "Neural networks for encrypted data." Google Patents, 2018.

[241] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 3, pp. 123–142, 2018.

[242] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami, and N. Shiratori, "New privacy preserving back propagation learning for secure multiparty computation," *IAENG Int. J. Comput. Sci.*, vol. 43, no. 3, pp. 270–276, 2016.

[243] H. Kim, S.-H. Kim, J. Y. Hwang, and C. Seo, "Efficient privacy-preserving machine learning for blockchain network," *IEEE Access*, vol. 7, pp. 136481–136495, 2019.

[244] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[245] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Secur. Priv.*, vol. 17, no. 2, pp. 49–58, 2019.

[246] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[247] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in

Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, 2019, pp. 1–11.

[248] S. Kuri *et al.*, "Privacy preserving extreme learning machine using additively homomorphic encryption," in *2017 IEEE symposium series on computational intelligence (SSCI)*, 2017, pp. 1–8.

[249] X. Ma, F. Zhang, X. Chen, and J. Shen, "Privacy preserving multi-party computation delegation for deep learning in cloud computing," *Inf. Sci. (Ny).*, vol. 459, pp. 103–116, 2018.

[250] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[251] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2010, pp. 24–43.

[252] M. Brenner *et al.*, "A Standard API for RLWE-based Homomorphic Encryption," HomomorphicEncryption.org, Redmond WA, USA, Jul. 2017.

[253] Y. Tsiounis and M. Yung, "On the security of ElGamal based encryption," in *International Workshop on Public Key Cryptography*, 1998, pp. 117–134.

[254] N. Patel, P. Oza, and S. Agrawal, "Homomorphic Cryptography and Its Applications in Various Domains," in *International Conference on Innovative Computing and Communications*, 2019, pp. 269–278.

[255] K. El Makkaoui, A. Ezzati, and A. Beni-Hssane, "Securely adapt a Paillier encryption scheme to protect the data confidentiality in the cloud environment," in *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, 2016, pp. 1–3.

[256] S. Halevi and V. Shoup, "Algorithms in helib," in *Annual Cryptology Conference*, 2014, pp. 554–571.

[257] W. Dai and B. Sunar, "cuHE: A homomorphic encryption accelerator library," in *International Conference on Cryptography and Information Security in the Balkans*, 2015, pp. 169–186.

[258] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "{TFHE}: Fast Fully Homomorphic Encryption Library." .

[259] C. Aguilar-Melchor, J. Barrier, S. Guelton, A. Guinet, M.-O. Killijian, and T. Lepoint, "NFLlib: NTT-based fast lattice library," in *Cryptographers' Track at the RSA Conference*, 2016, pp. 341–356.

[260] K. Laine and R. Player, “Simple encrypted arithmetic library-seal (v2. 0),” *Tech. report, Tech. Rep.*, 2016.

[261] E. Crockett and C. Peikert, “$Λ$o$λ$: Functional Lattice Cryptography,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 993–1005.

[262] Y. Polyakov, K. Rohloff, and G. W. Ryan, “PALISADE lattice cryptography library user manual,” *Cybersecurity Res. Center, New Jersey Inst. ofTechnology (NJIT), Tech. Rep*, 2017.

[263] T. Takagi and T. Peyrin, *Advances in Cryptology--ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings*, vol. 10625. Springer, 2017.

[264] “Lattigo 1.3.1.” Accessed on: Oct. 2, 2020, Feb. 2020, [Online]. Available: http://github.com/ldsec/lattigo.

[265] W. Li, D. Deka, M. Chertkov, and M. Wang, “Real-time Faulted Line Localization and PMU Placement in Power Systems through Convolutional Neural Networks,” *IEEE Trans. Power Syst.*, 2019.

[266] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(Leveled) fully homomorphic encryption without bootstrapping,” *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[267] J. Domingo-Ferrer, “A provably secure additive and multiplicative privacy homomorphism,” in *International Conference on Information Security*, 2002, pp. 471–483, doi: 10.1007/3-540-45811-5_37.

[268] E. U. Soykan, Z. Bilgin, M. A. Ersoy, and E. Tomur, “Differentially Private Deep Learning for Load Forecasting on Smart Grid,” in *2019 IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–6, doi: 10.1109/gcwkshps45667.2019.9024520.

APPENDIX A

PRIVACY PRESERVATION OF DATA-DRIVEN MODELS IN SMART GRIDS USING HOMOMORPHIC ENCRYPTION*[11]

A.1 Overview

The information layer embedded in the two-way communication electric smart grid has various data sources such as Advanced Metering Infrastructure (AMI), smart electrical measurement sensors, smart meters, detectors, Phasor Measurement Units (PMU), Remote Terminal Unit (RTU), and Supervisory Control and Data Acquisition System (SCADA) [11]. These sources generate huge amounts of data that satisfy the volume, velocity, variety, veracity, and value characteristics of big data [40]. The data generated from multiple sources are located at different nodes of an electric network. However, electric utilities demand to preserve the security and privacy of the collected and used data. The collected data represents the behavior of the customers and other network players. Those data contain sensitive information about customers, electricity consumption, trading, and operation of electricity distribution networks [223]. Those data can be utilized to develop machine learning algorithms to improve the electrical utility operation in terms of demand response, peak load shaving, and fault analysis.

Machine learning algorithms can be classified into two divisions; classical and deep learning techniques [224]. There are merits and demerits of each division compared to the other. Deep learning techniques have the merits of being more accurate, scalable

[11] Reprinted with permission from "Privacy Preservation of Data-Driven Models in Smart Grids Using Homomorphic Encryption." by Dabeeruddin Syed, Shady S. Refaat, and Othmane Bouhali, 2020. Information Journal 11, no. 7, 357, Copyright 2020 by Dabeeruddin Syed.

with more data, no requirement of complex feature engineering, adaptability, and transferability to different domains and applications through transfer learning. [225]. The classical machine learning techniques have merits of higher performance if the size of available data is small, lower requirements of computational resources, and simplicity [226]. However, all the machine learning approaches require the data which are collected for processing from several nodes in the electric network. The data are transmitted from the electrical nodes to a unified central system to develop the data-driven models. This data transmission and collection by model developers endanger the privacy and security of the information. Hence, security approaches such as homomorphic encryption and federated machine learning, that avoid the movement of plain data can be considered as potential solutions for securing the collected data. In recent years, many researchers have attained noteworthy progress in the field of encryption schemes.

In the past decade, the rate of generation of energy data from utilities has increased exponentially. The number of smart meters installed globally is around 660 million in 2017 with the data generated at a rate of 280 petabytes a year [227] [228]. Furthermore, the number of people who have access to the data has increased multi-fold. For business benefits, each of the teams of an entity has data scientists in-house who develop machine learning models to recognize the patterns in the data and to discover ways to improve business. This brings the security of the confidential information in the collected data at risk and may lead to sensitive information disclosure. It has been impartially assumed for decades that the encryption garbles up the numbers in the data and no practical mathematical operations can be executed on the encrypted data [229]. One of the earliest approaches to investigate the possibility of performing computations on the underlying data without actually having access to the unencrypted forms of data was presented by Rivest, Adleman, and Dertouzos in 1978 through their work entitled privacy homomorphisms [230]. In 1996, Ajtai et al. proposed the use of lattice-based constructions as an encryption scheme to develop a public-key cryptosystem [231]. The foundation of these works has set in motion the advancement of first working Fully Homomorphic Encryption (FHE) in 2009 [232]. The FHE scheme allows for the mathematical operations and function mapping to be employed on encrypted data.

The HE schemes have been applied with classical machine learning models initially and later, implemented with deep learning models. Earlier works on encryption schemes with deep learning models presented works with activation function as high degree polynomials whose computations are slow and require huge resources. The possibility of low degree non-linear polynomials (such as square function) as activation functions in the network layers was presented in [233]. Also, HE schemes allow the execution of non-polynomial functions on encrypted data. Later, bootstrapping procedure and use of sign function were introduced in later work with an aim to keep the scheme complexity linear to the depth of the neural networks [234]. These developments enable the possibility of use of, HE schemes in the real-world scenarios for encryption.

In this work, the homomorphic encryption model is applied for two smart grid case studies namely, fault identification and localization data from the simulated IEEE 68 bus system in order to develop highly secure and accurate deep learning model for fault localization predictions, and real distribution transformer energy consumption data to develop machine learning model for load forecasts. The main contributions of this work are classified into the following main categories:

1. Designing a secure and privacy-preserving deep neural network (DNN) model established on homomorphic encryption for smart grid applications.
2. Proposing a fault classification and localization model that has favorable accuracy considering the fact that the model is trained on encrypted data. The encryption of the data before modeling accentuates the need for data privacy and security. The accuracy of the model trained on encrypted data is very close to the model trained on non-encrypted data.
3. Proposing a load forecasting regression model such that the accuracy of the predictions using encrypted data is close to accuracy using non-encrypted data.

A.2 State-of-the-art works

In this section, we first present the security and privacy foundations required for secure machine learning. Then, the possible solutions to achieve the security foundations are discussed in the subsequent parts. At last, we discuss the literature review and previous works in the field of privacy-preservation of machine learning models using homomorphic encryption.

The computational research community has been interested in privacy-preserving machine learning to ensure that the data remains secure during all the stages of its processing i.e., from the training stage to the predictions stage. There are four main pillars needed to achieve privacy-preserving machine learning: namely, training data privacy, model input privacy, model weights privacy, and model output privacy. They are shown in Figure 54 and summarized as next.

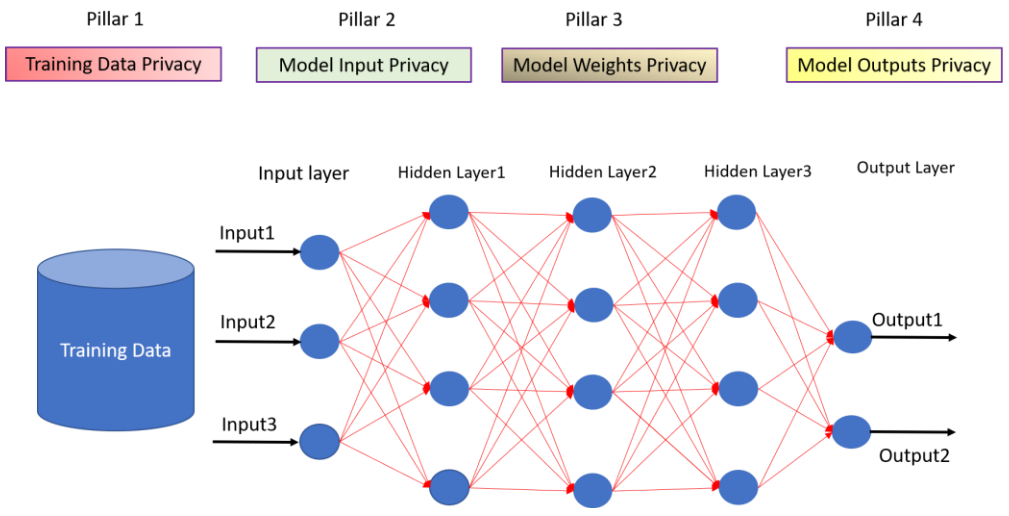


Figure 54 Four Pillars of Privacy-Preserving Machine Learning

1. Training Data Privacy: The privacy and security layer should be in place that no malicious agent can reverse engineer the training data from the model or the output.
2. Model Input Privacy: The input data cannot be obtained by any third party, including the model creator; the input data owner is assumed as different than the model creator.
3. Model Weights Privacy: The model parameters and weights cannot be obtained or inferred by a malicious party.
4. Model Output Privacy: The output of the model cannot be observable by any third party, except for the owner of the data.

Hence, there are multiple challenges that need to be addressed to achieve the aims of the four pillars of privacy in machine learning which is the goal of our paper. The following subsections discuss various approaches to privacy-preserving machine learning.

A.2.1 Training Data Privacy

Although it is difficult to reverse engineer the training data from the model parameters and model outputs, it is not impossible. In [235], the authors have indicated the leakage of information on training data using reverse engineering on the machine learning models. The results of their work have indicated that the generative sequence models can retain rare information from the raw data and that this memorization peaks when the test loss is set to the minimum.

To prevent the memorization of training data in the developed models, there are two major proposed solutions; Differentially Private Stochastic Gradient Descent (DPSGD) [236], and Private Aggregation of Teacher Ensembles (PATE) [237]. These solutions not only provide security to the data but also improve the generalizability of the machine learning models.

The memorization and exposure of private knowledge from training data can be reduced with the use of DPSGD. The DPSGD solution adapts differential privacy into SGD to conserve the security of training data while developing deep learning models [236].

PATE is a scalable alternative to DPSGD and is an ensemble model. The components of the ensemble model train on the independent and identically distributed (i.i.d.) subsets of the same dataset. If most of the models in the ensemble have the same output, then it can be inferred that the models and their output do not expose any secure information from the training data and hence, can be shared [237].

A.2.2 Model Input Privacy and Model Output Privacy

Input data and the outputs (predicted variable values) from the data should be accessible only to the owners of the data and protected from other parties including the model developers. There are three solutions in the literature that have been successfully applied to preserve the model input and output privacy. Those solutions are briefly discussed below:

1. Federated Learning (FL) [238]: FL is machine learning on device. It can be made secure with the use of differentially private stochastic gradient descent.
2. Homomorphic Encryption (HE) [239]: Homomorphic encryption makes provision for the use of non-polynomial functions on encrypted data. With this capability of homomorphic encryption, it is possible to apply classical machine learning algorithms such as linear or logistic regression, naive bayes, and random forest and deep learning models on encrypted data for training and obtain predictions [240]. However, initial works such as CryptoNets [240] have a limitation of high latency. Also, the cryptonets do not support the popular activation functions (Relu, and Sigmoid) and the pooling functions (Max Pooling). The limitations of the use of activation functions were later overcome in [241] which approximates the continuous functions of Sigmoid, Relu, and Tanh to lower degree polynomials based on Chebyshev polynomials. Their results indicated that the replacement of

activation function with approximated lower degree polynomials adopts neural networks to be effectively used with homomorphic encryption.

3. Secure Multiparty Computation (MPC) [242]: When multiple parties are involved, they can decide on functions to calculate outputs using their private inputs. The inputs are not revealed or exposed. The concept of secure MPC has been successfully used in generative models and machine learning algorithms to protect the data from reverse engineering.

A.2.3 Model Weights Privacy

The model privacy is very crucial for companies who own data to avoid having their AI applications and models easily copied or reverse engineered from inputs and outputs. Hence, model privacy and model weights' privacy are extremely crucial.

The FL, HE, and MPC solutions could also be used to enhance the model privacy. The solutions for model weights privacy include the following:

1. Differentially private stochastic gradient descent
2. Homomorphic Encryption

A.2.4 Related Work

Multiple methodologies have been utilized in the discipline of privacy-preserving machine learning [243] [244] [245]. These methodologies include, but are not limited to, differential privacy [246], federated machine learning [247], homomorphic encryption [248], and multi-party computation [249]. In this work, we utilize the homomorphic encryption as it achieves the four pillars of privacy-preserving machine learning for smart grid applications.

In [241], Hesamifard et al. proposed a framework called CryptoDL that aims to preserve the security and privacy of the training data and the classification predictions generated by the Machine Learning (ML) models especially Neural Networks (NN). The authors have applied CryptoDL to different open datasets with encouraging and accurate results. The framework accepts the training data in encrypted form, develop models, and generate classification predictions which are encrypted under the data owner's public key as well. Their experiments yield that the proposed approach of approximating activation functions to low-degree Chebyshev polynomials outperforms other HE methodologies and the developed models preserve the security of data.

In [250], Shokri et al. designed a privacy system that allows multiple parties to train the deep learning models on their private data and share the knowledge from their learned models. The sharing of the models improves the overall accuracy of the final averaged model while also preserving the data privacy because individual data is not shared between the multiple parties. Their system is based on Distributed Selective Stochastic Gradient Descent (DSSGD) and parameter exchange protocol for different parameters of the deep learning model.

In [229], the authors developed the classification models using decision trees, hyperplane decision, and naive bayes classifiers on encrypted data using security constraints. However, the accuracy of their classification models has not been reported in the article. The efficiency of the models is evaluated in terms of classification time and they reported that their models take a few milliseconds to a few seconds on large datasets.

A stable and significant implementation of fully homomorphic encryption was presented in [232]. The author presented a modular framework of FHE that supports the computations to a fixed depth. Then, he employed a bootstrap method to enhance the framework with successful computations to a larger depth with a few constraints or assumptions. The decryption works correctly if the noise in the encrypted text is small. For example, $c_1 \rightarrow Enc(p_1)$ and $c_2 \rightarrow Enc(p_2)$ are cipher texts and these encrypted data have noise n_1, n_2. Then, during addition and multiplication operations, these noise values increase to $n_1 + n_2$ and $n_1 \cdot n_2$ respectively. Hence, if the noise is significant after encryption of data, this methodology performs well only in shallow networks. The technique of bootstrapping is used to reduce the noise in the network after encryption as it refreshes the cipher data. Consider $c_1 \in Enc(p_1)$ has large noise and a helper cipher data $c_k \rightarrow Enc(secretkey)$. A homomorphic evaluation of the decrypting function is performed as $Eval(Dec, Enc(c_k), c_1)$ which refreshes the cipher data which encrypts $Dec(c_k, secretkey) = p_1$. This bootstrapping technique is used to refresh the cipher data and removes the noise in it. The cipher data are first calculated up to depth 'd' which is efficient for fully homomorphic encryption. Then, bootstrapping is used to refresh the cipher data which are devoid of noise. Finally, the encryption is performed starting from depth 'd' to depth '2d', and bootstrapping is then applied at depth '2d' and so on. Dijk et al. [251] simplified the complexity of Gentry's somewhat homomorphical bootstrappable encryption scheme and applied it over the integers.

There are a few technical challenges associated with the privacy-preservation of data [252]. These are listed as below:

- The data might come from multiple sources that use different secret keys for encryption. In such a case, the homomorphic encryption is not straight-forward. The solution to this will be MPC and HE.
- Also, developing deep learning models with homomorphic encryption may be hard when compared to developing shallow models. However, in this work, the developed deep learning models display high accuracy on testing.

There are multiple types of schemes in HE depending on the operations they can perform. The HE encryption schemes that can perform only one type of operation are called partially homographic encryption schemes. El Gamal scheme [253] and RSA scheme [254] are partially HE schemes, and these can perform only multiplication operations homographically whereas Paillier scheme [255] can perform addition operations homographically.

MPC is similar to homomorphic encryption with an exception that the two users involved X (user with data d) and Y (developer of function f) are required to interact over multiple iterations to train the model f(d). For specific applications, MPC has proven to perform better than Homomorphic encryption. However, MPC faces the difficulties of high bandwidth requirements and network latencies. Hence, HE is more scalable for generalized smart grid applications.

In practice, there are multiple open-source implementations of homomorphic encryption schemes. These libraries are described in Table 29.

Table 29 Open-source implementations of HE.

Implementation	**Description**

HELib [256]	The low-level library implements HE with faster evaluation time employing optimized Brakerski-Gentry-Vaikuntanathan (BGV) scheme with bootstrapping.
cuHE [257]	highly optimized GPU-accelerated library for Homomorphic encryption.
TFHE [258]	open-source gate-by-gate bootstrapping library which evaluates homomorphic encryption of binary gates, negation, and MUX gate operations and performs computation over encrypted data.
NFLlib [259]	open-source Number Theoretic Transform based Fast Lattice Library which uses low-level processor functionalities.
SEAL [260]	an extensively employed open-source library from Microsoft that supports BGV and Cheon, Kim, Kim, and Song (CKKS) encryption schemes.
$\boldsymbol{\Lambda o \lambda}$ [261]	open-source Haskell library for functional lattice-based cryptography.
PALISADE [262]	Open-source library for implementations of lattice-based encryption building blocks and HE scheme.
HeaAN [263]	Open-source implementation of HE encryption scheme using approximate arithmetic of numbers.
Lattigo [264]	library for lattice-based cryptography and MPC, written in Golang (Go) language.

Our researched system attains significant security objectives in the context of deep learning models applied for smart grid applications. It secures the training data before it is transferred to the model developers. The model developers are enabled to control the learning objectives of the data-driven deep learning models. The solution allows for the application of machine learning and deep learning models along with generalization to a wide variety of electrical applications.

A.3 Proposed Methodology for Privacy-Preservation of Data-driven Models

This section presents the proposed methodology for the application of deep learning over encrypted data with the homomorphic encryption method. The main objective of this methodology is to add additional security and privacy layer to deep learning models in smart grid electrical applications. As observed in Figure 55, the encryption keys are used to encrypt the data on the server-side. Only the encrypted data is communicated to the client-side or the model developers. The model is trained over the encrypted data, then the predictions are obtained. Moreover, the predictions are encrypted which requires decryption performed at the server side to obtain the final predictions. It is important to note that the client or model developer does not have access to the encryption keys, therefore the client can neither decrypt the data nor decrypt the predictions. In some applications, the client is provided with a previously trained model and encrypted test data to provide encrypted predictions back to the server. In such cases, the previously trained model may have been trained on plain data on the server-side. The homomorphic encryption allows for the privacy, reliability, and security of training data, model inputs, weights, and outputs. However, the HE does not secure the data from the point of generation to the server, but only after it reaches the server. The framework assumes that

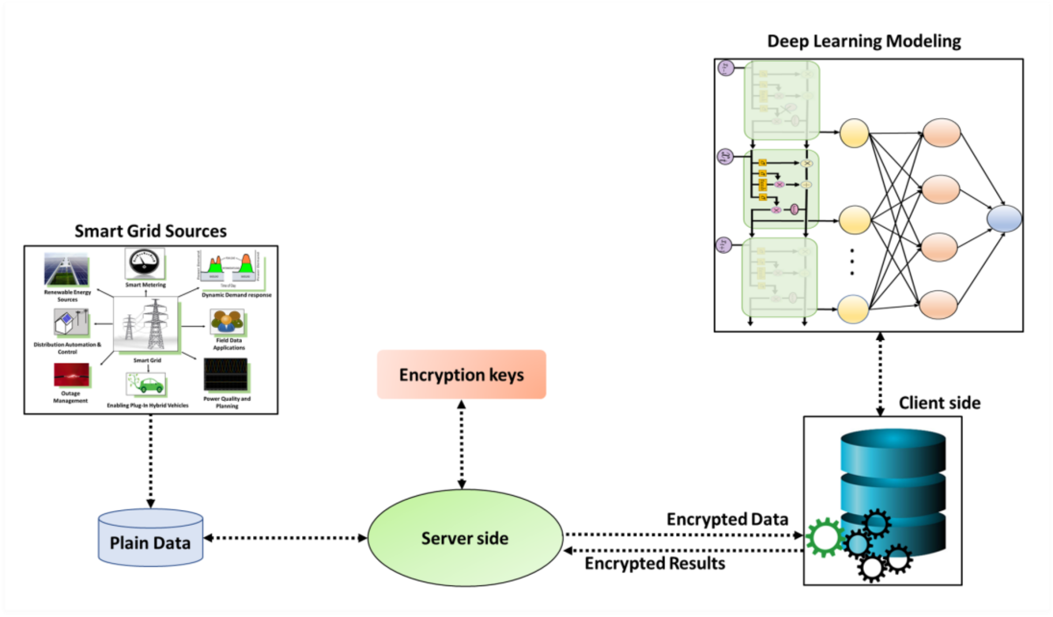


Figure 55 Homomorphic encryption-based deep learning modeling for smart grid applications

the smart grid platform supports secure data acquisition from different data sources. The diverse storage services and policies should enforce privacy and security between the points of data generation and server.

In the following sections, the main steps of the proposed methodology are described.

A.3.1 Data description

Two datasets have been collected for evaluating the performance of machine learning models employing homomorphic encryption. One of the datasets is the simulation data for fault localization in a power system network and another dataset is the time-series load demand data at the distribution grid level (transformers). The following section provides the description of the two datasets and the pre-processing steps employed on the datasets.

A.3.1.1 Dataset 4

The data utilized is the fault localization simulation data from the IEEE 68 bus system which consists of 68 buses, 16 machines, 16 generators, and 20 transformers [104] [265]. Figure 56 shows the reduced-order 68-bus system test simulation model. The simulation data is PMU measurements acquired for pre and during fault conditions for a subset of the grid buses in the system. The extracted features which characterize the location and occurrence of the faults are determined to be the bus voltage variations before and during the faults. The final feature set which is given as input to the classifier is given by the imaginary part of the following (51).

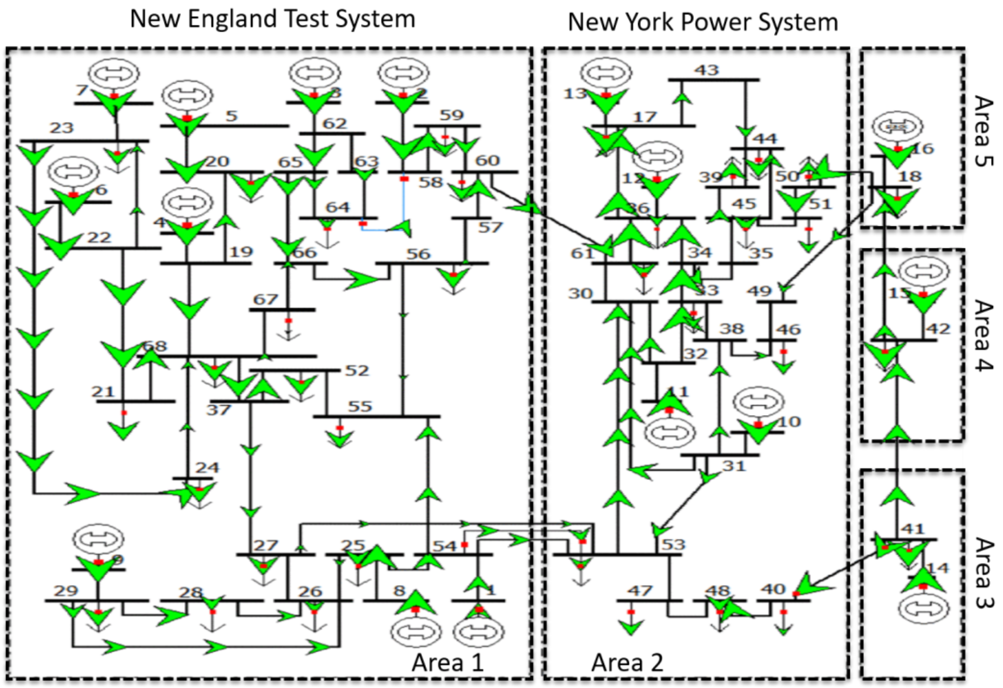


Figure 56 IEEE 68-bus system test simulation model

$$\Psi = Y^0 \Delta V \tag{51}$$

where Y^0 denotes the admittance matrix of the bus system before the faults and ΔV represents the difference in the bus voltage before and during the faults.

The label of the data indicates the line number at which the fault occurs and hence, there are 86 classes. 75% of the data is utilized as a training data set, and 25% of the data is utilized as a validation data set.

A.3.1.2 Dataset 5

The second case study is on the load forecasting at the distribution level. The data acquired is the time series hourly load demand data at the distribution transformers level for a real power grid between January 2018 to December 2018. After data acquisition, the only features were time stamp, transformer id, season, and hourly load demand values. In the pre-processing steps, the weather data is extracted and added as features to this dataset using a freemium Application Programming Interface (API) called Darksky [203]. The extracted features include maximum temperature, minimum temperature, cloud cover, dew point, humidity, precipitation intensity, pressure, Ultraviolet rays (UV) index, visibility, wind gust, and wind speed. This extraction introduces missing values that are filled using an average of forward and backward fill for numerical features [149]. Also, the 24 lag hour values are fed back to the dataset as additional features.

A.3.2 Homomorphic encryption

The fully homomorphic encryption or simply homomorphic encryption is a category of encryption methodology which differs from other classical methods in a way that it enables the computations to be executed on the encrypted data without having a requirement to access the secret encryption key [266]. The output of such computations is also encrypted, and it can be decrypted with the help of a secret key that the data owner possesses. A function $f: R_{\mathbb{1}} \longrightarrow R_{\mathbb{2}}$ is said to be additive and multiplicative homomorphic, if for every $r_1, r_2 \in R_{\mathbb{1}}$, it implies that $f(r_1 + r_2) = f(r_1) \oplus f(r_2)$ and $f(r_1 \cdot r_2) = f(r_1) \otimes f(r_2)$ respectively, where $\oplus$, and $\otimes$ are the operations in $R_{\mathbb{2}}$ [267].

A homomorphic encryption technique has a supplemental algorithm property called $Eval$ which can be executed or computed over encrypted data. Any party can run $Eval$ function on the encrypted data without requiring access to the private key with which data is initially encrypted. That is, the ciphertext need not be decrypted to allow computations on it in the evaluation function, and this maintains the privacy of the underlying information in the encrypted data. Figure 57 illustrates the different mapping functions and computations involved in Homomorphic Encryption Evaluation.

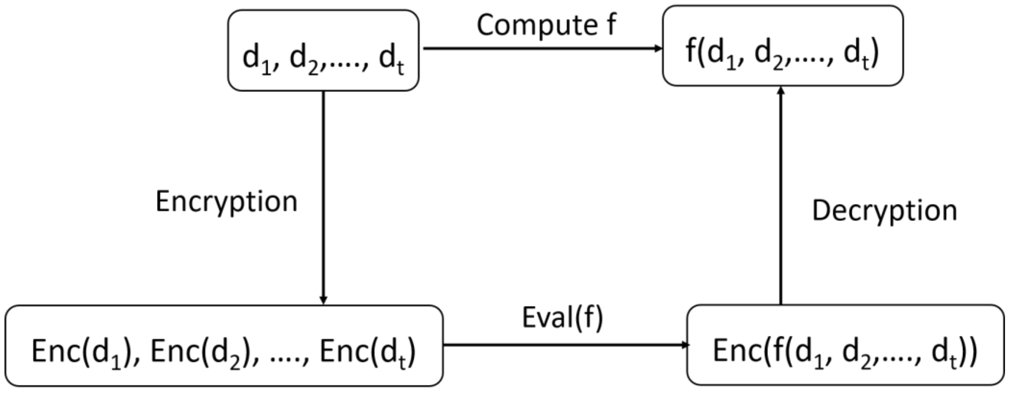

$d_1, d_2, \ldots, d_t$
Compute f
$f(d_1, d_2, \ldots, d_t)$
Encryption
Decryption
$Enc(d_1), Enc(d_2), \ldots, Enc(d_t)$
Eval(f)
$Enc(f(d_1, d_2, \ldots, d_t))$

Figure 57 Homomorphic Encryption Evaluation

The homomorphic evaluation of a deep neural network with discretized inputs and weights involves two steps mainly. These steps are enumerated as following:

1. Computation of multisum between the encrypted inputs that are given to the neurons and the discretized weights at the respective neurons: The calculation of multisum uses homomorphic addition as basic operation.
2. Extraction of the sign of the output at each neuron.

To make the neural network scalable in terms of the number of layers, the bootstrapping operation is performed at every neuron in the layers to effectively encrypt the sign of the output and use it in further computations to the next layers in the network.

For privacy-preserving machine learning, we use the orthogonal matrix transformations-based homomorphic encryption in the classification case study. The secret key of the proposed matrix transformations based HE scheme is an invertible matrix (U1) of size m, where m is the number of records in training data. And Paillier cryptosystem is utilized for the case study of regression.

A.3.2.1 Encryptions of the data

The data is encrypted using HE algorithm based on matrix transformations. The aim of the encryption of the data is to generate highly accurate deep learning or machine learning models and use the encrypted data for training or testing so that the model developers do not have access to the plain data all the time.

In the first case study for fault localization, two sets of simulations have been conducted. In the first set, the DNN model is developed when the training data is plain, and the model developer has access to the plain data. Here, there are no privacy-preserving techniques involved. In the second set of experiments, the first stage encrypts the fault identification and localization data using homomorphic encryption on the source side. The data provided to the model developers are the encrypted version of data. So, no information from the data can be leaked. The model developers develop the DNN model using the encrypted data and return the encrypted predictions to the source side. The data owner then uses the keys to decrypt the encrypted predictions to obtain the final predictions. These predictions are used to calculate the accuracy of the developed models using ground truth values.

Algorithms 6, 7, 8, and 9 present the encryption and decryption steps that are used on the server-side in the fault localization case study.

Algorithm 6. Encryption of Train Data

```
1:  function ENCRYPTIONTRAIN($(\boldsymbol{X})_{\boldsymbol{mxn}}$, $(\boldsymbol{y})_{\boldsymbol{mx1}}$)
2:      Choose $(Q_1)_{mxm}$ as orthogonal matrix.
3:      if $n > 1$ then
4:          Choose orthogonal $Q_2$ of size $n$
5:      else
6:          Choose orthogonal $Q_2$ of size 1
7:      end if
8:      $(X)_{encrypted} = Q_1^T.X.Q_2$
9:      $(y)_{encrypted} = Q_1^T.y$
10:     return $(X)_{encrypted}$, $(y)_{encrypted}$, $Q_1$, $Q_2$
11: end function
```

Algorithm 7. Decryption of Train Data

```
1:  function DECRYPTIONTRAIN($X$, $y$, $Q_1$, $Q_2$)
2:      Choose $(Q_1)_{mxm}$ as orthogonal matrix.
3:      $(X)_{decrypted} = Q_1^T.X.Q_2^{-1}$
4:      $(y)_{decrypted} = Q_1^T.y$
5:      return $(X)_{decrypted}$, $(y)_{decrypted}$
6:  end function
```

Algorithm 8. Encryption of Test Data

```
1:  function ENCRYPTIONTEST($X$, $Q_2$)
2:      Choose $(Q_1)_{mxm}$ as orthogonal matrix.
3:      if $noOfRows(X) > 1$ then
4:          Choose orthogonal $Q_3$ of size $noOfRows(X)$
5:      else
6:          Choose orthogonal $Q_2$ of size 1
7:      end if
8:      $(X)_{encrypted} = Q_3.X.Q_2^{-1}$
9:      return $(X)_{encrypted}$, $Q_3$
10: end function
```

Algorithm 9. Decryption of Test Data

```
1:  function DECRYPTIONTEST($y_{encrypted}, Q_3$)
2:      $(y)_{decrypted} = Q_3^{-1} . y_{encrypted}$
3:      return $(y)_{decrypted}$
4:  end function
```

In the second case study for load demand forecasting, a machine learning model is developed using the plain data where the model developers have initial access to data. The knowledge gained i.e., model and its parameters can be transferred to points where load demand predictions are required. At these points, access to the plain data is not required and is encrypted during the training phase. In this work, a paillier encryption scheme [255] is employed and public, and private key pair is generated. The test data is encrypted using a public key and the predictions are generated using the model on encrypted testing data. The load forecast values are encrypted and these can be decrypted only with the help of private key which is available only to the data owner.

The homomorphic encryption can secure a multitude of electrical applications in smart grids by providing the following advantages:

- securing data stored in cloud server.
- enabling data analytics in regulated electric utilities.
- HE protects the systems against eavesdropping attacks after the data has reached the server. The HE potentially renders any data leaked through eavesdropping attacks or man-in-the-middle attacks indecipherable to attackers.
- HE can protect the data against unauthorized sharing.

A.4 Results

This section presents the efficacy of the proposed homomorphic encryption- based deep learning models for fault localization and detection. Also, the results of another case study of Paillier scheme for load demand predictions are presented.

The experiments have been conducted utilizing Python 3.7 for encryption and decryption schemes. The execution was run on a computer with an Intel Core i7-7700HQ CPU @ 2.80 GHz with eight logical processors, four physical cores, and 16 GB of RAM.

A.4.1 Case Study 1

The machine learning model utilized to train on plain and encrypted data is the deep neural network (DNN). It comprises of three categories of layers – an input layer, an output layer, and hidden layers. The inherent non-linear characteristics of the layers enable the DNNs structure to recognize and generalize the underlying patterns and information in the data.

The size of the input layer of DNN is 64 and the size of the output layer is 86. The number of neurons in each hidden layer is 70 neurons. The activation function utilized in the DNN model trained on plain and encrypted data is hyperbolic tangent (tanh) activation. The input space is scaled to the range of (-1, 1) and the tanh activation function ranges between the same values. It is also maintained that the weights have discrete values over the domain of Z as real-valued weights render the use of homomorphic encryption incompatible. A dropout percentage of 20% is utilized for the model trained on plain data. A dropout percentage of 50% is utilized for the model trained on encrypted data.

Table 30 Evaluation of homomorphic encryption deep learning model for classification problem (fault localization).

Model Name	DNN	HE+DNN

Validation accuracy	98.32	97.71
Test accuracy	99.98	98.59
Execution time (s)	20.61	23.64
Computation complexity of activation function	O(1)	O(1)

As shown in Table 30, the performance of the deep learning model on encrypted data is close to the performance of the model on plain data. The validation accuracy of the DNN model on plain data of fault identification and localization is 98.32%. However, when the DNN model is trained on the encrypted data, the validation accuracy is 97.71%. Also, a separate fault simulation data is used as a test data set. And, on the test data set, the accuracy of DNN model trained on plain data is 99.98% whereas the accuracy of the model trained on encrypted test data is 98.59%. The graphical representation of the results is shown in Figure 58. Homomorphic encryption based DNN takes more time, but it effectively protects the data and generates a comparative accurate model.

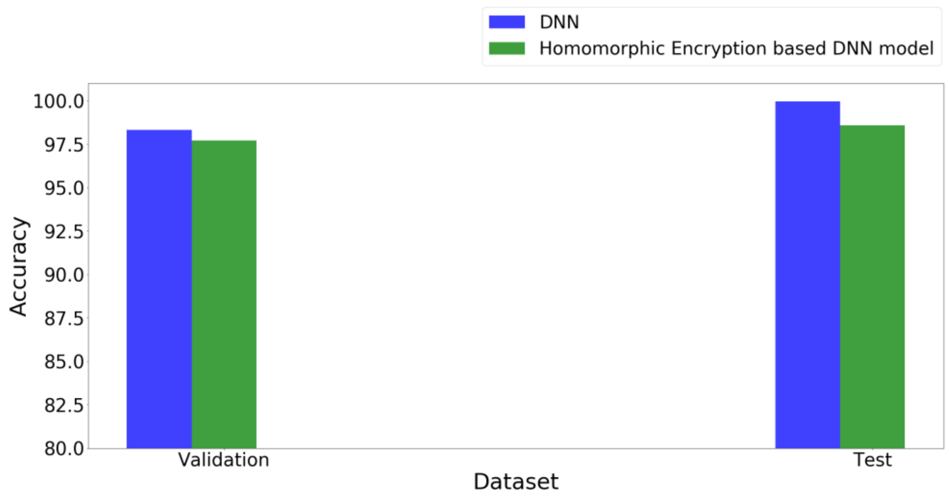


Figure 58 Accuracy-wise performance of the proposed methodology

A.4.2 Case Study 2 (a)

The classical machine learning algorithm of linear regression is applied over the Paillier scheme for the case study on load demand predictions. Mini-batch Gradient descent optimizer is used with learning rate 0.01 and training epochs as 1000 as the data is big and the computational memory is limited. Training of the model is performed on unencrypted data which is then transferred to the encrypted testing data to make encrypted predictions. Only the server-side private key can be used to decrypt the predictions. However, a public key can be used to encrypt any data. 80% of data is used for training and 20% of data is used as a validation dataset.

Figure 59 depicts the training history in terms of mean square error against the number of epochs of training. The smoothness in the graph indicates that the cost function of the machine learning model keeps on decreasing over the set number of epochs and the model approaches to convergence with minimum prediction error.

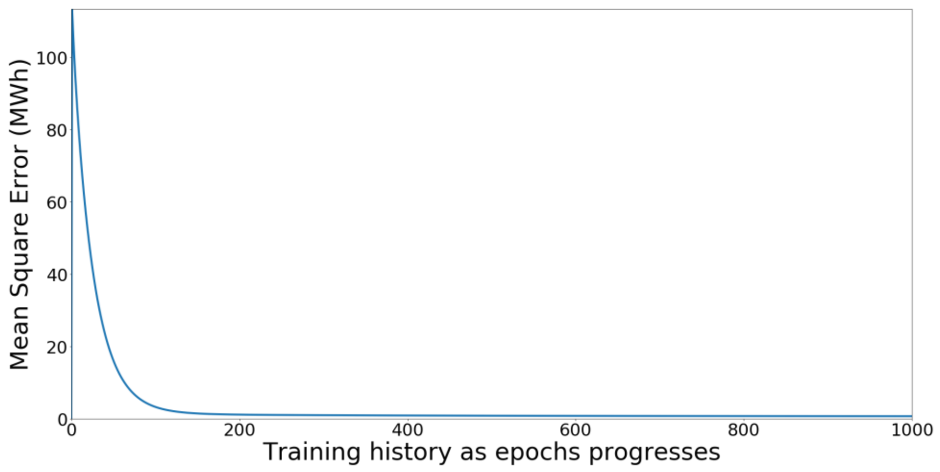


Figure 59 The mean square error as training progresses

Figure 60 plots the predicted values of load demand in testing data set after decryption against the true values of load demand. A point on the straight line through origin would indicate that there is no error in the predicted value, and it is the same as the actual load demand value. The homomorphic encryption-based modeling was thus evaluated for the regression problem of load demand forecasting. The results of the evaluation are displayed in Table 31. The results indicate that the machine learning model without encryption has a coefficient of variation (CV) of 7% and the CV is around 10% when encryptions are employed.

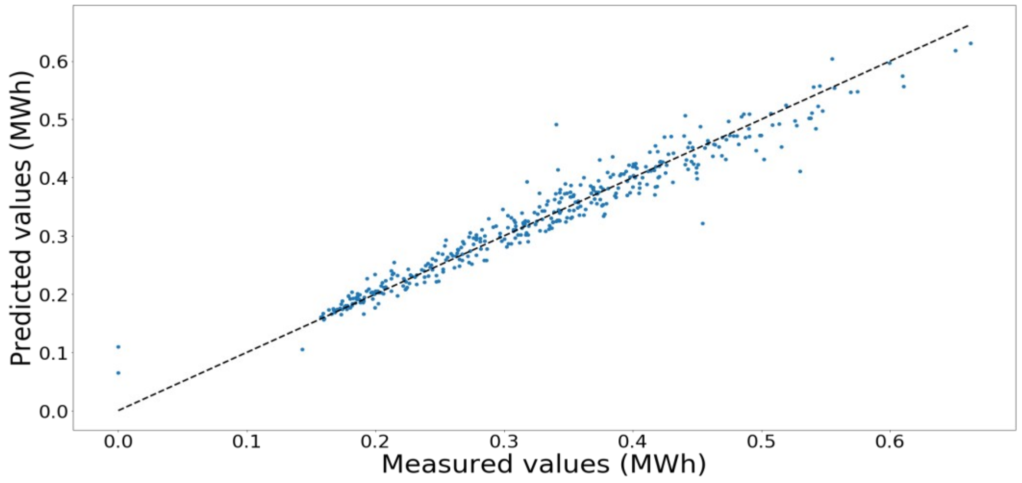


Figure 60 Predicted and actual values of load demand

Table 31 Evaluation of homomorphic encryption for regression problem (load forecasting).

Model Name	**Validation dataset**		**Test dataset**	
	RMSE (MWh)	**Coefficient of Variation (CV) (%)**	**RMSE (MWh)**	**CV (%)**
LR	0.0248	7.49	0.0250	7.55
HE+LR	0.0352	10.63	0.0374	11.30

A.4.3 Case Study 2 (b)

In this set of experiments, a similar methodology is applied on a different hourly load forecasting scenario with an aim to compare the results of our methodology to existing work. The public data set is provided by PJM Interconnection LLC and consists of hourly energy consumption data recorded by various distribution utilities in the Eastern Interconnection grid in the United States. We utilized the data from the EIC grid for area AECO between the periods of March 1, 2019 and June 19, 2020. The time-series data contains 8713 records. In the feature engineering step, twenty-four lag hour values of energy consumption are added as features to the data set. 90% of the data is employed for training and 10% of data is used as a testing data set.

Table 32 presents the results of our experiments. As observed in the Table 32, the first forecasting experiment is performed without any encryption on the data, and this is a baseline to relatively compare the results of encryption-based forecasting.

Table 32 HE for load forecasting problem - PJM dataset.

Model Name	**Testing dataset**	
	RMSE (MWh)	**MAPE (%)**
LR	28.18	2.007
HE+LR	31.65	6.15

As per the results in Table 32, when the encryption-based methodology is applied, MAPE is obtained as 6.15%. For the same data set, without any encryption involved, the load forecasting MAPE is 2.007%. As the security of data is increased using encryption, the forecasting accuracy is definitely decreased. However, we still obtain high accuracy with a high degree of privacy. The decrease in accuracy, after our proposed methodology is applied, is around 4.15% as against the 8.69% decrease in accuracy as reported by previous work [268] in the same load forecasting scenario.

A.5 Insights

In this paper, a privacy-preserving deep learning model based on homomorphic encryption is presented for classification and a classical machine learning model based on the Paillier scheme for homomorphic encryption is presented for regression. The proposed model can achieve machine learning modeling in the encrypted domain. The simulations of the model on different datasets indicate that the performance on encrypted data is as accurate as modeling on plain data. Application of the proposed methodology in the smart grids data processing is promising and implies a wide range of real-life electrical utility applications. Hence, data privacy can be maintained by the use of encryption and the machine learning models can be trained on encrypted data and still achieve the same accuracy and training. The main contribution of this work is a data-driven method that benefits from the accuracy of classical and deep machine learning algorithms with the lucidity of employing homomorphic encryption. The amalgamation of algorithms from cryptography, machine learning, and electrical engineering has demonstrated a methodology that achieves accuracy and security of electrical data while maintaining high throughput for computations. However, there is a scope of improvement in terms of computation time by using Graphics Processing Units (GPUs). Also, more effective encoding paradigms can be employed that require fewer parameters and obtain accelerated computations.

APPENDIX B

LIST OF PUBLICATIONS

B.1 Patent

1. Dabeeruddin Syed, Ameema Zainab, Haitham Abu-Rub, Shady S. Refaat, Othmane Bouhali, Ali Ghrayeb, Mahdi Houchati and Santiago Bañales Lopez. "Multi-stage Distributed Big Data Analytics Platform for Faster Processing of Heterogeneous Data Sources in Smart grids". Patent Filed (06/14/2021).

B.2 Published/Accepted Journal Papers

2. Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Shady S. Refaat, Mahdi Houchati, Othmane Bouhali, and Santiago Bañales. "Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid with Clustering and Consumption Pattern Recognition." IEEE Access 9 (2021): 54992–55008. doi:10.1109/access.2021.3071654.
3. Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, and Shady S. Refaat. "Household-Level Energy Forecasting in Smart Buildings Using a Novel Hybrid Deep Learning Model." IEEE Access 9 (2021): 33498–33511. doi:10.1109/access.2021.3061370.
4. Dabeeruddin Syed, Ameema Zainab, Ali Ghrayeb, Shady S. Refaat, Haitham Abu-Rub, and Othmane Bouhali. "Smart Grid Big Data Analytics: Survey of Technologies, Techniques, and Applications." IEEE Access 9 (2021): 59564–59585. doi:10.1109/access.2020.3041178.
5. Dabeeruddin Syed, Shady S. Refaat, and Othmane Bouhali. "Privacy Preservation of Data-Driven Models in Smart Grids Using Homomorphic Encryption." Information Journal 11, no. 7 (2020): 357. doi:10.3390/info11070357.
6. Zainab Ameema, Dabeeruddin Syed, Ali Ghrayeb, Haitham Abu-Rub, Shady S. Refaat, Mahdi Houchati, Othmane Bouhali, and Santiago Banales Lopez. "A Multiprocessing-Based Sensitivity Analysis of Machine Learning Algorithms for

Load Forecasting of Electric Power Distribution System." IEEE Access 9 (2021): 31684–31694. doi:10.1109/access.2021.3059730.

7. Zainab Ameema, Ali Ghrayeb, Dabeeruddin Syed, Haitham Abu-Rub, Shady S. Refaat, and Othmane Bouhali. "Big Data Management in Smart Grids: Technologies and Challenges." IEEE Access 9 (2021): 73046-73059. doi: 10.1109/access.2021.3080433.

B.3 Accepted Abstract for Journal Special session

8. Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Othmane Bouhali, Shady S. Refaat, Mahdi Houchati, "Cloud-based Big Data Analytics for Load Forecasting Targeting Demand Response". In IEEE Transactions on Cloud Computing [Accepted Abstract]

B.4 Published book chapters

9. Mohammed, Amira, and Dabeeruddin Syed. "Cloud Computing for Smart Grid." Smart Grid and Enabling Technologies (July 30, 2021): 333–357. doi:10.1002/9781119422464.ch14.

B.5 Published/Accepted conference papers

10. Dabeeruddin Syed, Haitham Abu-Rub, Ameema Zainab, Mahdi Houchati, Othmane Bouhali, Ali Ghrayeb, and Shady S. Refaat. "Investigation on Optimizing Cost Function to Penalize Underestimation of Load Demand through Deep Learning Modeling". In 47th Annual Conference of the IEEE Industrial Electronics Society. [Accepted]
11. Dabeeruddin Syed, Shady S. Refaat, Haitham Abu-Rub, Othmane Bouhali, Ameema Zainab, and Le Xie. "Averaging Ensembles Model for Forecasting of Short-term Load in Smart Grids." In 2019 IEEE International Conference on Big Data (Big Data), pp. 2931-2938. IEEE, 2019. doi:10.1109/bigdata47090.2019.9006183.

12. Dabeeruddin Syed, Haitham Abu-Rub, Shady S. Refaat, and Le Xie. " Detection of Energy Theft in Smart Grids using Electricity Consumption Patterns." 2020 IEEE International Conference on Big Data (Big Data) (December 10, 2020). doi:10.1109/bigdata50022.2020.9378190.
13. Dabeeruddin Syed, Shady S. Refaat, and Haitham Abu-Rub. "Performance evaluation of distributed machine learning for load forecasting in smart grids." In 2020 Cybernetics & Informatics (K&I), pp. 1-6. doi:10.1109/ki48306.2020.9039797.
14. Dabeeruddin Syed, Shady S. Refaat, Haitham Abu-Rub, and Othmane Bouhali. "Short-term Power Forecasting Model Based on Dimensionality Reduction and Deep Learning Techniques for Smart Grid." In 2020 IEEE Kansas Power and Energy Conference (KPEC), pp. 1-6. doi:10.1109/kpec47870.2020.9167560.
15. Zainab, Ameema, Shady S. Refaat, Dabeeruddin Syed, Ali Ghrayeb, and Haitham Abu-Rub. "Faulted Line Identification and Localization in Power System using Machine Learning Techniques." In 2019 IEEE International Conference on Big Data (Big Data), pp. 2975-2981, 2019. doi:10.1109/bigdata47090.2019.9006377.

B.6 Submitted Journal papers (under review)

16. Dabeeruddin Syed, Haitham Abu-Rub, Shady S. Refaat, Othmane Bouhali, Ali Ghrayeb, Ameema Zainab, Mahdi Houchati, and Santiago Bañales Lopez, "Enhancement of the Performances of Cross-model Power Forecasting in Smarts Grids using Transfer Learning". In IEEE Systems Journal [Submitted, Under Review].

B.7 Papers to be submitted

17. Dabeeruddin Syed, Haitham Abu-Rub, Ali Ghrayeb, Shady S. Refaat, "Enhancing Short Term Load Forecasting using partitioned NN with recurrent connections". In IEEE Access [In pipeline]