Computing and Informatics, Vol. 42, 2023, 75–97, doi: 10.31577/cai_2023_1_75

SHORT TERM LOAD FORECASTING FOR SMART GRIDS USING APACHE SPARK AND A MODIFIED TRANSFORMER MODEL

Aditya Upadhyay, Dhruv Garg

Computer Science and Engineering Department
Thapar University
Patiala (Punjab), India
e-mail: adityaupadhyay1912@gmail.com, dhruvgarg01@gmail.com

Mukesh Singh

Electrical and Instrumentation Engineering Department
Thapar University
Patiala (Punjab), India
e-mail: mukesh.singh@thapar.edu

Abstract. Smart grid is an advanced electrical grid that enables more efficient distribution of electricity. It counters many of the problems presented by renewable energy sources such as variability in production through techniques like load forecasting and dynamic pricing. Smart grid generates massive amounts of data through smart meters, this data is used to forecast future load to adjust distribution. To process all this data, big data analysis is necessary. Most existing schemes use Apache Hadoop for big data processing and various techniques for load forecasting that include methods based on statistical theory, machine learning and deep learning. This paper proposes using Apache Spark for big data analysis and a modified version of the transformer model for forecasting load profiles of households. The modified transformer model has been tested against several state-of-the-art machine learning models. The proposed scheme was tested against several baseline and state-of-the-art machine learning models and evaluated in terms of the RMSE, MAE, MedAE and R^2 scores. The obtained results show that the proposed model has better performance in terms of RMSE and R^2 which are the preferred metrics when evaluating a regression model on data with a large number of outliers.

Keywords: Apache Spark, big data, deep learning, load forecasting, smart grid, time-series forecasting, transformer model

Mathematics Subject Classification 2010: 68-04

1 INTRODUCTION

Traditional electric grids face many issues such as variable consumption and electrical outages. A smart grid (SG) is an electrical grid that can counter these issues in a cost effective manner with minimal loss. SG balances the gap between power supply and demand more efficiently than traditional electric grids. In an SG, data is collected from several sources like smart meters, sensors and external sources such as weather reports and demographics. Smart meters measure electricity consumption and communicate it to energy suppliers [1]. One million smart meters installed in an SG with a sampling rate of 4 times per hour can generate 35.04 billion records, which is equivalent to 2 920 terabytes of data in quantification [2]. This results in generation of huge amount of data which needs to be processed, analysed and used to make accurate load predictions for the SG to efficiently allocate energy. This is where two major issues in an SG arise. These issues are processing the massive generated data and making accurate load predictions using forecasting techniques.

SG data cannot be processed using standard data processing techniques as the data is massive, very wide in scope and has a high degree of variance. Therefore, preprocessing and cleaning the data becomes essential before it can be used for making predictions. As discussed earlier, data of such massive size cannot be effectively managed using traditional methods, instead big data analysis has to be utilized. Big data is a relative concept and no absolute threshold has been defined. It can be understood as an amount of data beyond the standard technology's capability to store, manage and process efficiently because of its sheer size or complexity [3]. The data collected from an SG can be classified as big data on the basis of 5 V's which are Volume, Velocity, Variety, Veracity and Value [4].

Smart meters record terabytes and exabytes of measurement data every single day. Furthermore, approximately 220 million smart meter measurements are recorded daily in a large SG [1]. Data generation in an SG is also a continuous streaming process. This makes the volume and velocity of data generation very high as data is recorded in small intervals. Moreover, data collected can be present in many different formats e.g. time-series uni-variate load data, multivariate weather data and census data [1]. Therefore, the data exhibits a high degree of variety. With such large amounts of data integration, its quality and accuracy becomes less trustworthy which increases its veracity [4]. Furthermore, since the data obtained is very massive, the density of valuable information will be low. As all the 5 V's are satisfied, the data obtained from SG can be classified as big data. Using a big data framework, the data is then preprocessed and cleaned. The data thus obtained is then used to create the energy consumption prediction models.

Predicting energy consumption of every node in an SG is very important for balancing power generation and filling the demand response gap. In order to predict load consumption, energy demand patterns of the consumers have to be estimated. These demand patterns are not always consistent since there are many unpredictable external factors that can influence load consumption. Therefore, to make accurate predictions, the shared uncertainties between multiple households, such as houses in similar neighbourhoods, need to be extracted along with several highly non-linear relations between the input features, like size of a household [5]. This makes load forecasting a fairly difficult task. In the past, several approaches have been proposed for load forecasting. These approaches have mainly been divided into two categories which are conventional and machine learning methods. This paper mainly focuses on deep learning methods which is a subset of machine learning methods and proposes a novel modified transformer model architecture. Most of the methods used for big data analysis and prediction models have been highlighted in the next section.

1.1 Related Works

A big data processing framework is required to effectively handle and process data in an SG. [6] and [7] used the Apache Hadoop framework to handle big data because of MapReduce and Hadoop Distributed File System (HDFS). [8] and [9] used the Apache Spark framework for clustering and real-time big data processing. Apache Spark was chosen by them because of its parallel data processing capabilities. [10] offer a detailed comparison of the Spark and Hadoop frameworks on remote sensing and [11] compare Spark and Hadoop for real-time processing. Both papers conclude that Spark is the better platform for real-time data. It is faster than Hadoop but more resource-intensive, requiring far more RAM than Hadoop for effective usage. After reviewing this information, the Apache Spark framework was chosen for data pre-processing in this paper.

After the appropriate pre-processing and feature selection, time-series models are used for load forecasting. These models can be categorized into two types – conventional models and machine learning models. The most popular amongst the conventional models is auto-regressive integrated moving average (ARIMA). [12] compared ARIMA with support vector machine (SVM) which is a machine learning model for load forecasting. They concluded that SVM has lower overall loss but ARIMA obtains better results when predicting peaks. [13] used ARIMA and autoregressive integrated moving average with exogenous variables (ARIMAX) and concludes that ARIMAX makes more accurate predictions than ARIMA. [14] combined differential evolution and Support Vector Regression (SVR) to find the most optimum model parameters for load forecasting. [15] combined the two and took advantage of ARIMA for predicting linear basic part of load and used SVM to forecast the non-linear sensitive part of load.

Papers using deep learning for load forecasting generally use Long Short-Term Memory (LSTM) or Gated Recursive Unit (GRU). [16] used an LSTM model for short-term load forecasting and compared it with several machine learning models. They concluded it to have better results than the other models. [17] used a bi-directional LSTM and [18] used an A-LSTM model which is a modified LSTM model with added attention mechanism and both obtained higher accuracy than standard LSTM models. [19] introduced a new model (EGA-STLF) based on bidirectional gated recursive unit model (Bi-GRU) and attention mechanism. It was found to perform better than most other models, such as KNN and ELM, in terms of accuracy and efficiency. [20] compared various models such as vanilla recurrent neural network (RNNs), RNN-LSTM and RNN-GRU with seq2seq (S2S) models. They found that S2S models with a Bahdanau attention mechanism outperforms all other models in terms of MAPE and MAE. [21] proposed using a transformer model for short term load forecasting and found it to have excellent performance compared to baseline models. [5] used a novel deep RNN model which outperformed several state-of-the-art models used for household load forecasting like ARIMA, SVR and classic RNN. Based on this information, several baseline models for comparison with the proposed model were chosen.

[22] used an LSTM model with pinball loss instead of Mean Square Error (MSE) to predict the long-term and short-term dependencies within the load profiles. [23] made a novel architecture by combining Convolutional Neural Network (CNN) and RNN architectures to predict load. [24] compared various statistical methods such as ARIMA and deep learning methods like Artificial Neural Networks (ANN) and proposes a dynamic pricing scheme using the results. [25] proposed a novel factored conditional restricted Boltzmann machine (FCRBM) optimized using a genetic wind-driven optimization algorithm (GWDO) with a modified mutual information (MMI) technique used for feature selection. They validated the model by comparing it against state-of-the-art mutual information ANNs, fast converging ANNs and LSTM models. [26] proposed using an auto-encoder only for feature extraction and Random Forest for the actual forecasting. [27] proposed combining the Fruit Fly Optimization algorithm (FOA) and a Generalized Regression Neural Network (GRNN) to solve this problem.

A new deep learning model called the transformer model was first proposed by [28] for performing NLP tasks which gave excellent results. It has since become the standard model for most NLP tasks. Recently, modified versions of the transformer model have been found to perform very well for time-series forecasting as shown by [29]. The transformer model proposed by them has performed better than baseline time-series forecasting models such as LSTM, ARIMA and S2S+Attention in terms of Root Mean Square Error (RMSE). The transformer model provides a much higher degree of parallelism, uses the attention mechanism and also considers the previous and next state before making predictions similar to bi-directional models. Due to these reasons, this paper proposes a modified version of the transformer model. To the best of our knowledge, this model has not been used for SG load forecasting in the past.

1.2 Motivation

As discussed in the previous section there are many ways to handle big data and forecast load consumption in an SG however most of them lack the efficiency and accuracy to be implemented practically. Most papers on SG load forecasting utilize Apache Hadoop as their big data framework, however it is unsuitable for SG big data. In terms of forecasting models, conventional methods such as ARIMA tend to lack accuracy in predictions. While, deep learning models although more accurate in forecasting load consumption, require a lot of time and resources to obtain results that are accurate enough to be implemented. The most popularly used deep learning models are LSTM and GRU, which have sequential processing that results in long training times and also assume that every state is only dependent on previous states. In contrast, the transformer model provides a much higher degree of parallelism and also introduces self-attention mechanism which assumes a particular state can be dependent on states both before and after that state. Keeping these issues in mind, this paper proposes a modified transformer model coupled with the Apache Spark framework for SG load forecasting and big data analysis.

1.3 Contribution

The primary contribution of this paper is to present an approach for load forecasting using big data analysis in an SG that is efficient and accurate in making predictions.

- This paper proposes a heavily modified version of the transformer model, which was originally made for NLP tasks, that has now been altered to work for time series tasks.
- This modified model is used alongside Apache Spark for fast computation and accurate results.
- This model uses uni-variate time series data and it provides a much higher degree of parallelism than other deep learning time-series forecasting models.

1.4 Organization

The rest of the paper is organized as follows. Section 2 gives the brief description about the working of the proposed scheme. Section 3 elaborates the working of the modified transformer model. The results and discussions are presented in Section 4. The paper is finally concluded in Section 5.

2 PROPOSED SCHEME

This section illustrates the working of the proposed scheme for SG load forecasting. This paper has used smart meter energy consumption data in London Households which contains load data collected from 5 567 London households between November 2011 and February 2014. As shown in Figure 1, this data is passed through several data pre-processing stages. First, the data is cleaned by removing outliers and null values. Next, data is reduced by removing the redundant details. This data is then normalised by scaling it between 0 and 1. After the pre-processing, the dataset is split into train and test dataset. The chosen deep learning model learns on the train dataset observations. The trained model is then evaluated on test dataset. The model can then be used to forecast consumption data. The data obtained from an SG is massive and a big data framework is needed to effectively complete the pre-processing steps. Therefore, Apache Spark has been used for data pre-processing.

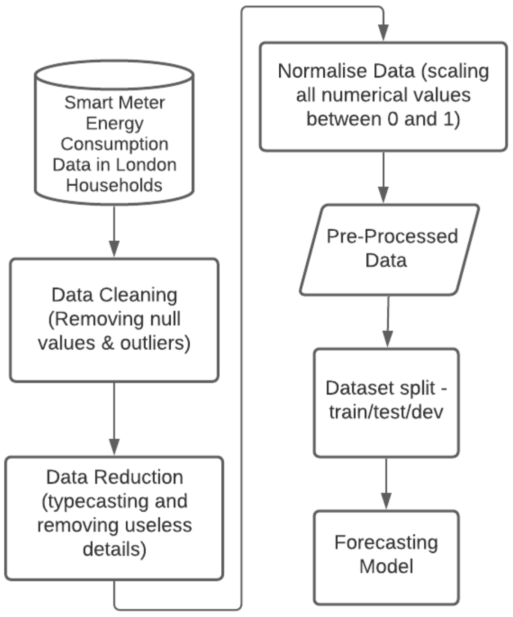

Smart Meter Energy Consumption Data in London Households
Data Cleaning (Removing null values & outliers)
Data Reduction (typecasting and removing useless details)
Normalise Data (scaling all numerical values between 0 and 1)
Pre-Processed Data
Dataset split - train/test/dev
Forecasting Model

Figure 1. Flowchart of data pre-processing

Apache Spark was chosen for many reasons. Firstly, it is capable of real-time processing which makes it preferable for SG data processing. This is because most of the data in an SG is generated and stored in real-time. Spark can handle real-time data very efficiently using spark streaming. Other frameworks such as

Hadoop would require the use of a different framework such as Apache Storm to handle real-time data. Furthermore, Apache Spark is also capable of parallel processing, with automated load distribution to prevent load imbalance, and uses an in-memory data engine. Both of these features make processing faster than MapReduce which is used by Hadoop and most other frameworks. Spark achieves parallel processing using Resilient Distributed Datasets (RDDs), as shown in Figure 2. An RDD is a programming abstraction that represents a collection of objects that can be split across a computing cluster. This split allows it to process multiple instances of data in parallel. Apache Spark also includes Spark MLib, which allows us to construct a pipeline to feed incoming data into a deep learning model. Using a similar pipeline, the pre-processed data is fed into the modified transformer model.

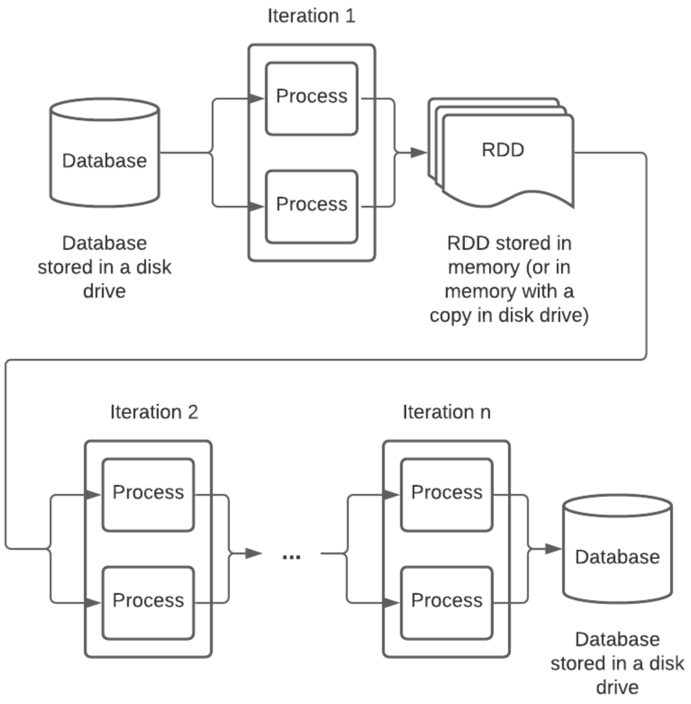

Iteration 1
Database
Process
Process
RDD
Database stored in a disk drive
RDD stored in memory (or in memory with a copy in disk drive)
Iteration 2
Iteration n
Process
Process
...
Process
Process
Database
Database stored in a disk drive

Figure 2. Working of RDDs in Apache Spark

The detailed architecture of the transformer model is depicted in Figure 3. The transformer model follows an encoder-decoder structure using stacked self-attention and fully connected layers for both encoder and decoder. The data is input to the transformer model in a sequence format with 0s in place of unknown values. It first passes through an embedding layer, which acts as a linear transformation layer. The input sequence is then passed through the encoder. The encoder layers consists of a series of stacked self-attention and fully connected layers. The attention layers give us the weighted relation between values in the input sequence with respect to each other. The input sequence is then encoded using the weights obtained. The output of encoder is passed to the decoder. Inside the decoder layers, the inputs will first be masked because the modified transformer model generates a completely new sequence using only the values that appear before it in the sequence. The decoder layers are very similar to the encoder layers with just an added component. The decoder layer also contains an encoder-decoder attention layer which takes in outputs from decoder self-attention layer and also encoder outputs, then finds the weighted relations between the two. The output sequence is then encoded using the weights obtained. The decoder output then passes through a linear layer with sigmoid activation which converts embedding to the actual value and gives us the new sequence containing predictions for all the unknown values. All of these layers have been explained in detail in Section 3.

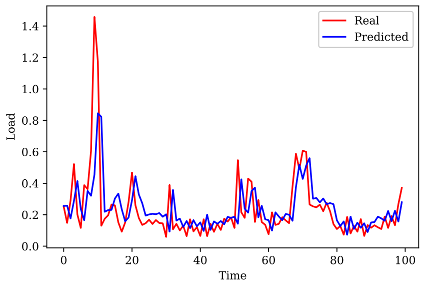


Figure 3. Modified transformer model architecture

3 MATHEMATICAL MODELLING OF MODIFIED TRANSFORMER MODEL

The modified transformer follows the overall architecture as shown in Figure 3 using stacked self-attention and point-wise fully connected layers for both the encoder and decoder. Encoder and decoder are shown in the left and right halves of Figure 3, respectively. The main purpose of the modified transformer model is to map an input sequence in the format $(x_1, x_2, \dots, x_{ts-1}, x_{ts})$ to a newly generated output sequence such as $(y_1, y_2, \dots, y_{ts-1}, y_{ts})$, where ts is the timestep at which the load is to be predicted and y_{ts} is the predicted load. In this case, the first 24 timesteps (i_1 to i_{ts-1}) followed by a 0 are passed as input $(i_1, i_2 \dots, i_{ts-1}, 0)$. The passed 0 is replaced by the predicted value at timestep ts in the output sequence.

The input is first passed through a linear layer to generate embeddings. Embeddings are used to represent our load values in an n-dimensional space. These values are then passed to the encoder layer. Next, the encoder output is passed to the decoder layer as decoder input. The encoder and decoder layers are discussed in detail in subsection A. The decoder outputs a sequence of 25 timesteps with embeddings. These values pass through a linear layer which converts the embedded values into the load values. Next, the output is passed through a sigmoid activation layer which maps it between zero and one. Final output will be in the form $(z_1, z_2 \ldots, z_{ts-1}, z_{ts})$. These values are compared to $(i_1, i_2 \ldots, i_{ts-1}, i_{ts})$ where the first 24 timesteps are same as input but 25th timestep is the original value which was input as 0 initially. The objective of the transformer model is to replicate the first 24 values and predict the 25th value in this case. The final loss is only calculated for the 25th timestep. This model can very easily be modified to predict multiple timesteps ahead and also to handle multivariate data. The encoder and decoder are explained in detail in the next subsection.

3.1 Encoder and Decoder

Encoder: The encoder is composed of a stack of N identical layers. Each layer has two components, multihead self-attention and position-wise feedforward. Both of these are discussed in detail later. Layer normalization has been applied after each of these components. The output of each component is given by (1)

$$LN(x + f(x)), \tag{1}$$

where LN is layer normalization, $f(x)$ is the function implemented by the component and x is the input value passed to that function. The vector is normalized using (2).

$$LN(x) = \gamma \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} + \beta, \tag{2}$$

where $E[x]$ is the expected value of x, $Var[x]$ is the variance of x and γ and β are learnable affine transform variables.

Decoder: The decoder is also composed of a stack of N identical layers. Each layer in the decoder has three components, they are multihead self-attention, position-wise feedforward and multihead attention on encoder-decoder output. The primary difference between the decoder and the encoder is that the self-attention sublayer in the decoder is modified using look ahead masking to prevent a position from attending to subsequent positions. This is done to ensure that the predictions for the position can depend only on the known outputs at earlier positions. The decoder also has layer normalization after each component similar to the encoder.

3.2 Attention

3.2.1 Scaled Dot Product Attention

The Transformer model uses a unique attention mechanism called scaled dot product attention. Queries (Q) and key (K) vectors of dimension d_k and values (V) of dimension d_v are input into the function. The dot product of a query and all keys are computed and divided by $\sqrt{(}d_k)$. The output obtained from this is passed through a softmax layer to obtain weights for the corresponding value as seen in (3) where *Attn* is the attention function. Softmax (SM) of the vector is computed using (4).

$$Attn(Q,K,V) = SM\left(QK^T/\sqrt{d_k}\right)V, \tag{3}$$

$$SM(X_i) = \frac{e^{X_i}}{\sum_{j=1} e^{X_j}}, \tag{4}$$

where X_i is the input vector and X_j is the output vector.

3.2.2 Multihead Self-Attention

The modified transformer model applies linear transformations on the queries, keys and values h times (h is the number of heads) to obtain vectors of dimension d_k and d_v. Scaled dot product attention is then performed on the obtained transforms to get outputs of dimension d_v. The obtained vectors are concatenated and passed through another linear transform to obtain the final values as seen in (5), where Cc is concatenation and h_1, h_2, …, h_i are the heads which are computed using (6). The working of multihead self-attention can be seen in Figure 4.

$$Mh(Q,K,V) = Cc(h_1,\ldots,h_h)W^O, \tag{5}$$

$$h_i = Attn(QW_i^q, KW_i^k, VW_i^v), \tag{6}$$

where the projections are parameter matrices

$$W_i^Q \in R^{d_{model}d_k},$$
$$W_i^K \in R^{d_{model}d_k},$$
$$W_i^V \in R^{d_{model}d_v},$$
$$W^O \in R^{hd_v d_{model}}.$$

3.3 Feedforward

Position-wise feedforward layer (FFN) is a fully connected layer present in each of the encoder and decoder layers. It consists of 2 linear transformations with a Rectified

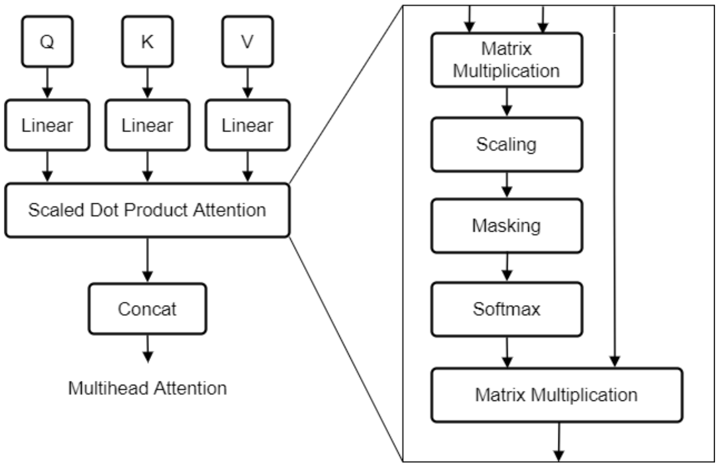


Figure 4. Mechanism of multihead self-attention layer

Linear Unit (ReLU) activation after the first layer as seen in (7).

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2. \tag{7}$$

The linear transformations are the same across all layers but the dimensionality differs based on the position. The input and output have a dimensionality d_{model} while the inner layers have dimensionality d_{ff}.

3.4 Sigmoid

Sigmoid (Sig) serves as the activation function in the output layer for the proposed model.

$$Sig(x) = \frac{1}{1 + e^{-x}}. \tag{8}$$

Sig function results in an S shaped output curve that exists between $[0, 1]$. Sig of an input vector x is computed using (8).

4 RESULTS

This section presents the results of the proposed model and compares it with the results of seven other state-of-the-art forecasting models across four different error metrics. This paper proposes using the Apache Spark big data framework along with a modified version of the transformer model for SG short term load forecasting. The dataset used for training and testing the models have been discussed in the following subsection.

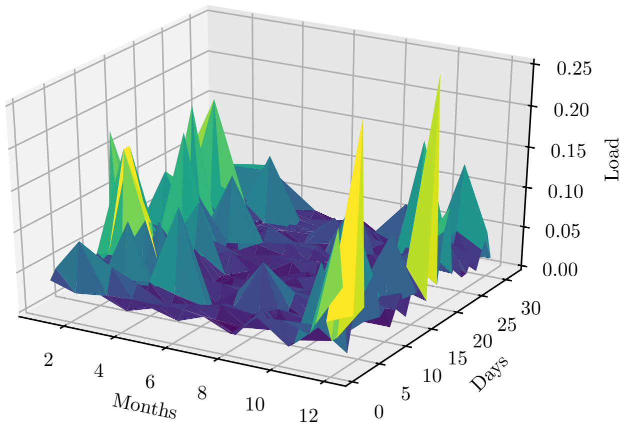


Figure 5. Monthly energy consumption for one house

4.1 Dataset Description and Analysis

The proposed model is trained and validated on a publicly available dataset containing energy consumption readings for a sample of 5 567 London households that took part in the Low Carbon London project between November 2011 and February 2014 led by UK Power Networks [30]. The dataset contains energy consumption readings, in kWh (per half hour), taken at half hourly intervals along with a unique household identifier, date-time and CACI Acorn group. CACI Acorn is a segmentation tool that categorizes the houses on the basis of the socio-economic conditions of the household. The dataset contains a total of 167 million rows of data. All the data was pre-processed and prepared for the models using the Apache Spark framework. Due to technical limitations, the entire dataset could not be used to train and validate the models. The dataset was divided into smaller subsets that were used for this purpose instead. A subset of 750 000 rows randomly selected from data of 100 houses belonging to the affluent acorn was used as the training dataset. For the validation dataset we took approximately 200 000 rows from the same set of houses. The final models were tested on a separate subset containing the data for one year from 10 houses which has 175 000 rows.

Table 1 shows a small sample of the training dataset. Figure 6 visualizes a small subset of the training dataset for eight different houses. Monthly change in energy consumption can be seen for one house in Figure 5. It can be concluded from this figure that peak usage occurs in the winter months. Figure 7 shows the

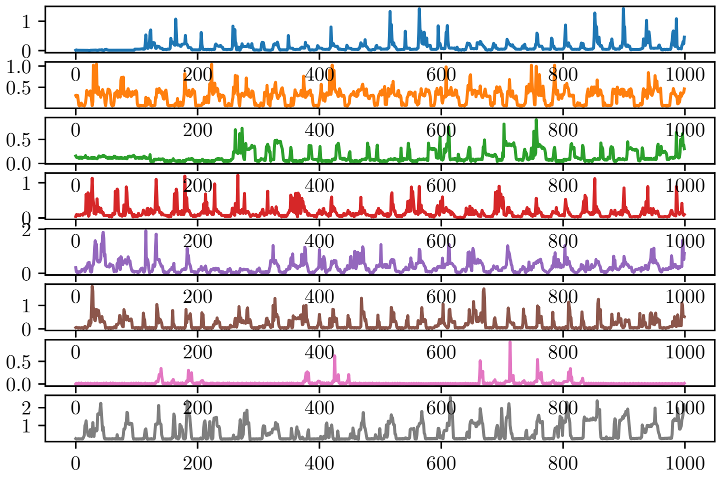


Figure 6. Energy consumption for eight houses over 1 000 timesteps

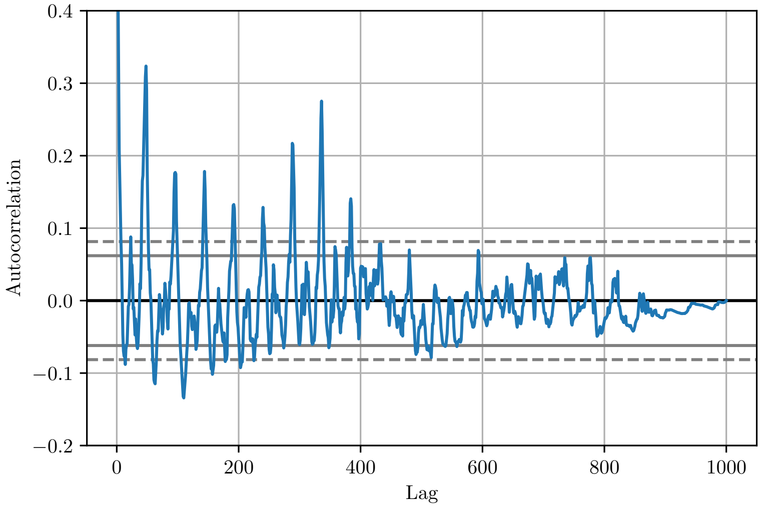


Figure 7. Self-Autocorrelation over 1 000 timesteps for one house

self-correlation of the sample taken in Figure 5, as seen in this figure the dataset has a fair amount of seasonality with alternating positive and negative peaks. Figure 8 is a heat map that shows the average hourly consumption for each day of the week. It can be seen from Figure 8 that peak usage occurs in the early hours of the day and usage is minimum around noon.

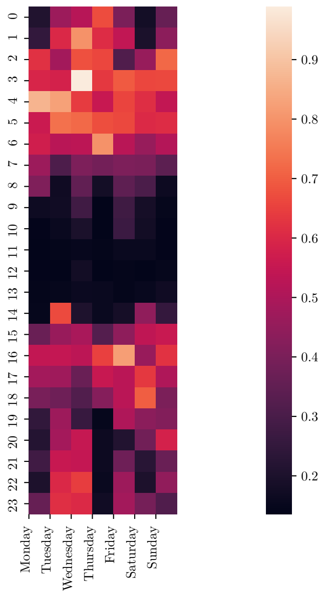


Figure 8. Heatmap of average hourly usage by the day

LCLid	Consumption (kWh/hh (per half hour))	Acorn
MAC000072	0.417	Affluent
MAC000072	0.311	Affluent
MAC000072	0.249	Affluent
MAC000072	0.295	Affluent
MAC000072	0.259	Affluent

Table 1. Training dataset sample

4.2 Experimental Results

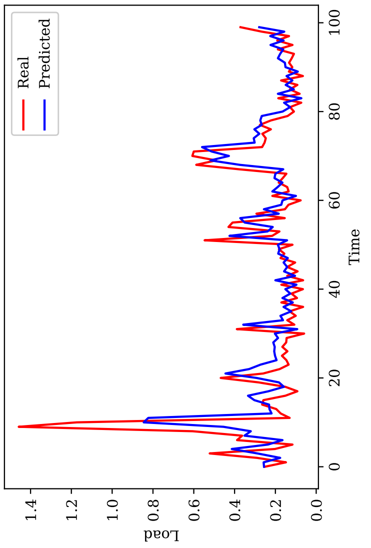
For this dataset, a 4-layer stacked encoder-decoder modified transformer model architecture has been used. The transformer model has been tested against 4 baseline deep learning models: LSTM, GRU, CNN and CNN-LSTM and 3 baseline machine learning models: LightGBM, CatBoost, Random Forest. All these models are trained to use 12 hours of data to predict half an hour of data. The LSTM and GRU are both 3 layer models. CNN is 2 convolutional layers followed by a dense layer and CNN-LSTM is a single time distributed convolutional layer followed by a dense layer along with an LSTM layer. All the models were trained for 20 epochs with a batch size of 64 on the training dataset. The 4-layer stacked modified transformer model took 5 hours to train for 20 epochs on the train dataset with 750 000 rows. All these computations were carried out on an Nvidia K80 with 2 496 CUDA cores operating at 4.1 TFLOPS with 12 GB of primary memory. The data pre-processing step took 2 hours using Apache Spark using a hyper-threaded Intel Xeon processor with 2 cores operating at 2.3 GHz with 12 GB of primary memory. As seen in Table 2, the transformer model has performed very well in comparison to all the other baseline deep learning models in terms of most error metrics.

Model	RMSE	MAE	MedAE	R^2
Modified Transformer	0.22004029	0.10875492	0.04158187	0.61770006
LSTM	0.22186998	0.11009181	0.04684585	0.61131435
GRU	0.22208220	0.10727535	0.04077151	0.61057043
CNN-LSTM	0.22541212	0.12090687	0.05422697	0.59880441
CNN	0.23921741	0.11336003	0.04301256	0.54815757
LightGBM	0.224019	0.106347	0.036593	0.6037490
CatBoost	0.225925	0.1071185	0.0395297	0.5969781
RandomForest	0.226586	0.1140612	0.0443616	0.5946145

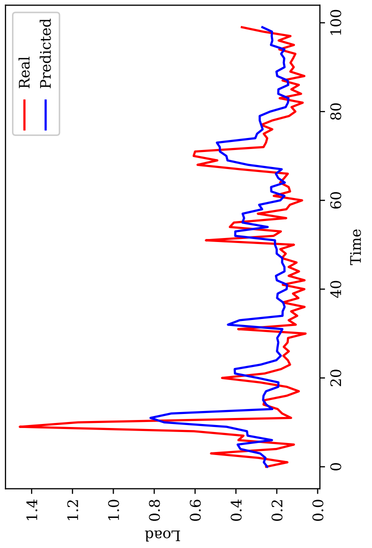
Table 2. Comparison of loss values obtained by all models on the test set

To evaluate the performance of the proposed method and the baseline models, RMSE, Mean Absolute Error (MAE), Median Absolute Error (MedAE) and R-squared score metrics have been used. RMSE is square root of the average of the square of the differences between the predicted and the actual values. MAE is the absolute value of the difference between the predicted value and the actual value. It tells us how big of an error we can expect from the forecast on average. MedAE works similarly to MAE and uses absolute value to calculate the loss. So, lower values of RMSE, MAE and MedAE mean that the proposed model performed well. R-squared score signifies the proportion of the variance in the dependent variable that is predictable from the independent variable. So a greater value of R-squared score means the model performed better. This paper was unable to use percentage error metrics like MAPE because the dataset used contains zero values and MAPE cannot be used with those due to division by zero error.

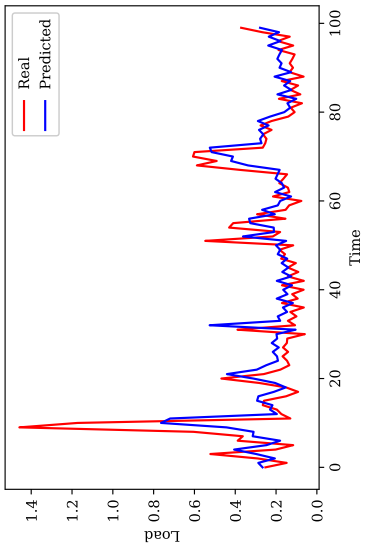
As shown in Table 2, the modified transformer model has the best results in terms of RMSE and R^2 error. In terms of MAE and MedAE, the modified trans-



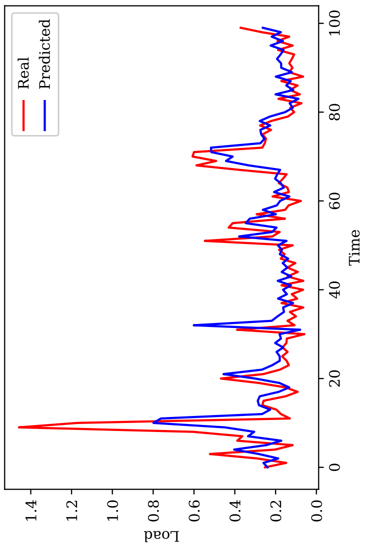
a) Modified Transformer



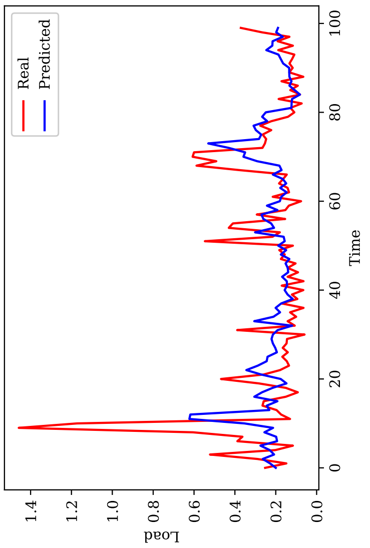
b) CNN-LSTM



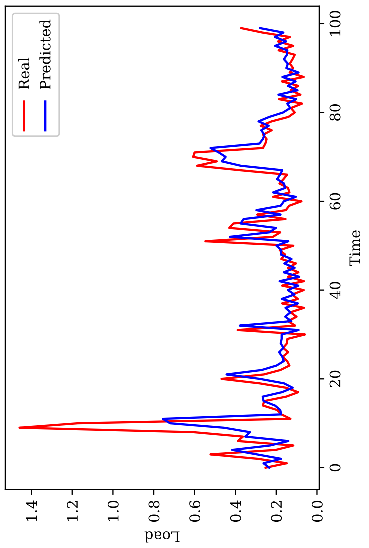
c) LSTM



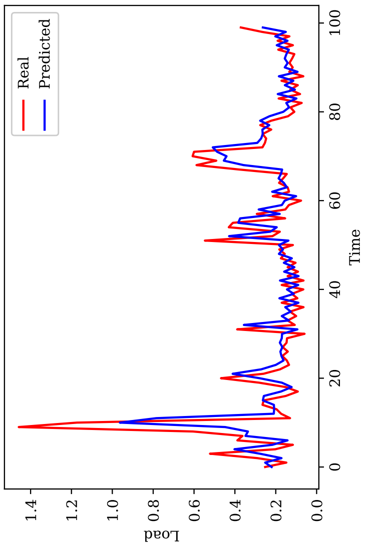
d) GRU



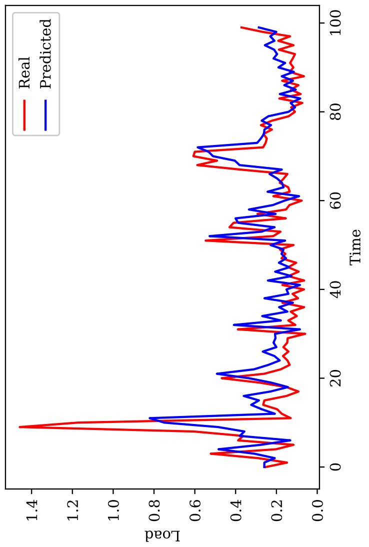
e) CNN



f) LightGBM



g) CatBoost



h) Random Forest

Figure 9. Load vs time graphs of real and predicted values of all the models

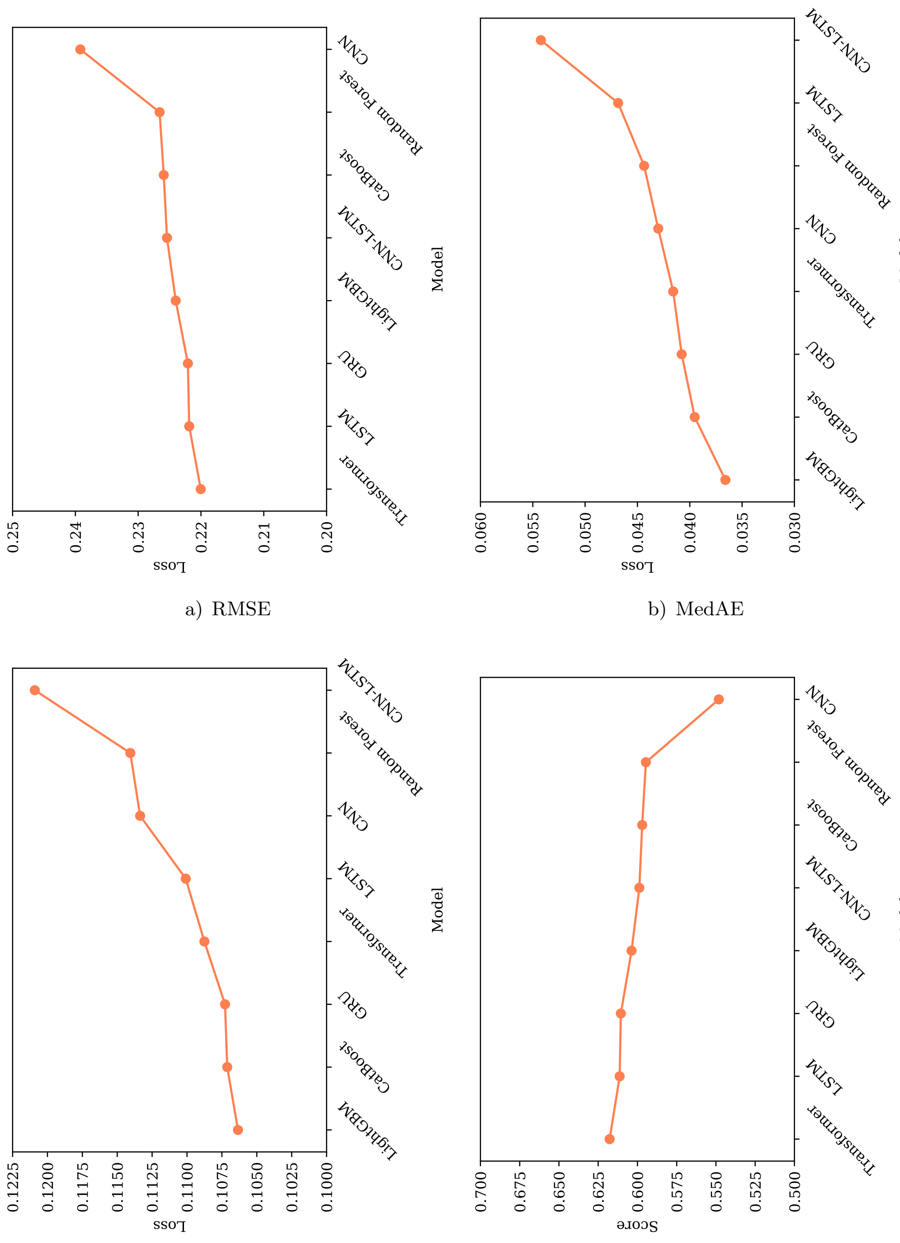


Figure 10. Evaluation metrics on different models

former model has the fourth best performance behind GRU, LightGBM and CatBoost. RMSE and R^2 penalize large errors more since they are quadratic measures as compared to MAE and MedAE which are linear measures. Therefore, on average the modified transformer model has performed better than all other deep learning models and machine learning models as it has the best performance in RMSE and R^2 and has performed very close to the models above it in MAE and MedAE, based on this we can infer that the transformer model makes slightly more errors but the total amplitude of errors is much smaller. Visual representation of the predicted values vs time and real values vs time given by all the models can be seen in Figure 9. It can be observed from this figure that the values predicted by modified transformer, LightGBM, CatBoost, LSTM and GRU are much closer to the real values than the values predicted by the other models. Also, the modified transformer model seems to have the best performance when it comes to predicting peaks. This is due to the modified transformer's abilities of predicting values by detecting correlations between the load values using attention mechanism.

5 CONCLUSION

SG is designed to tackle issues like variable electricity consumption and electrical outages to decrease the gap between energy supply and demand. In order to achieve this, efficient big data analysis and accurate load forecasting is necessary. This paper proposes using Apache Spark coupled with a modified version of the transformer model to predict energy consumption using short term load forecasting. Apache Spark is much faster compared to other big data frameworks such as Hadoop and has more features such as the Spark MLlib, which makes it ideal for SG load forecasting. Moreover, the modified transformer model was compared to other baseline machine learning models on a univariate time-series dataset and it performed better than all the baseline models in terms of most evaluation metrics. Thus, it can be concluded that the proposed scheme, using the modified transformer model and Apache Spark, is very accurate and efficient. Further work needs to be done to evaluate the usability of the proposed scheme on multivariate data using both weather and previous load information and further modifications to the transformer model need to be tested for time-series forecasting.

REFERENCES

[1] Mujeeb, S.—Javaid, N.—Javaid, S.—Rafique, A.—Ilahi Tamimy, M.: Big Data Analytics for Load Forecasting in Smart Grids: A Survey. International Conference on Cyber Security and Computer Science (ICONCS '18), 2018, pp. 193–202.

[2] Sagiroglu, S.—Terzi, R.—Canbay, Y.—Colak, I.: Big Data Issues in Smart Grid Systems. 2016 IEEE International Conference on Renewable Energy

Research and Applications (ICRERA), 2016, pp. 1007–1012, doi: 10.1109/ICRERA.2016.7884486.

[3] Kaisler, S.—Armour, F.—Espinosa, J. A.—Money, W.: Big Data: Issues and Challenges Moving Forward. 2013 46th Hawaii International Conference on System Sciences, 2013, pp. 995–1004, doi: 10.1109/HICSS.2013.645.

[4] Zhang, Y.—Huang, T.—Bompard, E. F.: Big Data Analytics in Smart Grids: A Review. Energy Informatics, Vol. 1, 2018, No. 1, Art. No. 8, doi: 10.1186/s42162-018-0007-5.

[5] Shi, H.—Xu, M.—Li, R.: Deep Learning for Household Load Forecasting – A Novel Pooling Deep RNN. IEEE Transactions on Smart Grid, Vol. 9, 2018, No. 5, pp. 5271–5280, doi: 10.1109/TSG.2017.2686012.

[6] Zhang, P.—Wu, X.—Wang, X.—Bi, S.: Short-Term Load Forecasting Based on Big Data Technologies. CSEE Journal of Power and Energy Systems, Vol. 1, 2015, No. 3, pp. 59–67, doi: 10.17775/CSEEJPES.2015.00036.

[7] Zhang, Z.—Wang, X.—Ji, Y.: The Power Load Forecasting of SVR Based on Hadoop. 2018 37th Chinese Control Conference (CCC), 2018, pp. 4484–4488, doi: 10.23919/ChiCC.2018.8482869.

[8] Yang, C.—Song, Y.—Qian, J.—Zhao, H.—Jiang, W.—Tang, H.—Wu, J.: Apache Spark Based Urban Load Data Analysis and Forecasting Technology Research. 2017 IEEE Conference on Energy Internet and Energy System Integration (EI2), 2017, pp. 1–6, doi: 10.1109/EI2.2017.8245396.

[9] Kumar, S.—Hussain, L.—Banarjee, S.—Reza, M.: Energy Load Forecasting Using Deep Learning Approach-LSTM and GRU in Spark Cluster. 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), 2018, pp. 1–4, doi: 10.1109/EAIT.2018.8470406.

[10] Chebbi, I.—Boulila, W.—Mellouli, N.—Lamolle, M.—Farah, I. R.: A Comparison of Big Remote Sensing Data Processing with Hadoop MapReduce and Spark. 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), 2018, pp. 1–4, doi: 10.1109/ATSIP.2018.8364497.

[11] Aziz, K.—Zaidouni, D.—Bellafkih, M.: Real-Time Data Analysis Using Spark and Hadoop. 2018 4th International Conference on Optimization and Applications (ICOA), 2018, pp. 1–6, doi: 10.1109/ICOA.2018.8370593.

[12] Amin, M. A. A.—Hoque, M. A.: Comparison of ARIMA and SVM for Short-Term Load Forecasting. 2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON), 2019, pp. 1–6, doi: 10.1109/IEMECONX.2019.8877077.

[13] Goswami, K.—Ganguly, A.—Kumar Sil, A.: Comparing Univariate and Multivariate Methods for Short Term Load Forecasting. 2018 International Conference on Computing, Power and Communication Technologies (GUCON), 2018, pp. 972–976, doi: 10.1109/GUCON.2018.8675059.

[14] Wang, J.—Li, L.—Niu, D.—Tan, Z.: An Annual Load Forecasting Model Based on Support Vector Regression with Differential Evolution Algorithm. Applied Energy, Vol. 94, 2012, pp. 65–70, doi: 10.1016/j.apenergy.2012.01.010.

[15] Nie, H.—Liu, G.—Liu, X.—Wang, Y.: Hybrid of ARIMA and SVMs for

Short-Term Load Forecasting. Energy Procedia, Vol. 16, 2012, pp. 1455–1460, doi: 10.1016/j.egypro.2012.01.229.

[16] Kong, W.—Dong, Z. Y.—Jia, Y.—Hill, D. J.—Xu, Y.—Zhang, Y.: Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. IEEE Transactions on Smart Grid, Vol. 10, 2019, No. 1, pp. 841–851, doi: 10.1109/TSG.2017.2753802.

[17] Jahangir, H.—Tayarani, H.—Gougheri, S. S.—Golkar, M. A.—Ahmadian, A.—Elkamel, A.: Deep Learning-Based Forecasting Approach in Smart Grids with Micro-Clustering and Bi-Directional LSTM Network. IEEE Transactions on Industrial Electronics, Vol. 68, 2020, No. 9, pp. 8298–8309, doi: 10.1109/TIE.2020.3009604.

[18] Zhou, H.—Zhang, Y.—Yang, L.—Liu, Q.—Yan, K.—Du, Y.: Short-Term Photovoltaic Power Forecasting Based on Long Short Term Memory Neural Network and Attention Mechanism. IEEE Access, Vol. 7, 2019, pp. 78063–78074, doi: 10.1109/ACCESS.2019.2923006.

[19] Lv, P.—Liu, S.—Yu, W.—Zheng, S.—Lv, J.: EGA-STLF: A Hybrid Short-Term Load Forecasting Model. IEEE Access, Vol. 8, 2020, pp. 31742–31752, doi: 10.1109/ACCESS.2020.2973350.

[20] Sehovac, L.—Grolinger, K.: Deep Learning for Load Forecasting: Sequence to Sequence Recurrent Neural Networks with Attention. IEEE Access, Vol. 8, 2020, pp. 36411–36426, doi: 10.1109/ACCESS.2020.2975738.

[21] Zhao, Z.—Xia, C.—Chi, L.—Chang, X.—Li, W.—Yang, T.—Zomaya, A. Y.: Short-Term Load Forecasting Based on the Transformer Model. Information, Vol. 12, 2021, No. 12, Art. No. 516, doi: 10.3390/info12120516.

[22] Wang, Y.—Gan, D.—Sun, M.—Zhang, N.—Lu, Z.—Kang, C.: Probabilistic Individual Load Forecasting Using Pinball Loss Guided LSTM. Applied Energy, Vol. 235, 2019, pp. 10–20, doi: 10.1016/j.apenergy.2018.10.078.

[23] He, W.: Load Forecasting via Deep Neural Networks. Procedia Computer Science, Vol. 122, 2017, pp. 308–314, doi: 10.1016/j.procs.2017.11.374.

[24] Khan, A. R.—Mahmood, A.—Safdar, A.—Khan, Z. A.—Khan, N. A.: Load Forecasting, Dynamic Pricing and DSM in Smart Grid: A Review. Renewable and Sustainable Energy Reviews, Vol. 54, 2016, pp. 1311–1322, doi: 10.1016/j.rser.2015.10.117.

[25] Hafeez, G.—Alimgeer, K. S.—Khan, I.: Electric Load Forecasting Based on Deep Learning and Optimized by Heuristic Algorithm in Smart Grid. Applied Energy, Vol. 269, 2020, Art. No. 114915, doi: 10.1016/j.apenergy.2020.114915.

[26] Son, M.—Moon, J.—Jung, S.—Hwang, E.: A Short-Term Load Forecasting Scheme Based on Auto-Encoder and Random Forest. In: Ntalianis, K., Vachtsevanos, G., Borne, P., Croitoru, A. (Eds.): Applied Physics, System Science and Computers III (APSAC 2018). Springer, Cham, Lecture Notes in Electrical Engineering, Vol. 574, 2019, pp. 138–144, doi: 10.1007/978-3-030-21507-1_21.

[27] Li, H. Z.—Guo, S.—Li, C. J.—Sun, J. Q.: A Hybrid Annual Power Load Forecasting Model Based on Generalized Regression Neural Network with Fruit Fly Optimization Algorithm. Knowledge-Based Systems, Vol. 37, 2013, pp. 378–387, doi:

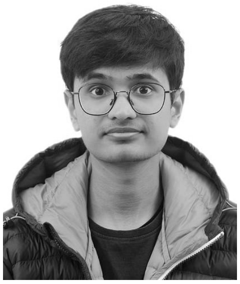
10.1016/j.knosys.2012.08.015.

[28] Vaswani, A.—Shazeer, N.—Parmar, N.—Uszkoreit, J.—Jones, L.—Gomez, A. N.—Kaiser, L.—Polosukhin, I.: Attention Is All You Need. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017). Curran Associates, Inc., 2017, pp. 5998–6008, doi: 10.48550/arXiv.1706.03762.

[29] Wu, N.—Green, B.—Ben, X.—O'Banion, S.: Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. 2020, doi: 10.48550/arXiv.2001.08317.

[30] Smartmeter Energy Consumption Data in London Households. 2015, `https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households`.

Aditya Upadhyay is currently a final year computer engineering student at the Thapar University. His main research interests include AI, deep learning, machine learning, reinforcement learning and natural language processing. He is also currently interning at Wysa.



Dhruv Garg is currently a final year computer engineering student at the Thapar University. His main research interests include AI, deep learning, machine learning, reinforcement learning and natural language processing. His goal is to significantly improve people's lives through artificial intelligence. He is also currently interning at the Aganitha Cognitive Solutions.

Mukesh Singh is Full Professor at the Thapar Institute of Engineering and Technology. He obtained his Ph.D. from IIT Guwahati in 2013, and has since then made significant contributions to the field of electric vehicle, vehicle to grid, smart grid, and renewable energy. He has been bestowed with the Engineer Gurcharan Singh Award by the Punjab Science Academy. Currently, he is leading a project worth 20 Crores on electric vehicles as Project Leader under DST, CSIR, DHI, and Industry. He has published numerous research papers in renowned journals and has served as a resource person for several conferences and workshops both in India and abroad. His research interests lie in the areas of renewable energy sources, smart grids, and vehicle-to-grid technologies. He currently supervises 10 Ph.D. students and has successfully guided 5 Ph.D. students in defending their Ph.D. Thesis. Recently, he has devoted his attention to the development of a machine learning algorithm for electric vehicle components, with his major breakthrough being the creation of a netmeter for vehicle to grid applications.