Short term load forecasting using LSTM ensembled network on utility scale load demand.

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

13-10-2021 / 29-12-2021

CITATION

Fattah, Fathun Karim; Mojumder, Pritom; Fuad, Azmol Ahmed; Ahmad, Mohiuddin; hossain, Eklas (2021): Short term load forecasting using LSTM ensembled network on utility scale load demand.. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.16807213.v2>

DOI

[10.36227/techrxiv.16807213.v2](https://doi.org/10.36227/techrxiv.16807213.v2)

Short term load forecasting using LSTM ensembled network on utility scale load demand

Fathun Karim¹, *Member, IEEE*, Pritom Mojumder², Azmol Ahmed Fuad³, Mohiuddin Ahmad⁴, *Member, IEEE*, Eklas Hossain⁵, *Senior Member, IEEE*

Abstract—This paper aims to assess the potential of Long Short Term Memory (LSTM) ensembled networks for short-term load forecasts (STLF) over other conventional neural networks on utility-scale load demand. The study shows that LSTM ensembled networks perform substantially better than other conventional neural networks. For this study, the LSTM network has been used as the primary benchmark against which our ensemble strategy was thoroughly assessed since our proposed network is merely an extension of LSTM networks. An improvement of 0.04% for hour-ahead forecasts and 0.14% for the day ahead forecasts had been observed for the LSTM ensembled approach. However, the error increased 2.377% for the ensembled approach and 2.47% for the LSTM approach for day-ahead forecasts. Although LSTM networks have been found out to be more efficient than other conventional neural networks, LSTM ensembled techniques can improve the efficiency even further. The significant error reduction capability of the LSTM ensembled network compared to the LSTM network carries the potential to save utilities 224000 dollars for the day ahead forecasts and 64000 dollars for an hour ahead forecast. Moreover, it can play an even more efficient role compared to the other conventional neural network strategies that are widely established.

Index Terms—STLF, Neural Network, LSTM network, Ensemble learning, Bagging strategy, Hourly load forecast, Day ahead load forecast

I. INTRODUCTION

LOAD Forecasting is an essential component for utility companies to manage the production and distribution of electric power in the most convenient and cost-efficient way possible. A better forecast model can save the utilities a lot of money and also help the consumers get an interruption-free low-cost energy supply. Short term load forecasting (STLF) is essential for maintaining appropriate coordination between the basic generation scheduling functions, providing insights on the security condition of the whole system at any particular point, and also for presenting the system dispatcher with appropriate and timely information with regards to the load behavior [1]. The fact that load pattern between short intervals appears to be pretty erratic is what makes a highly accurate forecast quite difficult to achieve. But the benefits of a precise forecast is worth all the effort, considering the economic and security advantages, which are one of the driving motivations behind the large volume of researches being quite regularly produced in this field.

Short-term load forecasting has been approached through several techniques. Statistical models had largely been used for this purpose, for example, Multiple Linear Regression, Stochastic Time Series, General Exponential Smoothing, State Space Method, etc. [2] with some handful of exceptions,

for example, knowledge-based algorithmic approach [3], and ALFA approach [4]. Among other statistical models most popularly seen in the literature are Autoregressive Moving Average (ARMA) [5], Autoregressive Integrated Moving Average (ARIMA) [6], General Exponential Smoothing [7], Holt-Winters smoothing approach [8], Periodic AR modelling [8], Kalman Filtering [9], Box Jenkins Method [10], Autoregressive Moving Average with Exogenous Inputs (ARMAX) [11]. Despite such a wide pool of statistical alternatives, linear regressions methods were seen to be mostly prevalent because due to their relative simplicity, fewer trade-offs along with a reasonable performance [12], [13]. However, a problem associated with linear and time-series models is that their performance is found to be relatively poor when capturing the nonlinear nature of the load due to a variety of exogenous factors.

To reconcile the nonlinear nature of the load, researchers brought forward a multitude of forecasting methods other than statistical models. Fuzzy logic systems and their different variants were experimented with since they could cover up the drawbacks of rigid mathematical models bound within a stringent rule, allowing them to accommodate erratic fluctuations, unforeseen loads, mechanical uncertainties, transmission losses, voltage instability, etc. Fuzzy approaches had been found to perform with a relatively lower, single-digit error compared to other contemporary approaches [14], [15], but the error fluctuations had been observed to behave quite irregularly when mapped for a whole day [15]. With the advent of machine learning technologies, a new domain had appeared for carrying out more reliable forecasts. Such techniques include support vector machine [16], [17], extreme learning machine [18], support vector regression [19], random forest models [20], extreme gradient boosting (XGBoost) [21]. While support vector models were seen to perform relatively better as compared to other conventional models [22], one drawback associated with such technique had been that it was unable to perform with similar efficiency on large scale dataset because SVM complexity is directly dependent upon the size of the dataset. E. Aguilar et al. have shown the superiority of XGBoost over SVR and RF methods in carrying out more precise forecasts [21]. However, XGBoost methods should be utilized for dataset with abundant features and less noise. Otherwise, it may cause the model to underperform. Random Forest models are computationally time-consuming and do not outperform XGBoost models concerning the accuracy of the forecast, as shown by E. Aguilar et al. [21]. Since the beginning of their inception, Neural Networks have always lured researchers

into accommodating their potential into the field of load forecasting. It has better learning potential, and with time many novel variants emerging with time with even better learning capacity, the prospects become even more promising. Hippert et al. have reviewed a wide set of neural network models that had been implemented for short-term load forecasting purposes, and they identified some common recurring issues in such models, such as overfitting and overparameterizing [23]. ANN was first seen to be in use in the early days of machine learning [24]. Since then, with further developments in the field of neural networks, many different variants and approaches were experimented with, such as radial basis function neural networks [25], wavelet neural network approaches [26], neural networks with genetic algorithms executed with backpropagation [27], neural networks with particle swarm optimization [28], evolutionary ELM [29] etc. Many hybrid approaches were explored as well to augment the efficiency of the machine learning and neural network models. Mamun et al. have done an extensive review of all available literature on the hybrid ML approach towards STLF [30]. Their work exhibited the superiority of hybrid SVM and ANN models for better accuracy compared to other hybrid models. From the ANN hybrids (ANN-FA, ANN-FOA, ANN-GA, ANN-CT, ANN-NFIS, ANN-WT, ANN-AIS, ANN-PSO), they had found out ANN-NFIS hybrid can provide the highest accuracy, but it suffered from gradient descent problem, longer runtime, and lacked real-time response. From SVM hybrids (SVM-FA, SVM-GA, SVM-FOA, SVM-PSO, SVM-ABC, SVM-SA). They had also observed that SVM-ABC could provide better forecasts results with the least error, but the problem associated with this technique was its slow convergence speed. Mayur et al. have introduced a different SVM hybrid (SVM-GOA) which was not included in the assessment of Mamun et al. [31]. It has shown its relative superiority over other hybrid methods, for example, SVM-GA and SVM-PSO.

Following the breakthrough research on deep learning by Hinton and Lecun [32], researchers dived into the world of deep learning neural networks to tinker with these novel techniques and check their viability for STLF purposes. From the many methods of Deep learning networks that are available, one popular technique was to apply convolutional neural networks [33]. However, CNN networks were mostly seen to be used as a component for hybrid models such as CNN-LSTM networks [34], [35]. Bendaoud et al. has observed CNN models to perform better than other ML trends such as ANN, SVM, RF, GBRT (Gradient Boosting Regression Trees) for short-term forecasts. Other neural networks that had been explored include Feed Forward deep neural network [36], Deep neural network with Restricted Boltzman Machines (RBM) pretraining [37], [38], RBM with Elman Neural Network [39]. Feed Forward networks and RBM pretraining methods were found to have augmented the efficiency of the deep neural networks, but these models become harder to train with an increasing number of layers. Therefore, the restriction of layer numbers ends up hobbling the models' efficiency. Deep recurrent neural networks were found to be more effective than feedforward networks [36] and hence they have been widely studied throughout the literature [40]. RNN was found

to be useful for learning highly nonlinear relationships and also to learn shared uncertainties. However, it had an issue of overfitting and layer constraints, which were later tried to be addressed by a novel pooling method using PDRNN, as proposed by Heng Shi et. Al. [41]. The shortcomings of deep learning models were further addressed by incorporating residual neural networks instead of stacking multiple hidden layers between the input and output [42]. RNN techniques also saw decent improvement when coupled with input attention mechanisms and hidden connection mechanisms [43].

Using RNN for forecasting had a different concern: the vanishing gradient problem. A little tweak into the RNN architecture was introduced to address this problem by adding a memory unit called LSTM. Incorporating LSTM units has been shown to significantly increase the gap length for RNN backpropagation without fear of the long-term gradients vanishing away since LSTM units allow long-term gradients to propagate unchanged. LSTM networks have not been left unexplored in the literature [44]. W. kong has shown LSTM networks to perform better than other neural network models [44]. Like other standard statistical and machine learning models, LSTM models have also been coupled with various models and experimented with for STLF purposes. Some of those hybrid models include CNN-LSTM [34], [35], MLR-LSTM [45], EMD-LSTM [46], and Resnet-LSTM [47], all of which have proven to significantly improve the error factor in the forecasts of standalone LSTM models.

It is fairly easy to achieve a MAPE value within a range of 10% with any neural network approach, but on the other hand, dropping the error further below is found to be equally challenging [23]. Being able to drop the prediction error even as little as 1% has a significant impact on the economic aspect of the whole power system. It has been found out that for a 10,000 MW utility, a 1% drop in forecasting error can save the company up to 1.6 million dollars annually [48]. This paper proposes an LSTM ensembled network with bagging architecture and assesses its relative performance against a regular LSTM model. We have picked LSTM as our primary benchmark since LSTM networks had been previously tested, and their superiority over other neural networks has been widely established. Some ensembled LSTM architectures can be found throughout literature which propose solutions for residential or industrial load levels [49], [50]. Load patterns for industrial and residential levels vary significantly compared to utility level load demand, requiring distinct feature engineering approaches to incorporate the trends of the load datasets. Our proposed network follows a different architecture and is designed to solve utility-scale load concerns since it has not been addressed so far. This paper aims to fill up that gap in the literature. We had tried to test if our proposed LSTM ensembled network can improve forecast precision compared to a single LSTM network and, if so, to quantify that effect. Although LSTM is a sufficient benchmark itself, we had nonetheless tested our proposed network further with other conventional neural network models to make it further convincing. However, our primary benchmark had been regular LSTM model since LSTM ensembled architecture are merely an extension of basic LSTM architecture. The rest of the paper

has been organized as follows, Section II discusses some of the fundamentals of the networks, section III discusses the methodology, section IV discusses the results and discussion, and section V concludes the paper.

II. LSTM & LSTM NETWORK ENSEMBLE TECHNIQUE

A. LSTM

Long Short-Term Memory is a variant of recurrent neural network proposed by Hochreiter & Schmidhuber [51]. It tackles the complications arising from the long-term dependencies of RNN.

The chain structure of LSTM contains four neural networks and different cells, which are basically memory blocks. The networks used have a single input layer, a single hidden layer, and one output layer. The network uses cells for the storage of information, while the gates are used to perform memory manipulations.

As stated by Gers et al. [52], the memory block can be referred to be the basic unit of the hidden layer of an LSTM network. At the core, these memory cells have a recurrently self-connected linear unit called ‘‘Constant Error Carousel’’ or CEC. The activation of these CEC is called cell state. When activation is around zero, the irrelevant noise and inputs are blocked from entering the cell, keeping the cell state pristine.

As each of these memory cells is self-looping, they contain temporal info in their cell states. The information flow through the network is done by manipulating the cell state (i.e., writing, erasing, and reading).

Three gates are used to achieve cell manipulation, respectively. The entire operation of the LSTM can be elucidated using a few mathematical equations.

i) Forget Gate: Forget gate is utilized for the removal of the information that is no longer useful in the cell state. This gate is described by the following equation.

$$forget_{gate} = \sigma(in_{[t]}W_{forgetint} + out_{[t-1]}W_{forgetcell} + b_{forget}) \quad (1)$$

Where $forget_{gate}$, $in_{[t]}$, $out_{[t-1]}$ denotes the forget gate activation, input state and previous output state vector respectively. $W_{forgetint}$ and $W_{forgetcell}$ are the weight matrices for forget gate to intermediate state and forget gate to cell vector respectively. The term b denotes individual bias vector for each of the gates. σ denotes the logistic sigmoid function.

ii) Input Gate: Input gate is applied for the addition of necessary information to the cell. The input gates operation can be denoted using the following equation:

$$input_{gate} = \sigma(in_{[t]}W_{inputint} + out_{[t-1]}W_{inputcell} + b_{in}) \quad (2)$$

Where σ , $in_{[t]}$, $out_{[t-1]}$ and b bear their previously stated meanings. The $input_{gate}$ denotes the activation state vector of the input gate. $W_{inputint}$ and $W_{inputcell}$ are the weight matrices for input gate to intermediate state and input gate to cell vector individually.

iii) Output Gate: The task of the output gate is to extract useful information from the current cell state and then to

present this information in the form of the output. The output gate equation is denoted by the following equation,

$$output_{gate} = \sigma(in_{[t]}W_{outputint} + out_{[t-1]}W_{outputcell} + b_{output}) \quad (3)$$

Where the term $output_{gate}$ denotes the output gate activation vector. $W_{outputint}$ and $W_{outputcell}$ are the weight matrices for output gate to intermediate state and output gate to cell vector chronologically. Apart from the equations describing the working of the three gates the LSTM network memory cell has 3 more equations which are given below:

$$up = \tanh(in_{[t]}W_{upint} + out_{[t-1]}W_{upcell} + b_{up}) \quad (4)$$

$$int_{[t]} = int_{[t-1]}forget_{gate} + input_{gate}up \quad (5)$$

$$out_{[t]} = output_{gate}\tanh(up) \quad (6)$$

Where equations 4 through 6 represent the update signal, value of the state at time step t , and output of the cell, respectively. W_{upint} and W_{upcell} are the weight matrices for updating to intermediate state and for updating to cell vectors.

B. LSTM Network Ensemble Technique

The ensembling strategy for artificial neural networks is fabricated by training multiple models and combining their attributes to produce a better output. Due to its very nature of incorporating multiple outputs, it is more natural for an ensemble strategy to perform better than any individual model because the shortcomings of different models get averaged out while simultaneously exploiting their relative advantages.

In an ensemble learning method, time forecasting is improved by combining multiple forecasts (predictions) generated through several distinct machine learning networks. Such models utilize their relative advantages and henceforth minimize the overall prediction error. One of the benefits of the ensembled approach is that the parameters do not need to be heavily optimized according to one network, relieving the system from being overparameterized. Dietterich has categorized the benefits of using the ensembling technique into three types, statistical, computational, and representational [53]. In our particular study, the volume of the dataset is quite huge. Our goal is to achieve statistical advantages due to its ability to smooth out data and feature shortcomings. To bypass computational complexities with regards to time and resource and also to bypass the diminishing returns of performance, the number of ensemble networks are kept at an optimal level, usually around 3 or 5 and in some cases around 10 depending on the context and the neural networks which are to be ensembled. LSTM ensembled approach uses a diverse number of LSTM networks that are distinct in terms of the attributes they incorporate, such as the kernel initializers, optimizers, activation functions, number of layers, etc. An ensemble approach is deemed eligible if the different networks employed in the ensemble model can produce distinct outputs.

Fig. 1: Proposed LSTM ensemble network with different activation functions and bagging generalization

III. METHODOLOGY

Prior to designing the LSTM and LSTM ensemble networks, a public dataset had been collected from ISO North England. This dataset included the hourly load information of the North American region starting from 1/3/2003 to 31/12/2014. In addition, this dataset also includes the corresponding temperature value for each hour which we would further use as a feature in our network. After availing the dataset, it was processed and engineered in order to make the data eligible for our proposed and test networks. The networks were built in a python 3.7 environment and executed on Google Colaboratory cloud platform using Keras library and TensorFlow backend. GPU allocated for the tasks were NVIDIA TESLA k80 11GB, which comes as a freemium feature from google colab.

Five parameters were passed on the LSTM Module from the given dataset. Month, Day, Weekday, Hourly Demand, and Temperature were taken as input parameters. Two more features, i.e., the season and festival data, were added alongside the input parameters. Unlike the rest of the parameters, these parameters have been calculated with the given information from the dataset. Weekday, Festival, Season data had been processed further using one-hot encoding to make them more expressive and useful for the networks since they are categorical data. Both the proposed and other benchmark networks were individually designed to produce hour-ahead and day-ahead forecasts. Although a different approach to feature engineering were required for the day-ahead forecast, the neural networks for both cases remained the same. The whole work can be summarized using the following block diagram in Figure 2.

A. Hourly Forecasting

1) *Data Processing*: For hourly load forecasting, a set of features were accommodated into the sample. One of the features was the inclusion of load data from the previous six hours. The intuition behind this approach was to help

Fig. 2: Generalized Block Diagram of the Work

the model capture the load trend even more efficiently. Since load patterns depend on a day's timeframe, a distinct pattern can be observed for a 24-hour timespan, which gets repeated across the entire time series following a similar pattern but having distinct contours. Therefore, we took an approach to incorporate the insights of the day to make our prediction more efficient. Keeping this in mind, we had fed our neural network models with the load trend of the past six hours to help the model capture the course of the load curve more specifically. Passing such data would help the model tune its weights in accordance with the particular timeframe of the forecast point. Passing in the region of timeframe thereby adds to efficiency of the models.

The dataset was randomly split into three subsets before being fed into the prediction networks. The split ratio was maintained as 64% for training data, 16% for validation data, and 20% for test data. The split was randomized in order to avoid bias. The batch size was kept to be 32 and the total number of batches passed on the network per iteration was 2076, making the total number of training data 66432.

The dataset features were aimed at helping the model assess the time series pattern more precisely since all of these factors significantly influence the nature of load consumption alongside other meteorological factors, such as temperature. Therefore, temperature data was provided as input without

Features	Details	Size
M	Month	12
W	Weekday	1
H	Hour	1
S	Season	4
T	Temperature	1
F	Festival	1
prev_hour_x	Previous x^{th} hour demand	6

TABLE I: Details of the features passed onto the network

further manipulation or normalization. Table I shows the naming of the features their deatails and their size.

2) *LSTM Model Architecture*: The dataset was then fed into the LSTM network with samples containing 26 features each. The previous load demand parameter had been set to null for initial samples. The LSTM network consisted of 6 layers. The first 2 layers comprised regular LSTM layers, while the latter four were picked to be dense layers. The activation function chosen for the output layer was SELU (Scaled Exponential Linear Unit), for its unique normalizing and faster convergence property and also because it frees the network from vanishing and exploding gradients concerns. Table II shows the different stages of the network, their outputs, and some parameters associated with each of these stages.

Layer type	Output shape	Parameter
input	[1 , 26]	0
lstm	[1, 20]	3760
lstm	[20]	3280
dense	[20]	420
dense	[20]	420
dense	[20]	420
dense	[20]	420
dense	[1]	21

TABLE II: LSTM model architecture details in tabulated form

A total of 300 epochs were chosen for training the dataset. Beyond 300 epochs, the error was not improving any further, which indicated that extending epochs any further would have caused the network to suffer from redundancy and runtime inefficiency. The learning rate was fixed at $1e-03$.

3) *LSTM Ensemble Model Architecture*: After the LSTM execution, the dataset was run through our designed LSTM ensembled network, and its efficiency was observed in terms of forecast precision. The idea behind the ensembling is that different models will learn the input features differently and there combined advantage would therefore help to achieve a better generalization. For an ensemble approach, we have used four different LSTM variants with four different activation functions and concatenated them to produce an output. We have used four exponential linear units, ELU, SELU, GELU, and a softplus activation function. Chen k. opined that RELU

would have been a preferable activation function to utilize, but since it comes with the drawback of dead RELU problem, they have resorted to a modified RELU function-PRELU. However, we have used ELU as an alternative since it addresses the dead RELU problem and provides a better noise-robust deactivation state than modified RELUs like PRELU or LRELU. SELU was also picked for a similar reason: to address dead RELU issues while also providing a self-normalizing feature. SELUs have been observed to perform faster than most other activation functions. Vanishing and exploding gradient problems are virtually nonexistent for this cases. GELU was used because it was a fairly new model being used in transformer models like GPT-3 and others, which has been observed to perform relatively better in terms of nonlinearity to other activation functions, especially ELU and other RELU variants. GELU also is capable of being differential to all values for the input. Moreover, the softplus function was used to smoothen the approximations even further. For ELU and SELU variants, we have used glorot_uniform as kernel initializer, and for the latter two, GELU and Softplus variants, random_normal had been used. The four different LSTM variants worked in tandem producing individual outputs, which were then passed into a bagging process, yielding our final output. Bagging or bootstrap aggregating approach was employed instead of other ensemble algorithms because it can work with minimal complexity without having to compromise its prediction performance [54]. The bagging process takes in the initial forecast outputs and aggregates them to achieve a final prediction. Through this process, the error differences get averaged out with maximum advantage having been garnered. After the bagging process, the aggregated output was channeled through a dense output layer to receive the final forecast. The general idea of our proposed network can be visualized through figure 1. The learning rate was kept $1e-03$ for the ensembled method as well.

B. Day Ahead Forecast

1) *Data Processing*: We further extended our work to check how efficiently our network works in the case of day-ahead load forecasts. Since short-term load forecasts incorporate prediction into minutes ahead to hours ahead and even some days ahead, a fair assessment of day-ahead forecasts alongside the hour-ahead forecasts was reckoned to further establish the model's resiliency. The previous prediction networks for the hour ahead forecast were tweaked to generate the day ahead forecast instead of the hour ahead forecast. We had only one output for the hour-ahead forecast model, but we should be expecting 24 forecast outputs through this approach. Since our output size had increased, we had to increase the number of features in our input sample to suit the model. For day-ahead forecasts, we had dropped the previous 6 hours demand from the input sample, which we previously included for an hour ahead forecasts to capture hourly trends, and instead added the previous seven days demand set into the sample to capture daily trends. We have also included the previous month's load data into the sample to provide the model with even more trend cues. A null value was passed for initial samples

with no previous seven-day demand data. Season, festival, and weekend/weekday data were derived similarly to the hour ahead forecast model. Month and season data had been hot encoded. The total size feature size in the sample ended up being 258 after data processing. The dataset was splitted just as before it was done for the hour ahead approach, which is 64% for train data, 16% for validation data and 20% for test data. Similarly to the hour ahead approach, the batch size here was chosen to be 32 as well. Through each iteration 87 batches were being passed in the network, making the total number of training data 2784. Table III shows the feature details and their sizes for the day ahead portion of the work. Table IV tabulates the system details highlighting all of its stages.

Features	Details	Size
M	Month	12
W	Weekday	1
S	Season	4
F	Festival	1
dh_x	Demand of h^{th} hour of the day	24
th_x	Temperature of the h^{th} hour of the day	24
prev_Dn_dh_x	Demand of the h^{th} hour of the previous n^{th} day	168
prevM_dh_x	demand of the h^{th} hour of the previous month	24

TABLE III: Features and their corresponding details for day ahead forecasting

2) *LSTM model*: The dataset was then passed into the LSTM model without any further processing. The LSTM network contained six hidden layers. The first two layers were kept as regular LSTM layers, and the subsequent layers were kept as Dense Layers. The output was a dense layer having an activation function SELU. A total of 1200 epoch with a learning rate of $1e-03$ was adopted to train the model.

Layer type	Output shape	Parameter
input	[1, 258]	0
lstm	[1, 250]	367200
lstm	[200]	320800
dense	[200]	40200
dense	[200]	40200
dense	[200]	40200
dense	[200]	40200
dropout	[200]	0
dense	[24]	4824

TABLE IV: Table showing the different layers, associated outputs, and parameters in a day-ahead forecast using LSTM.

3) *LSTM Ensembled Model*: For the ensembled approach, we proceeded similarly as before, taking four different LSTM

Fig. 3: Training Loss Comparison for LSTM and LSTM Ensembled Network

variants and concatenating them to produce an output. The LSTM variants were created with four different activation functions, ELU, SELU, GELU, and Softmax, just as it had been done for an hour ahead forecast. The kernel initializers were adopted similarly too; glorot_uniform initializer for ELU and SELU, and random_normal for GELU and Softmax. Similar to the LSTM approach, 1200 epochs with a learning rate of $1e-03$ were chosen for the ensembled approach in order to avoid epoch redundancy. The batch parameters and the number of training data were kept the same as the LSTM approach previously defined.

IV. RESULTS AND DISCUSSIONS

A. Hour ahead forecast

The LSTM model took 98.43 minutes to be trained with the availed dataset. The loss curve was observed to converge at around 300 epochs. The correspondence was seen to be fairly high for both training and validation loss curve between the LSTM and LSTM ensembled network. The LSTM model generated a MAPE value of 0.6130 for an hour ahead forecast.

For the LSTM Ensemble method, the total training period for the dataset was found to be around 226.16 minutes, which is double longer than the LSTM method due to four different networks working in tandem. The epoch was also kept at 300 for this case for a standard comparison with the LSTM network. The convergence and loss correspondence pattern found here did not vary much from the LSTM approach. The ensembled model produced an average MAPE of 0.5723, which is 0.04% lower than the LSTM only method.

Fig. 3 and Fig. 4 show the comparison between training and loss validation between LSTM and LSTM ensemble approaches.

Both the LSTM method and the LSTM Ensembled method yielded quite similar results with similar loss patterns. No radical difference was observed between the outputs of both of the models. The training and validation losses were seen to be satisfyingly fitting. Although, from a comparative point

Fig. 4: Validation loss comparison for LSTM and LSTM Ensembled Network

Fig. 5: Actual vs Predicted load for July 2, 2014 with LSTM and LSTM ensemble approach

of view, the LSTM Ensembled approach appeared to have a relatively smoother and faster error convergence, as can be seen observing the loss figures.

A certain timeframe had been taken as a test sample to assess the performance of the models. The objective was to observe the contrast between the forecast performances of both of the models. With this purpose in mind, the actual load curve and the forecasted load curve had been mapped and juxtaposed to get a comparative view of both of the models' performances. The curves displayed in Fig. 5 exhibit a significant accuracy level for both of the models.

LSTM ensemble approach showed a better correspondence for the actual vs. predicted load. Nevertheless, there were slight deviations in both models' trajectory curves, especially in the transitional regions and sudden bends. The deviations were mostly observed in the peak hour region, whereas in the rest of the regions, both curves were seen to overlap closely. The predicted and actual curves were virtually indistinguishable for off-peak hours. However, the ensemble model had a

Fig. 6: 95% prediction interval for a day in January 2014 for hour ahead forecast.

relatively lower deviation than the LSTM model. Throughout the whole time series, the prediction curve appeared to be fairly consistent with the actual load curve.

We had further carried out probabilistic forecasting for this particular strategy to check its probabilistic nature. The probabilistic curve was determined within the 95% uncertainty in the forecast. Bootstrapping method was used for determining the probabilistic forecast curve (shown in Fig. 6).

The probabilistic curve showed a higher degree of confidence, as can be visualized from the stretch of the shade. Our forecast accuracy falls within this region. It further reassures that the models are working with decent reliability for operating within a short region of uncertainty.

B. Day ahead forecast

For the day-ahead forecast, similar procedures were followed as were done with the hour ahead forecast. The model was tweaked in places to produce day ahead forecasts instead of hour ahead ones. It trained faster than an hour-ahead forecast due to its reduced sample feeds. The total time spent training the LSTM model had been 4.43 minutes, which is around 22 times less than the hour ahead forecast training period. The inclusion of a higher number of features that had been crafted for this model caused the loss curve to converge at a higher epoch than the hour ahead scenario.

The convergence was found to be at 1200 epochs, and no further improvement could be observed beyond this level. The loss curves had shown similar fitting characteristics just as they did for the hour ahead scenario, although the noises were seen to be much higher. The LSTM model generated a Test MAPE of 3.0818 for a day ahead forecast after epoch completion.

For LSTM Ensemble, the test MAPE was found to be 2.9440 after epoch completion. The ensemble model took longer to be trained due to multiple networks working in tandem, but it performed a lot faster compared to the hour ahead forecast model. It took in a total of 14.61 minutes to train the LSTM ensemble model for day ahead forecast, which is 16 times less than the hour ahead forecasting needed. Fig. 8

Fig. 7: Comparison of training loss between LSTM and LSTM ensemble approach for day ahead forecast

Fig. 8: Comparison of validation loss between LSTM and LSTM ensemble approach for day ahead forecast

and Fig. 7 and 8 highlight the contrast between the LSTM and LSTM ensemble approaches regarding training and validation losses for a day ahead forecast. The fitting nature between the loss curves was found to be fairly satisfactory in order to be reassured of its efficacy.

Similarly to the hour ahead forecast, LSTM ensemble network was more efficient than the LSTM network, with a 0.1378% error reduction. Figure 9 juxtaposes the predicted and the actual load curves for both networks in order to better visualize their relative performances.

The efficiency of both models appeared to be relatively poorer for day-ahead forecast than an hour-ahead forecast. The gap between forecast points with actual load points for day ahead forecasts were larger than hour-ahead forecast due to the higher MAPE associated with it. However, the value was not alarmingly high as to disregard its utility. A probabilistic curve had been determined to visualize the uncertainty of this model. Figure 11 shows the prediction curve with a 95% confidence interval for the LSTM Ensemble approach. Like in hour ahead

Fig. 9: Actual load and Predicted load for a day in July 2, 2014 with LSTM and LSTM ensemble approach

forecast, LSTM method was skipped to avoid redundancy.

From the probabilistic curve, it can be inferred that the degree of reliability for the prediction model is pretty high, judging from the range of its uncertainty. Such findings corroborate that our proposed network functions pretty resiliently across the time series.

Furthermore, we substantiated the fact that LSTM ensemble model performs relatively better than other conventional models, for example, Feedforward Neural Network model (FFNN), Convolutional Neural Network model (CNN), Recurrent Neural Network Model (RNN), Residual Neural Network model (Resnet), Gated Recurrent Unit model (GRU), Autoencoder model, and Radial Basis Function model (RBFNN). From the model performances. It can also be seen that conventional LSTM model outperforms other neural network models as laid above, which concurs with other studies that also have found out a higher degree of performance for conventional LSTM models [44]. Previously we had found out that our proposed LSTM ensemble network outperforms conventional LSTM networks in terms of forecast accuracy. From the tested models, the error was found to be the lowest for our proposed ensemble network whereas the comparatively higher one was found out to be for RBFNN. Figure 10 highlights the comparative error chart for all the nine neural networks mentioned above.

However, the significance of this study is that it establishes that LSTM ensemble model performs better than other conventional neural network model for STLF. Although the improvement might seem little, it can be considered noteworthy if we take into account the economic ramifications that it might come with. The improvement that had been found out through this study has the potential to save thousands of dollars for utility companies. As per Hobbs analysis[48], a 1% reduction in error can save utilities 1.6 million dollars, following which we can conclude that the LSTM ensemble method, which has a 0.04% error reduction than LSTM only method, can save utilities 64000 dollars for hour-ahead forecasts and 224000 dollars for day-ahead forecasts due to a 0.14% error reduction.

Fig. 10: MAPE comparison chart for LSTM ensembled, LSTM, FFNN, CNN-1D, RNN, GRU, Resnet, Autoencoder, RBFNN models

Fig. 11: 95% Prediction interval for a day in January 2014 for a day ahead forecast

From the chart in figure 10, it is further evident that LSTM ensembled model can save utilities more financial resources compared to any other conventional neural network model. For instance, compared to the RNN model, the LSTM ensembled model can save 246000 dollars for hour-ahead and 2.424 million dollars in case of day-ahead forecasts.

V. CONCLUSION

Our findings can safely conclude that LSTM ensembled strategies outperform traditional LSTM models as well as other conventional neural network models. The performance is consistent for both hour-ahead and day-ahead forecasts. Furthermore, although the day-ahead forecast was found to be associated with a higher error value than the hour-ahead forecasts, the error margin was not seen to exceed a concerning threshold, which might have nullified its utility. Our proposed ensembled network carries the potential to assist utilities in

making more appropriate short-time load forecasts at the expense of relatively low resources along with a relatively higher degree of reliability. It also sheds some valuable insight into how ensemble learning fares better than single network learning for time series data. The output of this particular study reaffirms and quantifies that position.

The model might have performed better if more meteorological parameters could have been funneled into it alongside the temperature data. Humidity, for example, plays a very significant role in terms of load usage. Higher humidity drives up load demands in warm climates, and low humidity drives it down. Unfortunately, this study could not utilize such parameters because they were unavailable in the acquired dataset. Same with wind speed, precipitation, and other similar meteorological data, the unavailability of which handicapped us from doing any further experimentation with additional parameters and hobbled us from modifying our models any further. Such modifications might have had the potential to increase the precision of our network even further, presenting us with an even more preferable scenario.

REFERENCES

- [1] G. Gross and F. Galiana, "Short-term load forecasting," *Proceedings of the IEEE*, vol. 75, no. 12, pp. 1558–1573, 1987.
- [2] I. Moghram and S. Rahman, "Analysis and evaluation of five short-term load forecasting techniques," *IEEE Transactions on Power Systems*, vol. 4, no. 4, pp. 1484–1491, 1989.
- [3] S. Rahman and R. Bhatnagar, "An expert system based algorithm for short term load forecast," *IEEE Transactions on Power Systems*, vol. 3, no. 2, pp. 392–399, 1988.
- [4] K. Jabbour, J. Riveros, D. Landsbergen, and W. Meyer, "Alfa: automated load forecasting assistant," *IEEE Transactions on Power Systems*, vol. 3, no. 3, pp. 908–914, 1988.
- [5] S.-J. Huang and K.-R. Shih, "Short-term load forecasting via arma model identification including non-gaussian process considerations," *IEEE Transactions on Power Systems*, vol. 18, no. 2, pp. 673–679, 2003.
- [6] C.-M. Lee and C.-N. Ko, "Short-term load forecasting using lifting scheme and arima models," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5902–5911, 2011.
- [7] W. Christiaan, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 900–911, 1971.

- [8] J. W. Taylor and P. E. McSharry, "Short-term load forecasting methods: An evaluation based on european data," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2213–2219, 2007.
- [9] H. Al-Hamadi and S. Soliman, "Short-term electric load forecasting based on kalman filtering algorithm with moving window weather and load model," *Electric power systems research*, vol. 68, no. 1, pp. 47–59, 2004.
- [10] N. Amjadi, "Short-term hourly load forecasting using time-series modeling with peak load estimation capability," *IEEE Transactions on power systems*, vol. 16, no. 3, pp. 498–505, 2001.
- [11] H.-T. Yang and C.-M. Huang, "A new short-term load forecasting approach using self-organizing fuzzy armax models," *IEEE Transactions on Power Systems*, vol. 13, no. 1, pp. 217–225, 1998.
- [12] A. D. Papalexopoulos and T. C. Hesterberg, "A regression-based approach to short-term system load forecasting," *IEEE Transactions on power systems*, vol. 5, no. 4, pp. 1535–1547, 1990.
- [13] T. Haida and S. Muto, "Regression based peak load forecasting using a transformation technique," *IEEE Transactions on Power Systems*, vol. 9, no. 4, pp. 1788–1794, 1994.
- [14] S. C. Pandian, K. Duraiswamy, C. C. A. Rajan, and N. Kanagaraj, "Fuzzy approach for short term load forecasting," *Electric power systems research*, vol. 76, no. 6-7, pp. 541–548, 2006.
- [15] R. Mamlook, O. Badran, and E. Abdulhadi, "A fuzzy inference model for short-term load forecasting," *Energy Policy*, vol. 37, no. 4, pp. 1239–1248, 2009.
- [16] X. Zhang, J. Wang, and K. Zhang, "Short-term electric load forecasting based on singular spectrum analysis and support vector machine optimized by cuckoo search algorithm," *Electric Power Systems Research*, vol. 146, pp. 270–285, 2017.
- [17] D. Niu, Y. Wang, and D. D. Wu, "Power load forecasting using support vector machine and ant colony optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2531–2539, 2010.
- [18] S. Li, L. Goel, and P. Wang, "An ensemble approach for short-term load forecasting by extreme learning machine," *Applied Energy*, vol. 170, pp. 22–29, 2016.
- [19] E. Ceperic, V. Ceperic, and A. Baric, "A strategy for short-term load forecasting by support vector regression machines," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4356–4364, 2013.
- [20] G. Dudek, "Short-term load forecasting using random forests," in *Intelligent Systems' 2014*. Springer, 2015, pp. 821–828.
- [21] E. Aguilar Madrid and N. Antonio, "Short-term electricity load forecasting with machine learning," *Information*, vol. 12, no. 2, p. 50, 2021.
- [22] J. Luo, T. Hong, and S.-C. Fang, "Benchmarking robustness of load forecasting models under data integrity attacks," *International Journal of Forecasting*, vol. 34, no. 1, pp. 89–104, 2018.
- [23] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.
- [24] K. Lee, Y. Cha, and J. Park, "Short-term load forecasting using an artificial neural network," *IEEE transactions on power systems*, vol. 7, no. 1, pp. 124–132, 1992.
- [25] S. R. Salkuti, "Short-term electrical load forecasting using radial basis function neural networks considering weather factors," *Electrical Engineering*, vol. 100, no. 3, pp. 1985–1995, 2018.
- [26] C. Guan, P. B. Luh, L. D. Michel, Y. Wang, and P. B. Friedland, "Very short-term load forecasting: wavelet neural networks with data pre-filtering," *IEEE Transactions on Power Systems*, vol. 28, no. 1, pp. 30–41, 2012.
- [27] P. Ray, S. K. Panda, and D. P. Mishra, "Short-term load forecasting using genetic algorithm," in *Computational Intelligence in Data Mining*. Springer, 2019, pp. 863–872.
- [28] Z. Bashir and M. El-Hawary, "Applying wavelets to short-term load forecasting using pso-based neural networks," *IEEE transactions on power systems*, vol. 24, no. 1, pp. 20–27, 2009.
- [29] S. Li, P. Wang, and L. Goel, "Short-term load forecasting by wavelet transform and evolutionary extreme learning machine," *Electric Power Systems Research*, vol. 122, pp. 96–103, 2015.
- [30] A. Al Mamun, M. Sohel, N. Mohammad, M. S. H. Sunny, D. R. Dipta, and E. Hossain, "A comprehensive review of the load forecasting techniques using single and hybrid predictive models," *IEEE Access*, vol. 8, pp. 134 911–134 939, 2020.
- [31] M. Barman, N. D. Choudhury, and S. Sutradhar, "A regional hybrid goa-svm model based on similar day approach for short-term load forecasting in assam, india," *Energy*, vol. 145, pp. 710–720, 2018.
- [32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [33] S. M. J. Jalali, S. Ahmadian, A. Khosravi, M. Shafie-khah, S. Nahavandi, and J. P. Catalao, "A novel evolutionary-based deep convolutional neural network model for intelligent load forecasting," *IEEE Transactions on Industrial Informatics*, 2021.
- [34] M. Alhussein, K. Aurangzeb, and S. I. Haider, "Hybrid cnn-lstm model for short-term individual household load forecasting," *IEEE Access*, vol. 8, pp. 180 544–180 557, 2020.
- [35] B. Farsi, M. Amayri, N. Bouguila, and U. Eicker, "On short-term load forecasting using machine learning techniques and a novel parallel deep lstm-cnn approach," *IEEE Access*, vol. 9, pp. 31 191–31 212, 2021.
- [36] 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)*. IEEE, 2017, pp. 594–598.
- [37] S. Ryu, J. Noh, and H. Kim, "Deep neural network based demand side short term load forecasting," *Energies*, vol. 10, no. 1, p. 3, 2017.
- [38] X. Kong, C. Li, F. Zheng, and C. Wang, "Improved deep belief network for short-term load forecasting considering demand-side management," *IEEE Transactions on Power Systems*, vol. 35, no. 2, pp. 1531–1538, 2019.
- [39] X. Zhang, R. Wang, T. Zhang, Y. Liu, and Y. Zha, "Short-term load forecasting using a novel deep learning framework," *Energies*, vol. 11, no. 6, p. 1554, 2018.
- [40] B. Zhang, J.-L. Wu, and P.-C. Chang, "A multiple time series-based recurrent neural network for short-term load forecasting," *Soft Computing*, vol. 22, no. 12, pp. 4099–4112, 2018.
- [41] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—a novel pooling deep rnn," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2017.
- [42] M. Zhang, Z. Yu, and Z. Xu, "Short-term load forecasting using recurrent neural networks with input attention mechanism and hidden connection mechanism," *IEEE Access*, vol. 8, pp. 186 514–186 529, 2020.
- [43] 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 Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2017.
- [44] —, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2017.
- [45] J. Li, D. Deng, J. Zhao, D. Cai, W. Hu, M. Zhang, and Q. Huang, "A novel hybrid short-term load forecasting method of smart grid using mlr and lstm neural network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2443–2452, 2021.
- [46] H. Zheng, J. Yuan, and L. Chen, "Short-term load forecasting using emd-lstm neural networks with a xgboost algorithm for feature importance evaluation," *Energies*, vol. 10, no. 8, p. 1168, 2017.
- [47] H. Choi, S. Ryu, and H. Kim, "Short-term load forecasting based on resnet and lstm," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–6.
- [48] B. F. Hobbs, S. Jitrapakulsarn, S. Konda, V. Chankong, K. A. Loparo, and D. J. Maratukulam, "Analysis of the value for unit commitment of improved load forecasts," *IEEE Transactions on Power Systems*, vol. 14, no. 4, pp. 1342–1348, 1999.
- [49] M. Tan, S. Yuan, S. Li, Y. Su, H. Li, and F. He, "Ultra-short-term industrial power demand forecasting using lstm based hybrid ensemble learning," *IEEE transactions on power systems*, vol. 35, no. 4, pp. 2937–2948, 2019.
- [50] K. Yan, W. Li, Z. Ji, M. Qi, and Y. Du, "A hybrid lstm neural network for energy consumption forecasting of individual households," *Ieee Access*, vol. 7, pp. 157 633–157 642, 2019.
- [51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [52] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
- [53] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.
- [54] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.