INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

A Multi-step Short-term Load Forecasting using Hybrid DNN and GAF

Nilakanta Kshetrimayum[1]*, Khumukcham Robindro Singh[1], Nazrul Hoque[1]

1 Department of Computer Science, Manipur University, Indo-Brahma Road, Imphal, 795003, Manipur, India

OPEN ACCESS

Received: 27-12-2023

Accepted: 08-02-2024

Published: 29-02-2024

Citation: Kshetrimayum N, Singh KR, Hoque N (2024) A Multi-step Short-term Load Forecasting using Hybrid DNN and GAF. Indian Journal of Science and Technology 17(11): 1016-1027. https://doi.org/10.17485/IJST/v17i11.3246

* **Corresponding author**.

nilakanta.kakching@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Background: Short-term Load Forecasting (STLF) is vital for grid stability, ensuring a steady power supply and resource efficiency. However, the literature review underscores imperfections in current methods, emphasizing the necessity for additional research in this domain. **Objectives:** This study introduces an effective framework for multi-step STLF, enhancing predictive accuracy by integrating state-of-the-art DNN models like Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN) in a hybrid architecture, leveraging their complementary strengths. **Method:** A parallel hybrid network consisting of LSTM and a Two-Dimensional Convolutional Neural Network (2D-CNN), which operates load sequence input and 2D converted load input, is employed. Their combined context vector is finally used for predicting the preceding 24 load sequences. Extensive comparative testing involves various existing STLF methods (Artificial Neural Network (ANN), LSTM Sequence-to-Sequence (LSTM-S2S), Gated Recurrent Unit with Genetic Algorithm (GRU-GA), CNN-LSTM, GRU-CNN, and Parallel LSTM and CNN Network (PLCNet)) experimented across nine load datasets, using Mean Absolute Percentage Error (MAPE) as the evaluation metric. **Findings:** The proposed model demonstrates enhanced MAPE, with values of 3.34, 5.67, 4.92, 4.84, 5.24, 4.73, 5.15, 5.49, and 3.96 across nine datasets. The critical distance diagram further validates these results. The findings from this comparative analysis underscore the efficacy of the proposed multi-step STLF method, showcasing significant improvements in forecasting accuracy. **Novelty:** The presented network architecture is novel in its fusion of LSTM and 2D-CNN networks through the Gramian Angular Field (GAF) technique. Additionally, the parallel operation of LSTM and 2D-CNN generates distinct context vectors combined to form the final output.

Keywords: Parallel CNNLSTM; Gramian Angular Field; Electricity Demand Prediction; Short-term Load Forecasting; Multi-step Forecasting

1 Introduction

Short-term load forecasting (STLF) is crucial for efficiently managing energy resources, storage, and generation in the power grid. Yet, accurately forecasting this demand

is challenging due to its constantly fluctuating and unpredictable nature [1]. A multi-step STLF approach predicts the electricity demand over several consecutive time intervals in the immediate future, ranging from a few hours to a week [2]. Traditional time series modeling focused on capturing linear relationships and correlations among input parameters and output. As machine learning progresses, new and sophisticated techniques are emerging to learn complex patterns within data effectively and improve predictive capabilities. Nonetheless, these methods face challenges when dealing with larger training datasets that exhibit nonlinear relationships among features. Recent STLF approaches have shifted towards leveraging Deep Neural Network (DNN) techniques to address this limitation. Notably, the Multi-Layer Perceptron [3] exhibits plain architecture and can be trained reasonably, producing better results. Another DNN architecture deemed more suitable for STLF incorporates a One-Dimensional (1D) Convolutional Neural Network (CNN) to extract features from sequential load data [4]. However, its effectiveness diminishes with longer sequences. A more efficient DNN structure for handling sequential data entails the integration of a Recurrent Neural Network (RNN), including a Long Short Term Memory (LSTM) [5] and a Gated Recurrent Unit (GRU) [6]. Hybrid models are employed to leverage the strengths of both CNN and RNN. Many hybrid CNN and RNN models, such as CNN-LSTM [7] and CNN-GRU [8], adopt a linear architecture. In these models, CNN is initially utilized to extract features, and LSTM or GRU subsequently processes the extracted features. However, this arrangement can potentially result in the loss of crucial temporal information before it reaches the RNN layer. In certain hybrid models, CNN and LSTM modules are used independently, and their outputs are combined to create a unified feature vector. Examples of such models, like LSTM-CNN [9] and CNN-GRU [10] , have demonstrated the capability to yield superior results. In models employing 1D-CNN, the inherent limitation of 1D-CNN leads to insufficient modeling of spatial features, potentially impacting overall predictive performance. This motivates us to adopt a Two-Dimensional CNN (2D-CNN) network for a more comprehensive extraction of spatial features from load data. Concurrently, LSTM is utilized to capture temporal features. The 2D-CNN and LSTM modules run parallel, handling load sequences and 2D load inputs, respectively. We also utilized a Gramian Angular Field (GAF) technique as feature processing to transform the sequence to a 2D input. This additional feature vector derived from the 2D-CNN introduces an extra layer of contextual features to complement those generated by the LSTM network, ultimately creating a highly effective context vector. In summary, the main contributions of the research outlined in this paper are as follows.

1. We introduce a novel Multi-step STLF model called Parallel 2D CNN and LSTM (P2DCNNLSTM) that leverages the parallel fusion of 2D-CNN and LSTM architectures. The context vectors from LSTM and 2D-CNN are combined, resulting in a more effective context vector.
2. We propose a preprocessing framework utilizing the GAF technique, transforming load series into 2D vectors. These vectors serve as additional input for our proposed model, enhancing its performance.
3. We conduct an extensive experiment to assess the effectiveness of our proposed P2DCNNLSTM model. Comparative evaluations against existing models from previous work, such as an Artificial Neural Network (ANN) [11], LSTM Sequence to Sequence (LSTM-S2S) [12], GRU with Genetic algorithm (GRU-GA) [13], CNN-LSTM [14], GRU-CNN [15], and Parallel LSTM CNN Network (PLCNet) [16] are performed. Furthermore, a Critical Distance Diagram (CDD) statistically verifies the proposed model's effectiveness.

This paper's remaining parts are organized as follows: Section 2 explains the methodology of the proposed hybrid Multi-Step STLF framework. Section 3 provides experiment results and analyses the results. Finally, the conclusion is drawn in section 4.

2 Methodology

This section details the research problem and the methodologies employed to implement the proposed STLF model.

2.1 Preliminaries

2.1.1 Gramian angular field

To maximize the efficiency of the CNN network, we employ a 2D-CNN module that requires input features in a 2D format. GAF is a sequence encoding method that converts 1D data into a 2D representation [17]. This technique leverages the concept of representing a time series in a polar coordinate system where the sequence values are encoded as the angular cosine and the position of the values in the sequence as the radius. It produces a GAF image, a gram matrix with values represented by either the trigonometric sum or trigonometric difference between each corresponding angle. Figure 1 represents this transformation.

The following are the steps required in this transformation.

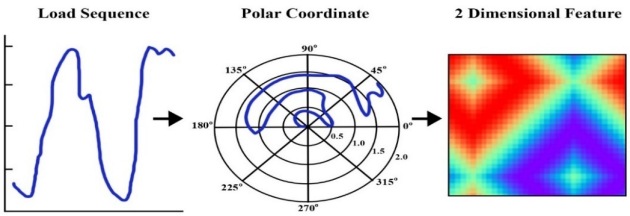

Fig 1. Transformation of sequence into 2D features using GAF

1. For a given sequence $X = \{x_1, x_2, x_3, \ldots, x_N\}$, rescale all the values of X to $\widehat{X}$ in the range of $[-1, 1]$ using the min-max scalar. The formula of min-max is given in the equation (1).

$$\widehat{X} = \frac{2X_i - max(X) - min(X)}{max(X) - min(X)} \tag{1}$$

2. Transform the rescaled values, $\widehat{X}$, into a polar coordinate system where values in sequence are encoded as angles and the corresponding position of values in sequence as radius with the following equations (2 - 3).

$$\theta_i = arccos\left(\widehat{X}_i\right), \quad -1 \leq \widehat{X}_i \leq 1, \widehat{X}_i \in \widehat{X} \tag{2}$$

$$r_i = \frac{i}{N}, \qquad i \in N \tag{3}$$

This mapping is a bijective function since $cos(\theta)$ is a monotonic function for θ between 0 and π. It also preserved the absolute temporal correlation within different time intervals, i.e., the time increases as we move from the left-top corner to the right-down corner.

3. Then $n * n$ GAF image can be generated in which ij^{th} elements are defined as cosine summation or difference of i^{th} and j^{th} angles in the polar coordinate system. So, the Gramian Angular Summation Field (GASF) and the Gramian Angular Difference Field (GADF) are given by equations (4 - 5).

$$GASF\,(i, j) = cos(\theta_i + \theta_j) \tag{4}$$

$$GADF\,(i, j) = cos(\theta_i - \theta_j) \tag{5}$$

4. Finally, to reduce the size of the generated image, a Piecewise Aggregation Approximation can be applied to smooth the results while retaining the trends.

2.1.2 Convolutional Neural Network

CNN simulates the working of the brain's visual cortex[18]. This network's strong capability to extract features from 2D data makes it a perfect tool for image classification. This neural network includes multiple stacks of convolutions and pooling layers, followed by a fully connected layer. Figure 2 provides a simple CNN topology.

The fundamental purpose of the convolution layer is to perform convolution operations on input data using a kernel, a weight matrix, by sliding it across the input. This operation relies on local connection and weight sharing. The pooling layer reduces the dimension of each feature by retaining only crucial values. For instance, the Max-pooling operation keeps the maximum feature value while Average-pooling retains the mean value. The fully connected layer is a dense network usually employed for generating the final output. In classification problems, the last layer typically features the same number of neurons as the classes, while for regression problems, the number of neurons in the final layer is determined by the number of values to be predicted.

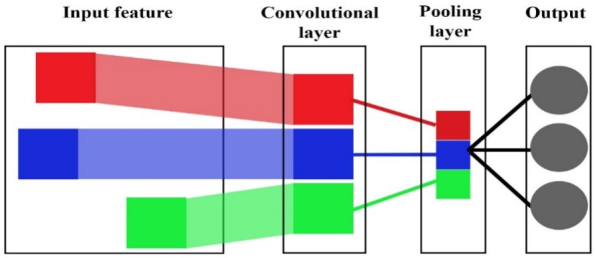

Fig 2. A CNN architecture

2.1.3 Long short-term memory

An RNN is a specialized form of DNN designed for handling sequential inputs. It excels in capturing recurrent patterns within sequence data. Nonetheless, it grapples with inherent issues like vanishing and exploding gradients, leading to performance decline as the input's time interval increases—rendering it inadequate for addressing long-term dependencies in input. In contrast, LSTM, a derivative of RNN, excels in retaining long-term context and solving gradient vanishing concerns. A block diagram of an LSTM cell is given in Figure 3.

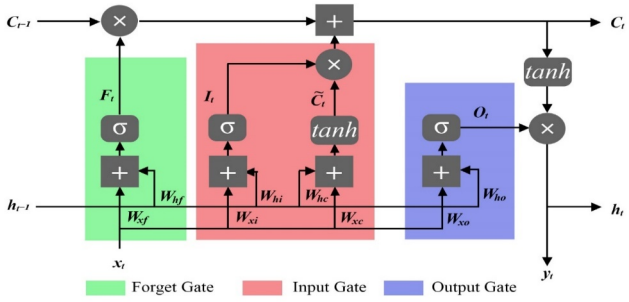

Fig 3. An LSTM architecture

LSTM is a complex model with several gates, like input, forget, and output gates, along with a memory cell. This intricate network enables it to effectively capture and retain long-term patterns in the sequence data. The following mathematical equations (6 - 13) show the computation of the LSTM cell.

$$f_t = \sigma\left(W_f\left(X_t, h_{t-1}\right] + b_f\right) \qquad (6)$$

$$C_{t-1} * f_t = 0, \quad if\ f_t \approx 0 \qquad (7)$$

$$C_{t-1} * f_t = C_t, \quad if\ f_t \approx 1 \qquad (8)$$

$$i_t = \sigma\left(W_i\left(X_t, h_{t-1}\right] + b_i\right) \qquad (9)$$

$$n_t = tanh\left(W_i\left(X_t, h_{t-1}\right] + b_i\right) \qquad (10)$$

$$C_t = C_{t-1} * f_t + i_t * n_t \quad (11)$$

$$o_t = \sigma(W_o(X_t, h_{t-1}] + b_0) \quad (12)$$

$$h_t = o_t * tanh(C_t) \quad (13)$$

A cell state is maintained inside the LSTM cell, crucial in generating the output. The manipulation of this cell state is governed by distinct gates, including the forget gate, responsible for regulating the removal of information from the cell state; the input gate, which controls the infusion of new information into the cell state; and the output gate, which determines the selection of information to be outputted from the cell state. This intricate interplay of gates orchestrates the flow of information within the LSTM cell, contributing to its ability to capture and retain relevant patterns over sequential data.

2.2 Problem description

This study addresses a Multi-step STLF problem, focusing on predicting the day-ahead load demand patterns. It involves utilizing the preceding seven-day load sequence, i.e., 168 load data points, to forecast the following 24 load sequences of the next day. Mathematically, $X = \{X_{t-167}, X_{t-166}, X_{t-165}, \ldots, X_{t-1}, X_t | X_T \varepsilon R\}$ be the sequence of electricity load values used for predicting the next 24 load values, denoted by $Y = \{X_{t+1}, X_{t+2}, X_{t+3}, \ldots, X_{t+24} | X_T \varepsilon R\}$.

Figure 4 shows the above setting of Multi-Step STLF.

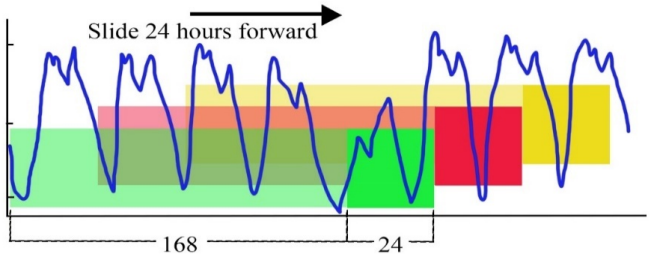

Fig 4. Multi-Step Forecasting

Here, the whole sequence is divided into individual segments to produce data instances where the feature consists of 168 load values and the label consists of the next 24 load values. Each instance of the supervised data is generated by sliding 24 points forward.

2.3 Proposed Multi-step STLF Framework

The proposed framework includes - 1) Data preprocessing steps such as data-split for training and testing datasets, reframing, 2D feature transformation, and scaling; 2) Modeling and training; and 3) Model evaluation. Figure 5 illustrates the different components of the proposed multi-step STLF framework. The subsequent sections provide detailed explanations of these steps.

2.3.1 Data preprocessing

Hourly short-term load data typically exhibits daily, weekly, monthly, and yearly patterns. Missing values or disorganized sequences can potentially disrupt these discernible patterns. In the initial preprocessing phase, we systematically detect and rectify any missing values by substituting them with the corresponding values from the preceding day at the same hour. Then, the time series load data is partitioned into training and testing sets. The training set constitutes the entire sequence, excluding the sequence belonging to the last 100 days, and is utilized for training the model. The testing set, consisting of sequences of the last 100 days, evaluates the model's performance. This testing set is an out-of-sample dataset, ensuring a thorough and accurate

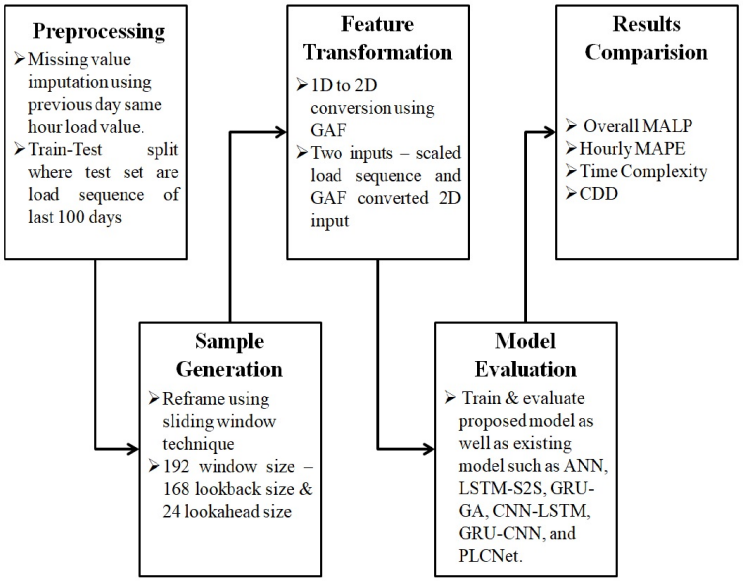

Fig 5. The proposed Multi-Step STLF Framework

evaluation process. Feature scaling is applied to both datasets using a scaler exclusively trained on the training dataset. This approach is employed to avoid any accidental transfer of information from the training phase to the testing phase, ensuring the integrity of the testing process. Applying feature scaling to the inputs of DNN-based models offers several advantages. Firstly, it accelerates the convergence process during model training. Secondly, it ensures an impartial input for the model by standardizing all features to a common range. Lastly, it sometimes enhances the model's performance. Various scaling techniques have been documented in existing literature, including standard, min-max, max-absolute, and robust scaling. We adopt the min-max scaling method within our framework, as illustrated above in Equation 1.

Using the entire sequence of short-term load data as input is unsuitable for an ML-based model, given that model learning relies on numerous supervised instances. A prevalent practice involves transforming the sequential data into a supervised dataset through a technique known as reframing. Sliding window techniques are commonly employed for this purpose. In our study, we adapted a modified version of this technique to handle our required configuration of input data. Figure 6 outlines this Algorithm to convert load sequence to supervise data.

This procedure segments the sequence into multiple segments, each aligning with instances comprising features and load labels. The lookback parameter dictates the count of feature instances, while the lookahead decides the number of output instances. During each iteration, it computes the indices linked to features and labels and then populates the respective lists with the corresponding values. As previously stated, our framework adopts a lookback size of 168, representing an hourly load sequence for seven days or a week, and a lookahead size of 24, corresponding to the prediction of 24 load values for the subsequent day. The feature extraction process involves converting the sequence or 1D representation into a matrix or image, specifically a 2D representation, utilizing the GAF technique, as detailed in section 2.1.

The deliberate selection of the GAF method for representing 1D signals in a 2D space is grounded in its unique advantages over simpler methods. The GAF method stands out for its capacity to capture intricate temporal patterns and relationships within the signal, providing a comprehensive representation crucial for preserving the temporal dynamics in load sequence data. Unlike simpler methods, GAF excels in retaining essential structural information, particularly valuable for non-linear and complex patterns where preserving phase information is essential. The versatility of the GAF method, balancing complexity and efficiency, makes it a robust choice applicable across diverse signal processing domains. In essence, the decision to employ the GAF method stems from its ability to faithfully capture the temporal intricacies of 1-D signals in a 2-D space, addressing nuances that might be overlooked by simpler methods. In this phase, each sequence of size 168 is transformed into a feature matrix with dimensions 168x168. This transformation aims to create a highly abstract feature, essentially represented as a 2D image, facilitating effective processing by a 2D-CNN.

Algorithm 1: Convert load sequence data to supervised data

Input: Load sequence data (S), lookback, and lookahead

Output: Feature vector (X) and Target label (Y)

```
procedure Reframe(S, lookback, lookahead)
    X, Y ← new empty lists
    start ← 0
    n ← length(S)
    for i ← 0 to n do
        feature_end ← start + lookback
        label_end ← feature_end + lookahead
        if label_end ≤ n then
            Insert S[start: feature_end] to X
            Insert S[feature_end: label_end] to Y
        end if
        start ← start + lookahead
    end for
end procedure
```

Fig 6. Convert Load sequence data to supervised data

2.3.2 The Proposed P2DCNNLSTM network

The proposed P2DCNNLSTM model comprises two distinct modules of LSTM and 2DCNN running in parallel. The LSTM network handles the load sequence input, while the 2D-CNN network processes the GAF-transformed load 2D input. A rich context vector is obtained by concatenating the outputs of these two networks. This feature vector is then used to produce the final output. This network architecture is outlined in Figure 7 .

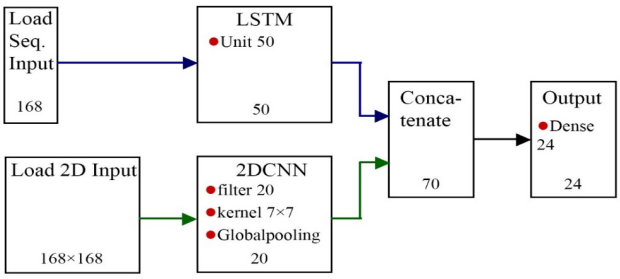

Fig 7. Network Architecture of the proposed P2DCNNLSTM model

As illustrated, the network takes in two versions of load inputs, i.e., a sequence-based input with dimensions of (N,168,1) and an image-based input with dimensions of (N,168,168,1), N is the number of samples. The sequence and image-based inputs are processed by LSTM and 2D-CNN networks, respectively. This LSTM network has a single LSTM layer containing 50 neurons and produces a feature vector of size (N,50). The 2D-CNN network comprises 20 filters, each with dimensions (7,7), applied for convolving the 2D feature. Following this, a Global Average Pooling layer calculates the average of the resultant feature vector, leading to an output feature vector of dimension (N,20). Combining these two output vectors yields a unified feature vector, which generates the final output through subsequent processing stages.

2.4 Experimental setup

All the experiments were conducted on Google Collaboratory, a cloud-based Python notebook environment that offers GPU acceleration for efficient model training. Data preprocessing is performed using the sklearn package, while the pyts package facilitates the GAF transformation. Our proposed P2DCNNLSTM model and other competitive models were developed using the Tensorflow-Keras framework, and to determine the optimum hyperparameters, we utilized the KerasTunner. Additional Python libraries, including numpy and pandas, were employed for data manipulation tasks. Visualizations such as graphs and diagrams were generated using the matplotlib and seaborn packages. Different STLF models are assessed using the MAPE as the comparative metric. The mathematical representation of this metric is provided in the equation (14).

$$MAPE = \frac{1}{n}\left(\sum_{i=1}^{n}\frac{\left|L_i^{true} - L_i^{forecast}\right|}{L_i^{true}}\right) \times 100 \quad (14)$$

where $L_i^{forecast}$ is the predicted load of ith element in the test set, L_i^{true} is the actual load of ith element in the test set, and n is the number of time steps in the test set. Higher forecasting accuracy is attained when the value of the MAPE is small, as the MAPE quantifies the percentage difference between the forecasted and actual observed values. When the MAPE is small, it indicates that the forecasted values are very close to the actual values, resulting in a minimal deviation between the predicted and observed data points.

2.4.1 Dataset

The complexity of short-term load patterns stems from the nonlinearity caused by diverse seasonal and external influences. To demonstrate the efficacy of our hybrid approach for STLF, we utilize nine datasets from the PJM collection, which contain hourly-indexed electricity consumption data. These datasets showcase the real challenges of the problem and serve as a basis for evaluating the performance of our proposed solution. Table 1 below describes the instance size and data ranges of these datasets.

Table 1. Description of various datasets employed in the experiment

Dataset	From	To	Total instances
AEP	01/10/2004	02/08/2018	1,21,296
COMED	01/01/2011	03/08/2018	66,504
DEOK	01/01/2012	03/08/2018	57,744
DUQ	01/01/2005	03/08/2018	119,088
EKPC	01/06/2013	03/08/2018	45,336
FE	01/06/2011	03/08/2018	62,880
NI	05/01/2004	01/01/2011	58,464
PJME	31/12/2002	01/01/2018	1,45,366
PJMW	01/04/2002	03/08/2018	1,43,232

3 Results and Discussion

We conducted a comparative analysis involving our proposed P2DCNNLSTM model and several other existing STLF models from recent literature. We employed the nine datasets shown in Table 1 to enhance the generality and appropriateness of the results, avoiding dependence on any specific dataset. The training set for each dataset encompassed the entire dataset, excluding the load sequence from the last 100 days. This excluded sequence served as the test set for evaluation purposes. All the models are trained for 50 epochs with a batch size of 32, a validation split ratio of 20%, Adam as the optimizer, and mean square error as training loss with the learning rate set at 0.01.

3.1 Comparing with existing methods

Table 2 presents the MAPE results for our proposed P2DCNNLSTM model and several state-of-the-art models from recent literature. The evaluation is based on the last100 days of each dataset's timesteps. Bold numbers in Table 2 signify the minimum MAPE a specific model achieves for its corresponding dataset. Notably, the P2DCNNLSTM model outperformed others by

Table 2. MAPE results of existing models vs the proposed P2DCNNLSTM

	Model	AEP	COMED	DEOK	DUQ	EKPC	FE	NI	PJME	PJMW
[11]	ANN	4.19	7.37	6.02	6.24	8.34	5.27	8.91	6.22	4.41
[12]	LSTM-S2S	4.13	5.98	5.88	5.35	5.83	4.99	6.02	6.32	4.25
[13]	GRU-GA	4.00	6.45	5.06	6.81	5.73	5.07	6.74	6.02	5.39
[14]	CNN-LSTM	3.43	6.05	5.07	4.95	6.06	5.16	5.28	6.05	4.11
[15]	GRU-CNN	3.53	6.38	5.34	5.73	5.92	6.47	5.25	7.84	4.67
[16]	PLCNet	3.61	6.56	5.00	6.63	5.29	5.11	5.55	8.06	6.56
Proposed	P2DCNNLSTM	**3.34**	**5.66**	**4.92**	**4.84**	**5.24**	**4.73**	**5.15**	**5.49**	**3.96**

attaining the minimum MAPE. On average, our model achieved an impressive 4.81% MAPE across all datasets, underscoring its significant superiority over existing STLF models. Additionally, we conducted a more detailed analysis by comparing the average MAPE for each 24 hours, as illustrated in Figure 8. The line plots depict the performance of various models across different datasets, highlighting how prediction errors increase with an increase in the forecast horizon.

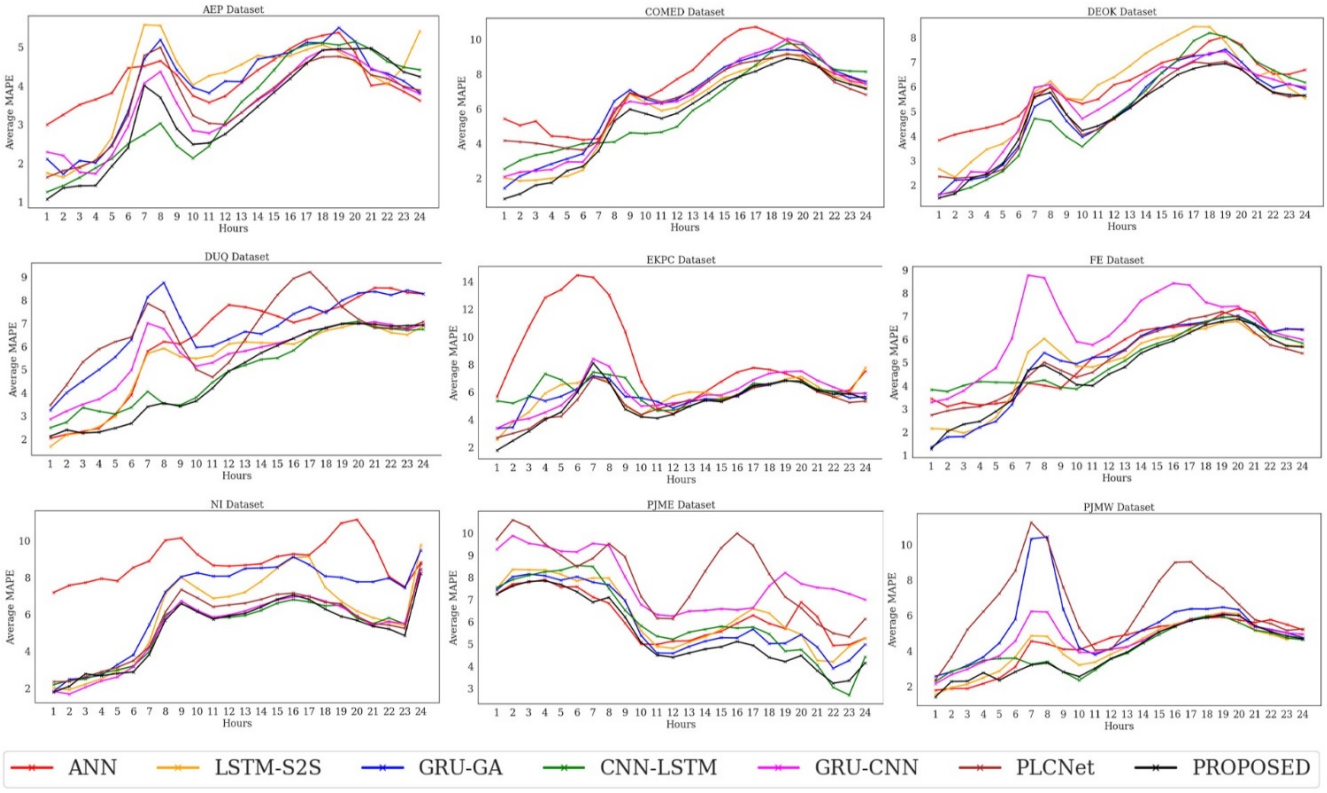

Fig 8. Plots of Average MAPE of each 24 timesteps for competing models

Two distinct patterns emerge by examining the line plots in Figure 8. Firstly, our model's line plots consistently show lower average MAPEs across all datasets. Secondly, there is a noticeable trend of increasing MAPE as the forecast hour extends into the future. This observation strongly suggests a growing level of uncertainty in predictions as the forecasting horizon expands.

To demonstrate the performance of our model and facilitate a direct comparison with existing results, we refer to the work that presented PLCNet [16]. The dataset utilized in this study comprises load and temperature data from Malaysia. We evaluate our model using the same settings employed in that work and present the results in Table 3. This table comprehensively compares the R^2 scores of the proposed P2DCNNLSTM with those of previously reported results. This comparison offers insight into how

our approach was implemented. The mathematical formula of R^2 is given in equation 15.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(L_i^{true} - L_i^{forecast}\right)^2}{\sum_{i=1}^{n}\left(L_i^{true} - \bar{L^{true}}\right)^2} \tag{15}$$

where $\bar{L^{true}}$ is the mean of the actual load values in the test set.

Table 3. Result of comparison with results from previous works

Model	R^2
DeepEnergy	87.92%
LSTM	95.21%
CNN-LSTM	96.62%
PLCNet	97.55%
Proposed(P2DCNNLSTM)	**98.60%**

The proposed P2DCNNLSTM model demonstrated remarkable performance with an impressive R^2 score of 98.6%, surpassing state-of-the-art models like PLCNet, DeepEnergy, and the hybrid CNN-LSTM model. By utilizing2DCNN on 2D transformed load series data, these models effectively incorporate crucial features into those generated by LSTM modules. The resulting integrated context vectors contribute to more accurate predictions. The outcomes of our comprehensive comparative analysis further substantiate and validate this assertion.

3.2 Training and Inference Time Comparision

Figure 8 shows a comparison of time complexity, considering both training and inferencing times for competing models. Specifically, the bar graph on the left side of Figure 9 visually represents the training times of different STLF models across nine distinct datasets.

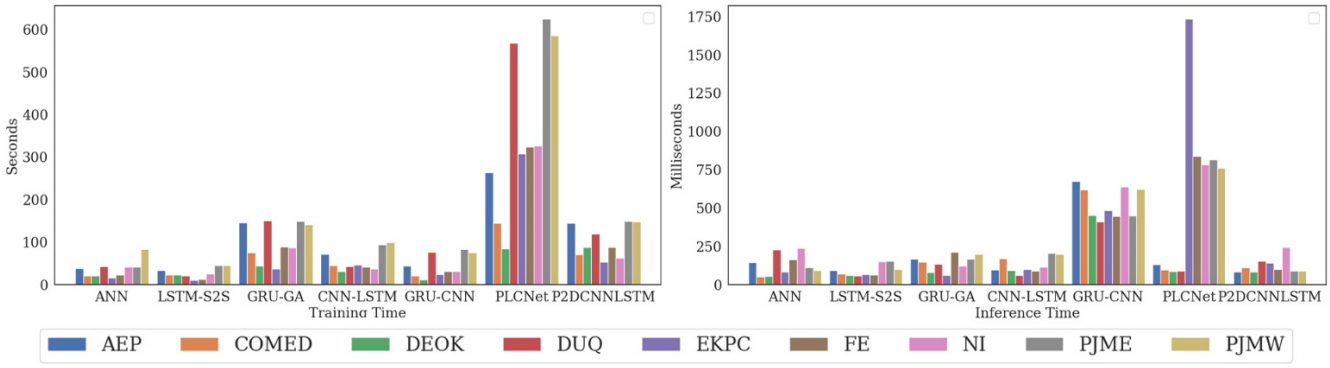

Fig 9. Bar graphs of training time (left) and inferencing time (right) of various STLF models.

Upon examination, it is observed that PLCNet exhibits the lengthiest training time, whereas the training time complexity of the P2DCNNLSTM model is comparable to that of other models. Training time depends on various parameters, such as network and hyperparameters, which may differ across different cases. For a more meaningful comparison, we evaluated the inference time required for predicting a single day's load, encompassing 24 timesteps. The bar graph on the right side of Figure 9 illustrates the inference times for predicting load values for a single day, expressed in milliseconds. GRU-CNN and PLCNet have the longest inference times among the models. Conversely, the proposed P2DCNNLSTM and other models demonstrate lower inference time complexity.

3.3 Statistical comparison: Critical Distance Diagram

To support the strength of our conclusions, we employ a statistical comparison through the Friedman-Nemenyi test. The non-parametric Friedman test utilizes a ranking-based approach to discern differences among various models. Under the null hypothesis, it is assumed that if all models perform equally, their average ranks (Rj) should be identical. We proceed with the Nemenyi post hoc test if the null hypothesis is rejected. When evaluating the performance of two models, their average ranks must exhibit a significant difference beyond a critical distance (CD) for the result to be deemed significant. The CD value is calculated by equation 15:

$$CD = q_a \times \sqrt{\frac{k(k+1)}{6N}} \tag{16}$$

where the critical value $q_{0.10}$= 2.693 is obtained from the critical value table of the two-tailed Nimenyi test; k is the number of competing models, and N is the number of datasets used in the experiment. The average ranks of the competing models based on MAPE values are given in Table 4.

Table 4. Representation of MAPE in rank form for various models

Model	AEP	COMED	DEOK	DUQ	EKPC	FE	NI	PJME	PJMW	Rj
ANN	6	7	7	5	7	6	7	4	4	5.89
LSTM-S2S	5	2	6	3	4	2	5	5	3	3.89
GRU-GA	4	5	3	7	3	3	6	2	6	4.33
CNN-LSTM	2	3	4	2	6	5	3	3	2	3.33
GRU-CNN	3	4	5	4	5	7	2	6	5	4.56
PLCNet	4	6	2	6	2	4	4	7	7	4.67
P2D-CNN-LSTM	1	1	1	1	1	1	1	1	1	1

Since the Rj are not uniformly equal, we reject the null hypothesis. Subsequently, as per recommendations, we proceed with the Nemenyi test. The calculated CD at a significance level of 0.10 is 2.74. Figure 10 illustrates a CDD depicting the results of the statistical analysis.

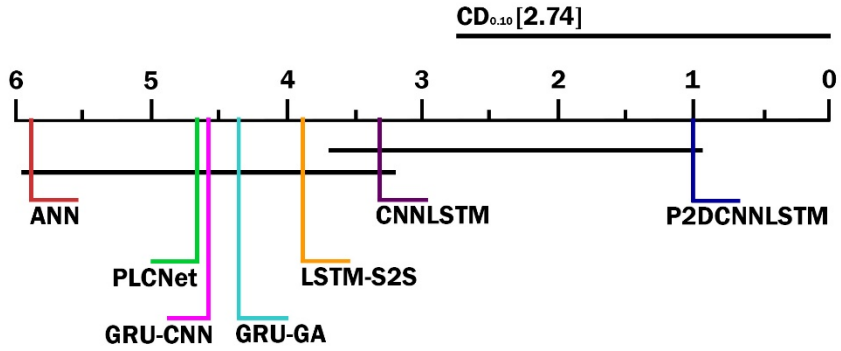

Fig 10. A Critical Distance Diagram

In this diagram, the average ranks of all models are plotted. The models that are not significantly different are connected by CD lines. Analyzing these CD lines, we may conclude that the P2DCNNLSTM model is significantly different and better than the ANN, LSTM-S2S, GRU-GA, GRU-CNN, and PLCNet models. However, it is at a critical distance to the CNN-LSTM model.

4 Conclusion

This study presents a novel, multi-step STLF method that integrates a hybrid approach, combining GAF and CNN-LSTM networks. The proposed methodology leverages context vectors generated from the LSTM and 2D-CNN networks to improve the STLF accuracy. To elaborate on their roles, within this framework, the 2D-CNN plays a crucial role in extracting context vectors from the GAF-converted input, and the LSTM focuses on capturing temporal dependencies in the sequence input. By combining the outputs of both, the proposed model harness the comprehensive representation of spatial and temporal aspects, enriching the overall context vector. This fusion enhances the model's ability to discern complex patterns and relationships in the load sequence, contributing to improved performance in forecasting. This study conducts an extensive comparative analysis across nine datasets, assessing the performance of the proposed method against existing STLF approaches, including state-of-the-art models. Across AEP, COMED, DEOK, DUQ, EKPC, FE, NI, PJME, and PJMW, the proposed model achieves MAPE values of 3.34, 5.67, 4.92, 4.84, 5.24, 4.73, 5.15, 5.49, and 3.96, respectively. Additionally, the CDD aligns with the conclusions drawn from the comparative experiment, further supporting the findings. In future research, we can explore the integration of additional exogenous variables to improve forecasting performance. Another research direction involves incorporating the proposed framework into mid-term and long-term forecasting studies.

References

1) Eren Y, Küçükdemiral İ. A comprehensive review on deep learning approaches for short-term load forecasting. *Renewable and Sustainable Energy Reviews*. 2024;189:114031–114031. Available from: https://doi.org/10.1016/j.rser.2023.114031.
2) Ullah I, Hasanat SM, Aurangzeb K, Alhussein M, Rizwan M, Anwar MS. Multi-horizon short-term load forecasting using hybrid of LSTM and modified split convolution. *PeerJ Computer Science*. 2023;9:e1487–e1487. Available from: https://doi.org/10.7717/peerj-cs.1487.
3) Chapagain K, Gurung S, Kulthanavit P, Kittipiyakul S. Short-Term Electricity Demand Forecasting Using Deep Neural Networks: An Analysis for Thai Data. *Applied System Innovation*. 2023;6(6):100–100. Available from: https://doi.org/10.3390/asi6060100.
4) Kiranyaz S, Onuravci O, Abdeljaber T, Ince, Moncefgabbouj DJ, Inman. 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing* . 2021;151:107398–107398. Available from: https://doi.org/10.1016/j.ymssp.2020.107398.
5) Pirbazari AM, Sharma E, Chakravorty A, Elmenreich W, Rong C. An Ensemble Approach for Multi-Step Ahead Energy Forecasting of Household Communities. *IEEE Access*. 2021;9:36218–36240. Available from: https://doi.org/10.1109/ACCESS.2021.3063066.
6) Abumohsen M, Owda AY, Owda MY. Electrical Load Forecasting Using LSTM, GRU, and RNN Algorithms. *Energies*. 2023;16(5):2283–2283. Available from: https://doi.org/10.3390/en16052283.
7) Agga FA, Abbou SA, Houm YE, Labbadi M. Short-Term Load Forecasting Based on CNN and LSTM Deep Neural Networks. *IFAC-PapersOnLine*. 2022;55(12):777–781. Available from: https://hal.science/hal-03809816/.
8) Sajjad M, Khan ZA, Ullah A, Hussain T, Ullah W, Lee MY, et al. A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting. *IEEE Access*. 2020;8:143759–143768. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9141253.
9) Li C, Hu R, Hsu CY, Han Y, Hua H, Liu M, et al. An ensemble framework for short-term load forecasting based on parallel CNN and GRU with improved ResNet. Electric Power Systems Research. *2022 IEEE 4th International Conference on Power, Intelligent Computing and Systems (ICPICS)*. 2023;10:109057–109057. Available from: https://ieeexplore.ieee.org/document/9873566.
10) Hua H, Liu M, Li Y, Deng S, Wang Q. An ensemble framework for short-term load forecasting based on parallel CNN and GRU with improved ResNet. *Electric Power Systems Research*. 2023;216:109057–109057. Available from: https://doi.org/10.1016/j.epsr.2022.109057.
11) Shilpa GN, Sheshadri GS. ANN Based Short Term Load Forecasting for Karnataka State Demand. *2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*;p. 1–5. Available from: https://www.ijerd.com/paper/vol13-issue7/J1377579.pdf.
12) Pirbazari AM, Sharma E, Chakravorty A, Elmenreich W, Rong C. An Ensemble Approach for Multi-Step Ahead Energy Forecasting of Household Communities. *IEEE Access*. 2021;9:36218–36240. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9366871.
13) Inteha A, Nahid-Al-Masood. A GRU-GA Hybrid Model Based Technique for Short Term Electrical Load Forecasting. *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. 2021;p. 515–519. Available from: https://doi.org/10.109/ICREST51555.2021.9331156.
14) Rafi SH, Nahid-Al-Masood, Deeba SR, Hossain E. A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network. *IEEE Access*. 2021;9:32436–32448. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9358156.
15) Wu L, Kong C, Hao X, Chen W. A Short-Term Load Forecasting Method Based on GRU-CNN Hybrid Neural Network Model. *Mathematical Problems in Engineering*. 2020;2020:1–10. Available from: https://doi.org/10.1155/2020/1428104.
16) Farsi B, Amayri M, Bouguila N, Eicker U. On Short-Term Load Forecasting Using Machine Learning Techniques and a Novel Parallel Deep LSTM-CNN Approach. *IEEE Access*. 2021;9:31191–31212. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9356582.
17) Estebsari A, Rajabi R. Single Residential Load Forecasting Using Deep Learning and Image Encoding Techniques. *Electronics*. 2020;9(1):68–68. Available from: https://www.mdpi.com/2079-9292/9/1/68.
18) Bhatt D, Patel C, Talsania H, Patel J, Vaghela R, Pandya S, et al. CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics*. 1920;10(20):2470–2470. Available from: https://doi.org/10.3390/electronics10202470.