

Article

# Adaptive Bi-Directional LSTM Short-Term Load Forecasting with Improved Attention Mechanisms

Kun Yu

College of Electrical Engineering New Energy, China Three Gorges University, Yichang 443002, China; kun@ctgu.edu.cn

**Abstract:** Special load customers such as electric vehicles are emerging in modern power systems. They lead to a higher penetration of special load patterns, raising difficulty for short-term load forecasting (STLF). We propose a hierarchical STLF framework to improve load forecasting accuracy. An improved adaptive K-means clustering algorithm is designed for load pattern recognition and avoiding local sub-optimal clustering centroids. We also design bi-directional long-short-term memory neural networks with an attention mechanism to filter important load information and perform load forecasting for each recognized load pattern. The numerical results on the public load dataset show that our proposed method effectively forecasts the residential load with a high accuracy.

**Keywords:** load forecast; clustering; bi-directional LSTM; attention mechanism; pattern recognition

## 1. Introduction

### 1.1. Background

Load forecasting is an essential component of power system energy management and operation, with benefits to system reliability and security. With the fast development of renewable energy sources in distribution power systems, such as wind and photovoltaics (PVs), local power load requires more accurate forecasting with less controllable power supply sources. Furthermore, the increasing penetration of electric vehicles (EVs) and the participation of demand response (DR) programs make it more challenging to forecast residential load. Therefore, accurate load forecasting ensures suitable economic dispatch decisions from system operators and reduces system operation costs.

Accurate load forecasting relies on historical load data to reveal practical electricity consumption patterns for load forecasting [1,2]. Traditional load forecasting methods include time series analyzing method and regression methods, which basically analyze current load states and extend them to the future. The autoregressive integrated moving average (ARIMA) model is designed to perform short-term load forecasting embedded with a lifting scheme. Both the seasonal autoregressive integrated moving average (SARIMA) model and artificial neural network (ANN) are utilized and compared in the real-time Turkish Market [3]. Although these methods only require few load data for analysis, they cannot reveal the real relationship between the historical and forecasted load data due to their simple mapping principles. Furthermore, their learning strategy may not suit complex load behaviors with patterns from EV and DR programs.

### 1.2. Literature Review

Machine-learning algorithms effectively forecast short-term load and reveal load behavior patterns with a huge amount of historical data. Measurement systems in distribution power systems provide adequate terminal load measurements with smart meter for load forecasting, making it possible to analyze customer electricity consumption patterns. The support vector machine (SVM) method is utilized to analyze and normalize critical factors to forecast short-term load [4]. A classification and regression tree data mining method is



**Citation:** Yu, K. Adaptive Bi-Directional LSTM Short-Term Load Forecasting with Improved Attention Mechanisms. *Energies* **2024**, *17*, 3709. <https://doi.org/10.3390/en17153709>

Academic Editors: J. Carlos García-Díaz and Óscar Trull

Received: 28 May 2024  
Revised: 17 July 2024  
Accepted: 23 July 2024  
Published: 27 July 2024



**Copyright:** © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

proposed to extract characteristic attributes in the frequency domain from various load profiles and is utilized for customer load classification [5]. In [6], the authors proposed an improved multi-task learning algorithm for a Bayesian spatiotemporal Gaussian process model (BSGP) to predict load. However, these methods usually face the difficulty of poor generalization performances.

More research focuses on combining deep learning methods with feature selection to deal with special load behavior patterns [7–9]. A deep learning method combining a recurrent neural network (RNN) with an input attention mechanism and hidden connection mechanism is utilized for a higher accuracy in short-term load forecasting [10]. Deep forest regression is utilized for STLF, combining a multi-grained scanning procedure and cascade forest procedure [11]. In [12], long-short-term memory (LSTM) is introduced into multiple load prediction for multi-task learning, with a single load feature and auxiliary coupling information learning. A convolutional neural network (CNN) and LSTM neural network are combined for load prediction in [13]. A hybrid neural network composed of CNN and bidirectional gated recurrent units is designed for short-term load forecasting with immersing DR programs [14]. In [15], a Markov-chain mixture distribution model (MCM) is utilized for very short-term load forecasting and is used to forecast one step ahead half-hour resolution residential load. A deep belief network and bidirectional RNN deep learning method with a bisectin K-means algorithm is proposed to achieve unsupervised pre-training and supervised adjustment training [16]. These methods usually face the difficulty of over-fitting or still require more accurate preprocessing methods for load pattern analysis [17,18].

Clustering plays an important role in the preprocessing stage of deep-learning load forecasting methods. They enhance load forecasting accuracy by grouping similar load behaviors and providing categorizing conditions, leading to forecasting each load pattern separately. Various clustering algorithms are utilized in load forecasting to improve accuracy, such as K-means, C-means, and expectation maximization methods [19–21]. Clustering algorithms also benefit the performance of deep-learning load forecasting algorithms, such as graph neural networks [22,23]. In [24], an optimal kernel function selection method is proposed for load forecasting, combining SVR and weighted votes. Although much research focuses on introducing clustering algorithms into deep-learning load forecast methods, the load behaviors of EV and DR programs with higher penetration require accurate recognition and forecasting.

### 1.3. Contributions

This work aims to forecast short-term load with a high penetration of EV and DR programs. We focus on designing a load recognition and forecasting framework to achieve special load pattern recognition and simplify the forecasting process. The contributions of this work are summarized as follows:

- (1) We propose a hierarchical short-term load forecast (STLF) framework to accurately forecast system load with a high penetration of EV and DR programs. The framework combines clustering and deep-learning methods to recognize load behavior patterns and improve forecasting accuracy.
- (2) We design an improved adaptive K-means clustering algorithm for load pattern recognition. By incorporating exploration probability into the centroid results, the proposed adaptive K-means clustering algorithm avoids local optimal searching. We utilize multiple indexes (mean square error, Davies–Bouldin, and separation index) to evaluate the algorithm performance.
- (3) We design bi-directional LSTM neural networks with an attention mechanism to forecast each recognized load pattern. The designed bi-directional LSTM neural network effectively utilizes the observed factors and captures long-term temporal characteristics.

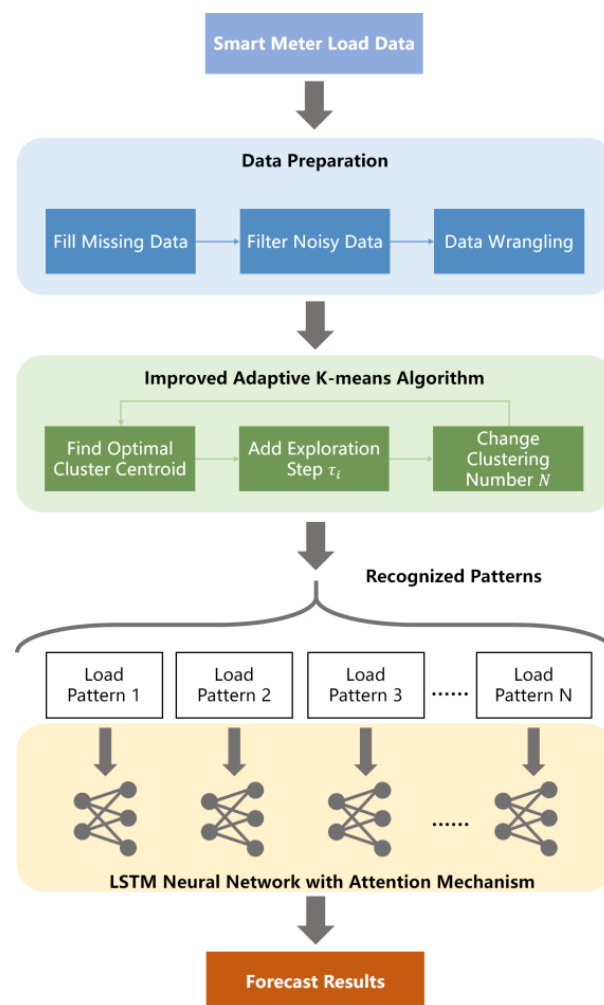
The remainder of this paper is organized as follows. In Section 2, we formulate the improved adaptive K-means clustering algorithm for load pattern recognition. We also design bi-directional LSTM neural networks with an attention mechanism for load

forecasting. Section 3 provides different case studies. Finally, Section 4 summarizes the conclusions and future research directions.

## 2. Hierarchical STLF Framework

### 2.1. Framework Structure

The proposed hierarchical STLF framework is shown in Figure 1. Data preparation fills missing load data and filters noisy data, which are further wrangled into daily load curves for the convenience of clustering and forecasting. An improved adaptive K-means algorithm clusters load data into several groups by adding an exploration step and iteratively changing the clustering number. The recognized load patterns are further forecasted separately using the bi-directional LSTM neural network with an attention mechanism.



**Figure 1.** The hierarchical STLF framework structure.

### 2.2. Improved Adaptive K-Means Algorithm

The traditional K-means clustering algorithm aims to cluster data into groups in an unsupervised manner. Each data belongs to the group with the closest clustering centroid. We design the improved adaptive K-means clustering algorithm as shown in Algorithm 1. We randomly select data points as the initial clustering centroids, which are at first given the clustering number  $k_t$ . The whole dataset is divided into  $k_t$  groups based on the distance, and we calculate the new clustering centroid as the average value of all data points in each group.

**Algorithm 1** Improved Adaptive K-means Clustering Algorithm**Input:** Load vector set P**Output:** Clustering group G

- 1: Initialize  $t = 1$ ;
- 2: While  $MSE_t \leq \chi MSE_{t-1}$ , do
- 3: Initialize clustering number  $k_t$  and clustering centroid  $g$ ;
- 4: Adjust  $\bar{\epsilon}, \underline{\epsilon}, \mu, p$ ;
- 5: Calculate  $MSE_t$  and update clustering centroid  $\tilde{g}$ ;
- 6: If  $Tol \leq Err_{tol}$ , then
- 7: Adjust  $g$ ;
- 8: End if
- 9:  $t = t + 1$ ;
- 10:  $k_t = k_{t-1} + 1$ ;
- 11: End while

When giving a load vector set and  $P = (p_1, p_2, p_3, \dots, p_n)$  with  $m$  dimension and  $n$  data, the K-means algorithm divides it into  $k$  groups  $G = (G_1, G_2, G_3, \dots, G_k)$  by minimizing the square sum of distances between each data  $p_i$  and clustering centroid  $g_j$  of group  $G_j$ :

$$\min_G \sum_{j=1}^k \sum_{i=1}^n \|p_i - g_j\|^2. \quad (1)$$

Each clustering centroid  $g_j$  is optimized and used to calculate distances, where Equation (1) can be turned into:

$$\min_G \sum_{j=1}^k \frac{1}{2\|g_j\|} \sum_{a \neq b \in G_j} (a - g_i)^T (g_i - b) \quad (2)$$

We introduce exploration probability into the optimized results to avoid easily converging toward local optimal results. The traditional K-means clustering algorithm is sensitive to the initial clustering centroid, and the special type of load with frequent variations leads to large noises and outliers. These abnormal values may reduce the effectiveness of the traditional K-means algorithm. A random exploration vector is added to the clustering centroid:

$$\tilde{g}_j = g_j + \epsilon_j. \quad (3)$$

$\epsilon_j$  is the random exploration vector, defined as:

$$\epsilon_j \sim \mu U(\underline{\epsilon}_j, \bar{\epsilon}_j) \times (B_1(1, p) - B_2(1, p)), \quad (4)$$

where  $\mu$  is the exploration factor,  $U(\underline{\epsilon}_j, \bar{\epsilon}_j)$  is the uniform distribution function with upper bound  $\bar{\epsilon}_j$  and lower bound  $\underline{\epsilon}_j$ .  $B_1(1, p)$ , and  $B_2(1, p)$  are two independent 0–1 distribution functions with probability  $p$ .  $\epsilon_j$  is randomly chosen from the multiplied individual distribution functions with the exploration factor controlling the exploration regions.  $(B_1(1, p) - B_2(1, p))$  controls the exploration directions by choosing  $-1, 0, 1$ , whose value represents the backward direction ( $-1$ ), no direction ( $0$ ), and forward direction ( $1$ ), respectively. Hence, we guarantee that the exploration direction is evenly determined.

We introduce various evaluation indexes to assess the performance of the proposed algorithm. The mean square error ( $MSE$ ) index calculates the average value of the square estimation error:

$$MSE = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^k (p_i - g_j)^2. \quad (5)$$

The  $MSE$  index measures the diversity between data and clusters without considering the internal density of each cluster.

The Davies–Bouldin (*DB*) index calculate the clustering density:

$$DB = \frac{1}{k} \sum_{j=1}^k \max_{l \neq j, l \in [1, k]} \frac{s_j + s_l}{\|g_j - g_l\|}, \quad (6)$$

where  $s_j$  is the diversity of  $j$  cluster internal diversity:

$$s_j = \left( \frac{1}{|g_j|} \sum_{p_i \in G_j} \|p_i - g_j\|^2 \right)^{\frac{1}{2}}. \quad (7)$$

The *DB* index measures the internal clustering density, while the distances between different clusters are not considered.

The separation (*SP*) index calculates the average distance between cluster centroids:

$$SP = \frac{2}{k^2 - k} \sum_{i=1}^{k-1} \sum_{j=i+1}^k \|g_i - g_j\|. \quad (8)$$

The *SP* index only measures the diversity of different clusters. Here, we design the convergence tolerance of the inner-level iteration as:

$$Err_{tol} = \left| 1 - \frac{MSE_t}{MSE_{t-1}} \right| \quad (9)$$

where  $Err_{tol}$  is calculated using the *MSE* indexes of inner-level  $t - 1$ th and  $t$ th iterations. Other indexes are also utilized to evaluate clustering performances. We gradually narrow the probabilistic bound  $[\underline{\epsilon}_j, \bar{\epsilon}_j]$  after each inner loop iteration and decrease the exploration factor  $\mu$ . For each iteration,  $\underline{\epsilon}_j$ ,  $\bar{\epsilon}_j$ ,  $\mu$ , and  $p$  are adjusted as follows:

$$\underline{\epsilon}_{j,t} = \alpha \underline{\epsilon}_{j,t-1} + (1 - \alpha) \frac{\underline{\epsilon}_{j,t-1} + \bar{\epsilon}_{j,t-1}}{2}, \quad (10)$$

$$\bar{\epsilon}_{j,t} = \beta \bar{\epsilon}_{j,t-1} + (1 - \beta) \frac{\underline{\epsilon}_{j,t-1} + \bar{\epsilon}_{j,t-1}}{2}, \quad (11)$$

$$\mu_t = \gamma \mu_{t-1}, \quad (12)$$

$$p_t = \delta p_{t-1}. \quad (13)$$

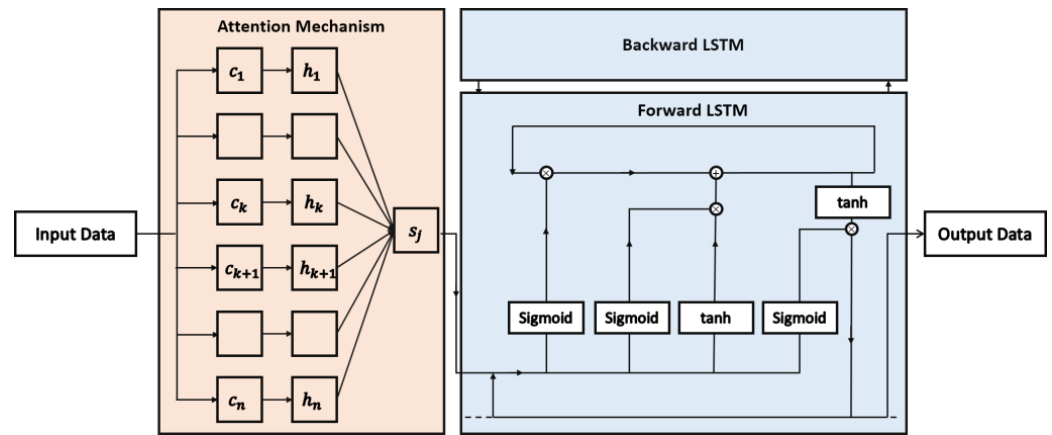
The clustering number  $k$  is adjusted during each outer-level iteration by assessing the relative changes of *MSE*. Other indexes, such as *DB* and *SP*, are compared to evaluate the clustering results' performance. The initialized  $k_0$  is 1 and plus 1 after each outer-level iteration:

$$k_t = k_{t-1} + 1. \quad (14)$$

Iteration stops when  $MSE_t \geq \chi MSE_{t-1}$ , as more clustering groups will not lead to a better performance for load forecasting. The improved adaptive K-means clustering algorithm provides output  $G$ .

### 2.3. Bi-Directional LSTM Neural Network with Attention Mechanism

We design the bi-directional LSTM neural network with an attention mechanism for the load forecasting of each recognized load pattern. Bi-directional LSTM neural networks with an attention mechanism analyze the input sequential load data of each load pattern and minimize its objective function to improve the load forecasting performance, as shown in Figure 2.



**Figure 2.** The structure of bi-directional LSTM with attention mechanism.

The LSTM neural network is designed based on recurrent neural networks (RNNs) and maintains a long-term memory for prediction. Each LSTM neuron has four inputs, where two inputs belong to the input gate and the other two inputs belong to the forget gate and output gate, respectively. Input gates update their states by combining the outputs  $h_{t-1}$  of the last neuron and the current observation  $x_t$  into:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i). \quad (15)$$

$W_i$  and  $b_i$  are the weights and offsets of the input gates. Forget gates update their states as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f). \quad (16)$$

$W_f$  and  $b_f$  are the weights and offsets of the forget gates. Output gates update their states as:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o). \quad (17)$$

$W_o$  and  $b_o$  are the weights and offsets of the output gates. Candidate state values are calculated as:

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c). \quad (18)$$

$W_c$  and  $b_c$  are the weights and offsets of the candidate state updates. Current state values are updated by old state values and candidate state values:

$$c_t = i_t \times \tilde{c}_t + f_t \times c_{t-1}. \quad (19)$$

Output values are determined by  $o_t$  and  $c_t$ :

$$h_t = o_t \tanh(c_t). \quad (20)$$

Single forward LSTM cannot extract information from the end to the start, which may cause difficulty for higher-resolution forecasting. We introduce the backward LSTM to extract the backward information from input features. Bi-directional LSTM can utilize future data information rather than only past information. We establish two LSTM neural networks, forward LSTM and backward LSTM, which share the same hidden states. The forward LSTM utilizes the input data to forecast the future data, and the backward LSTM forecasts the past data with the same inputs. It should be noted that  $h_t$  of bi-directional LSTM includes both forward  $h_t^f$  and backward  $h_t^b$ . The bi-directional LSTM further extracts more information from both directions. The hidden state outputs of the forward and backward layers are combined to forecast the outputs.

$$h_t = o_t \tanh(w_t^f c_t^f + w_t^b c_t^b). \quad (21)$$

We introduce attention mechanisms in the LSTM neural networks to filter and update important load information. The attention mechanism simulates the human thinking process by calculating attention weights in neural networks, representing the importance of each hidden state for outputs. Given the hidden states  $\{h_1, h_2, h_3, \dots, h_n\}$ , the attention mechanism calculates attention weights as:

$$\alpha_{jk} = \frac{\exp(e_{jk})}{\sum_{k=j}^n \exp e_{jk}}. \quad (22)$$

$e_{jk}$  represents the compatibility between the hidden states  $h_j$  and  $h_k$ :

$$e_j = v \tanh W_j [h_j, h_k] + b_j. \quad (23)$$

The weighted sum of the hidden states is calculated with attention weights:

$$s_j = \sum_{k=1}^n \alpha_{jk} h_k. \quad (24)$$

The final output value considers the weighted sum of the hidden states, current input values and last hidden states:

$$h_j = f(s_j, h_j, c_j), \quad (25)$$

where  $f(\cdot)$  is the LSTM function.

### 3. Case Study

#### 3.1. Data Preparation

We utilize the customer load data from the Ausgrid distribution network in Sydney and the NSW region, including 300 customers' load and PV data with 365 days [25]. We select 200 customers' data with 300 days and aggregate them into 5 aggregated customers, with a time resolution of 30 min. Each aggregated customer includes 35–45 m' data, with the consideration of avoiding overly large or small load values. We divide these data into the training set, validation set, and test set, with 150 days, 40 days, and 10 days, respectively.

#### 3.2. Clustering Results

We utilize the improved adaptive K-means clustering algorithm to recognize load patterns. For each aggregated customer, the proposed method adaptively recognizes the optimal clustering groups with a tolerance of 2.0%. The error index of each aggregated customer is shown in Table 1.  $Err_{tol}$  rapidly drops with increasing  $k$  and soon satisfies the tolerance of convergence.

**Table 1.**  $Err_{tol}$  of each aggregated customer.

Clustering Number $k$	C1	C2	C3	C4	C5
2	0.382	0.485	0.482	0.462	0.401
3	0.225	0.179	0.168	0.225	0.107
4	0.095	0.041	0.115	0.076	0.057
5	0.085	0.167	0.062	0.083	0.169
6	0.046	0.042	0.011	0.008	0.039
7	0.013	0.036			0.061
8		0.077			0.009
9		0.012			

The clustering number of each aggregated customer is 6, 8, 5, 5, 7, respectively. We calculate their  $DB$  and  $SP$  indexes with a clustering number ranging from 2 to  $k$ , as shown

as Tables 2 and 3. Nearly all *DB* indexes with the chosen clustering number *k* reach the lowest values, representing the closer internal density of each clustering group. The obtained clustering number *k* has a relatively larger *SP* index for each aggregated customer. The *SP* index results show that the clustering results contribute to improvements with larger *SP* results.

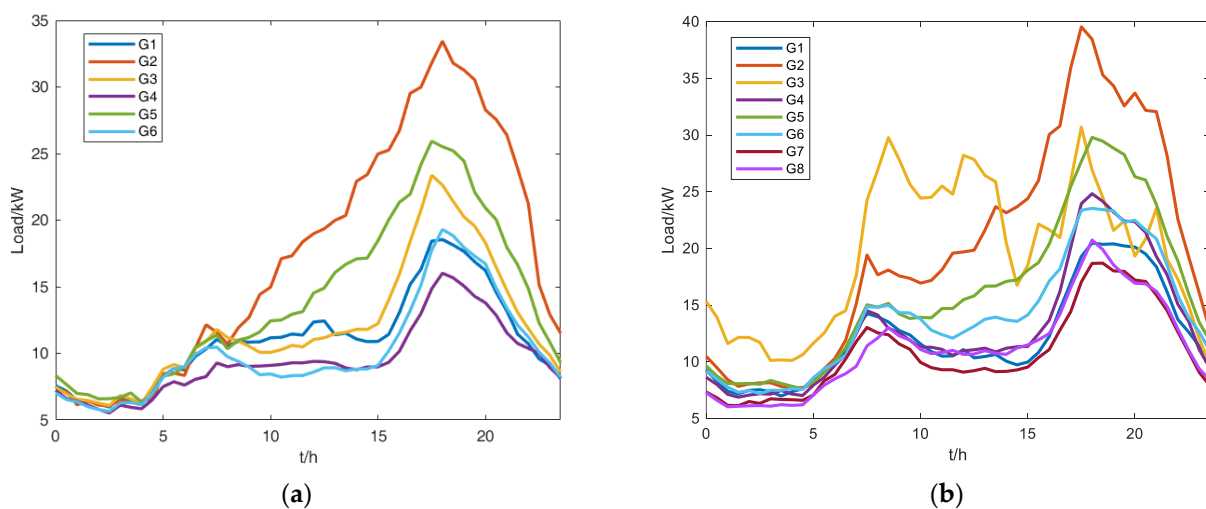
**Table 2.** *DB* index of each aggregated customer.

Clustering Number <i>k</i>	C1	C2	C3	C4	C5
2	0.0808	0.0454	0.0392	0.0558	0.0497
3	0.0767	0.0606	0.0396	0.0585	0.0679
4	0.0767	0.0514	0.0301	0.0540	0.0579
5	0.0696	0.0569	0.0390	0.0645	0.0465
6	0.0683	0.0543			0.0501
7		0.0417			0.0427
8		0.0344			

**Table 3.** *SP* index of each aggregated customer.

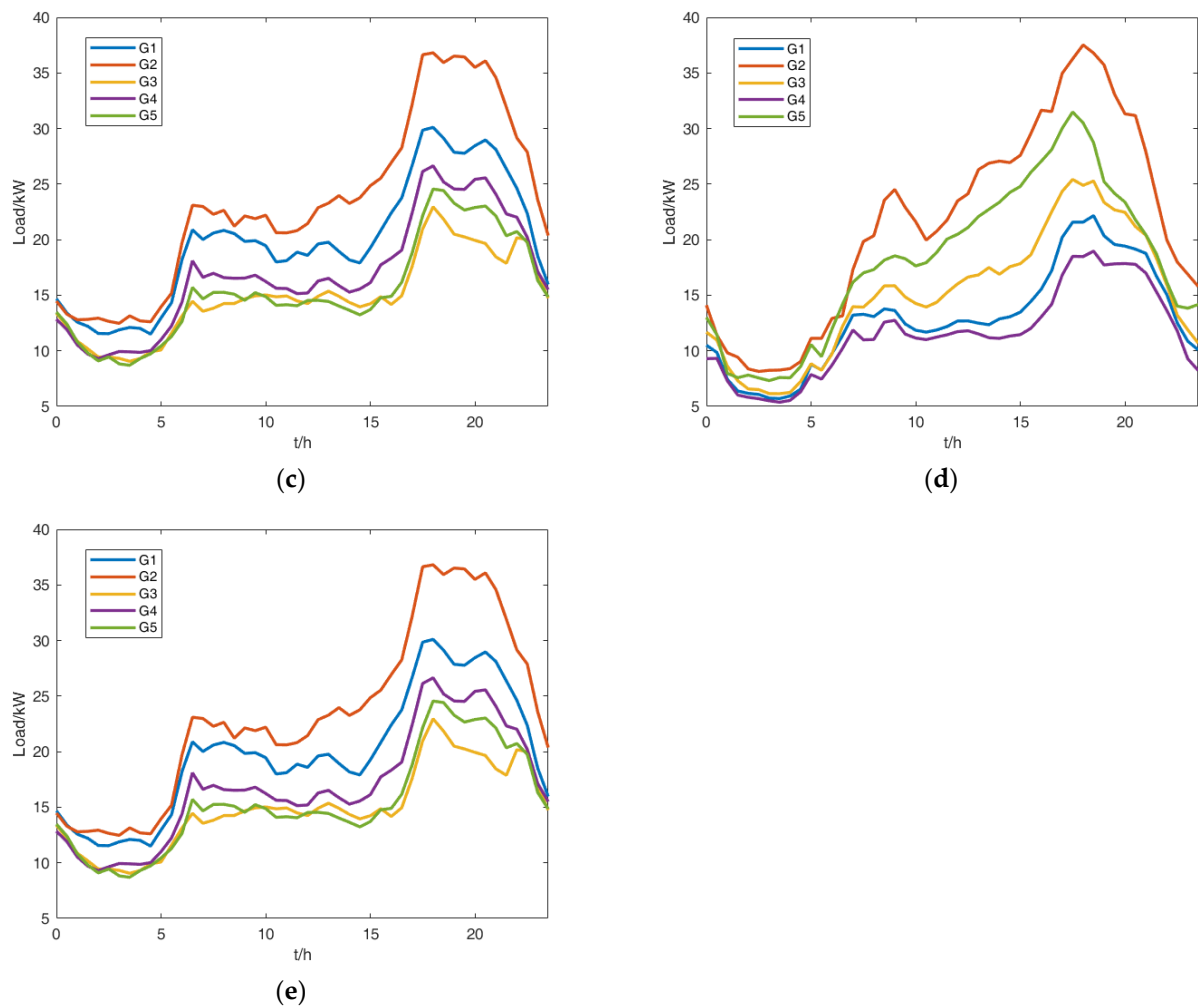
Clustering Number <i>k</i>	C1	C2	C3	C4	C5
2	28.13	46.94	36.80	37.25	30.95
3	35.50	38.55	35.66	39.08	25.76
4	30.61	41.28	43.09	38.55	27.51
5	33.05	38.78	37.65	34.06	44.61
6	31.48	32.96			25.04
7		30.63			31.48
8		40.17			

The clustering centroids of each aggregated customer with the improved adaptive K-means clustering algorithm using the training dataset are shown in Figure 3. Customer names and clustering group centroids are shown above each figure and shown in the legend. For example, “C1” represents the aggregated customer 1 with six clustering group centroids G1–G6. These clustering results have large differences in the late afternoon peak load and forenoon load. Meanwhile, the loads at mid-night and early morning in different clustering groups have similar values.



**Figure 3.** *Cont.*





**Figure 3.** (a) Clustering centroids  $G$  of aggregated customer C1; (b) Clustering centroids  $G$  of aggregated customer C2; (c) Clustering centroids  $G$  of C3; (d) Clustering centroids  $G$  of C4; (e) Clustering centroids  $G$  of C5.

The recognized patterns of each aggregated customer result in obvious differences in their load variations and are mainly divided into three types: large, medium, and small load variations. Large load variation patterns can be easily recognized in each aggregated customer, such as G2. Small load variation patterns usually have more differences in the afternoon and night. Some special load patterns are also recognized with the improved adaptive K-means algorithm, such as load pattern G3 with morning and afternoon peak loads. These load patterns help forecast their load variation trends.

### 3.3. STLF Results

We design the bi-directional LSTM neural network with an attention mechanism and utilize the 150-day data for training using Pytorch. The bi-directional LSTM neural network performs STLF for each type of recognized load patterns. We validate this method on the 40-day validation dataset and test its performance on the 10-day test dataset. All the training and tests are running on a computer with an Intel i5-9300 h CPU with 2.4 GHz and 16 GB memory.

We assess the validation and test results using root mean squared error (*RMSE*) and mean absolute percentage error (*MAPE*):

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}, \tag{26}$$

$$MAPE = \frac{100\%}{n} \times \sum_{i=1}^n \left| 1 - \frac{\hat{y}_i}{y_i} \right|. \tag{27}$$

Their results are shown in Figure 4. In the validation set, the proposed method achieves an *RMSE* ranging from 0.1020 to 0.1721 and an *MAPE* ranging from 0.7019% to 0.8035%, while it performs worse in C1, C2, and C5 in the test set compared with the results in the validation set. In C3 and C4, the proposed method achieves a better performance in the test set with *RMSE* of 0.1587 and 0.1207 and *MAPE* of 0.8021% and 0.7874%.

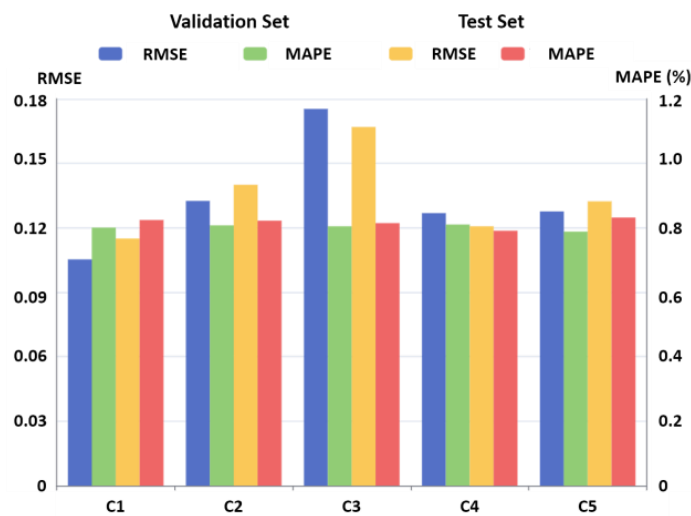


Figure 4. *RMSE* and *MAPE* results of validation set and test set.

We display the STL<sub>F</sub> results of both the validation set and the test set, as shown as Figure 5. The STL<sub>F</sub> results show that the proposed method achieves a good performance on both the validation set and the test set of each aggregated customer. We also find that the load forecasting errors during 11:00–14:00 and 19:00–21:00 are usually higher than other time periods, as shown as Figure 6. We notice that forecasted load errors only exceed 0.4 kW in several specific time points and that most forecasted load errors are smaller than 0.2 kW. The reason may rely on the forecasting errors arising from multiple load patterns during these time periods, leading to higher *RMSE*. However, the slightly forecasting errors during mid-night or early morning may result in higher *MAPE*, as the load at these time periods are relatively lower.

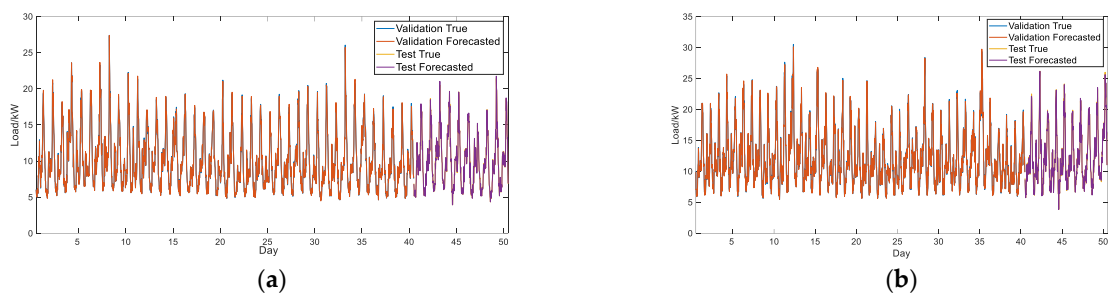
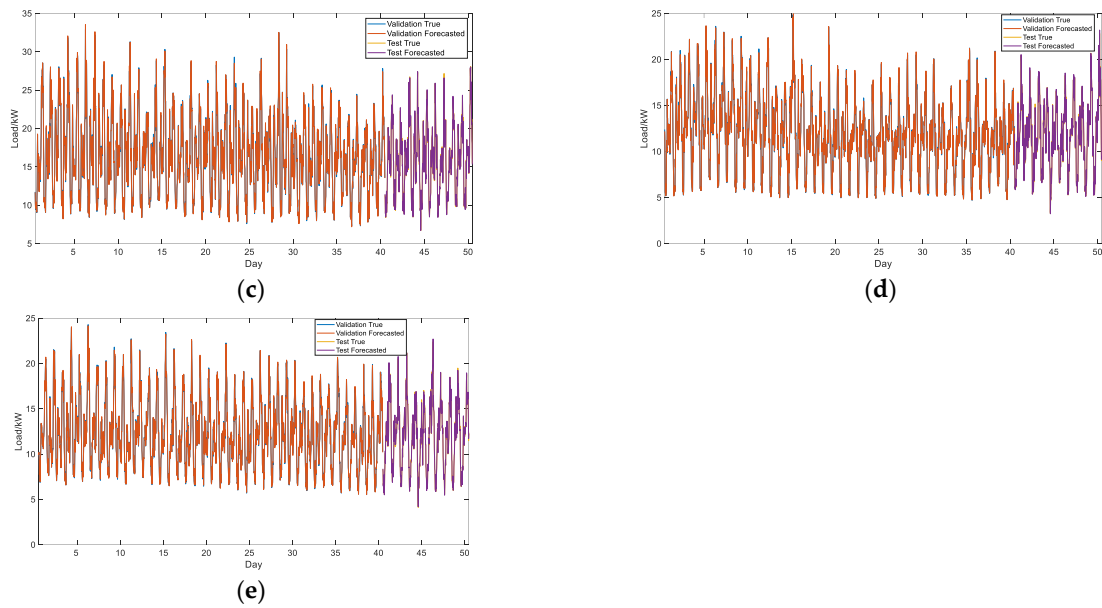
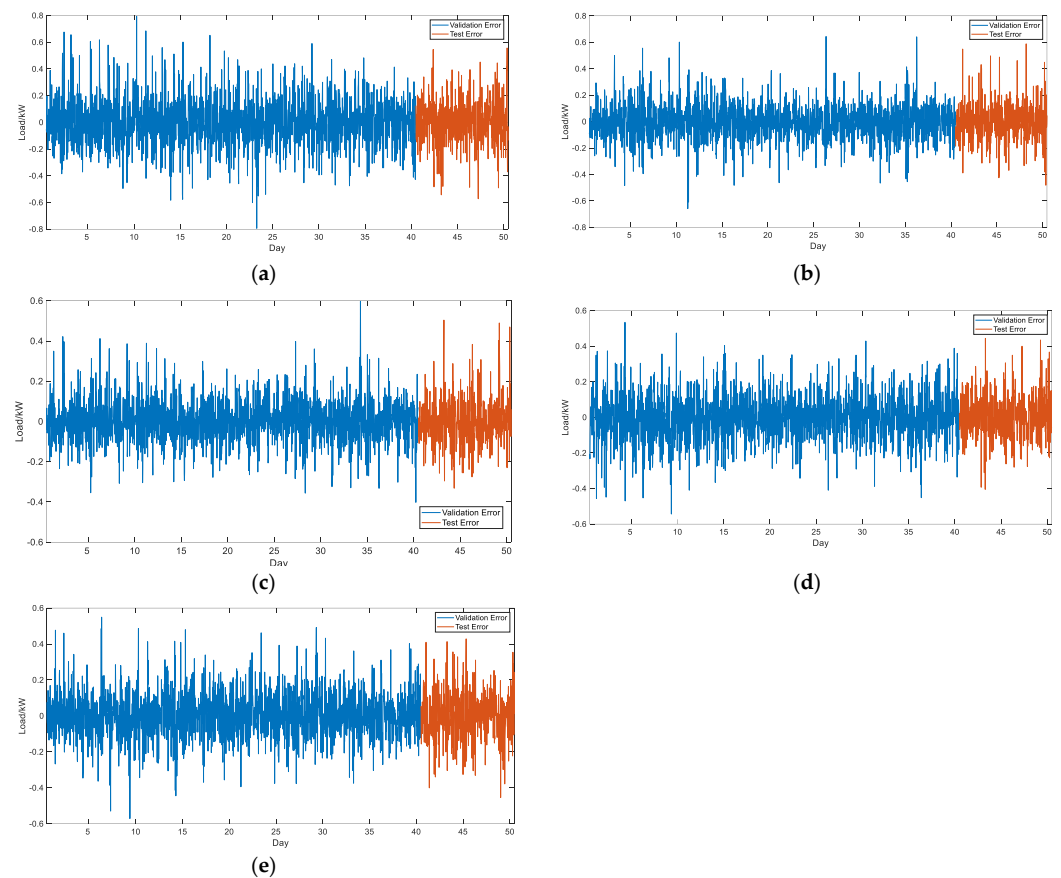


Figure 5. Cont.



**Figure 5.** (a) Forecasted and real load in the validation set and test set of C1; (b) Forecasted and real load in the validation set and test set of C2; (c) Forecasted and real load in the validation set and test set of C3; (d) Forecasted and real load in the validation set and test set of C4; (e) Forecasted and real load in the validation set and test set of C5.



**Figure 6.** (a) Forecasted load errors in the validation set and test set of C1; (b) Forecasted load errors in the validation set and test set of C2; (c) Forecasted load errors in the validation set and test set of C3; (d) Forecasted load errors in the validation set and test set of C4; (e) Forecasted and real load in the validation set and test set of C5.

#### 4. Conclusions

This work examines short-term load forecasting to reduce the forecasting errors of residential load with a higher penetration of EV and DR programs. We formulate the hierarchical STLTF framework with three parts: data preparation, improved adaptive K-means algorithm, and bi-directional LSTM neural network with an attention mechanism. We propose the improved adaptive K-means algorithm for load pattern recognition, where exploration steps are introduced to avoid local sub-optimal clustering centroids. The adaptive clustering number choosing scheme is also designed to choose the acceptable clustering number while maintaining fewer clustering groups to alleviate the difficulty of load forecasting. We apply the absolutely relative *MSE* changes as the tolerance for convergence and utilize *MSE*, *DB*, and *SP* indexes to evaluate the algorithm performance. Bi-directional LSTM neural networks with an attention mechanism are designed for forecasting loads for each load pattern. We introduce the attention mechanism to filter and update important load information and improve forecasting accuracy. The bi-directional LSTM also benefits information utilization with both past and future information extraction. We test the performance of the proposed hierarchical STLTF framework on actual load datasets. The results show that the proposed framework effectively recognizes load patterns and achieves *MAPE* around 0.75% in both validation and test sets. In future work, we would improve deep neural networks to reduce the training difficulty and improve accuracy. Traditional STLTF methods should be combined with machine-learning methods to achieve a better performance.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data sharing depends on acquisitions from readers.

**Conflicts of Interest:** The author declares no conflicts of interest.

#### References

1. Trull, O.; García-Díaz, J.C.; Troncoso, A. One-day-ahead electricity demand forecasting in holidays using discrete-interval moving seasonalities. *Energy* **2021**, *231*, 120966. [[CrossRef](#)]
2. Trull, O.; García-Díaz, J.C.; Troncoso, A. Application of discrete-interval moving seasonalities to spanish electricity demand forecasting during easter. *Energies* **2019**, *12*, 1083. [[CrossRef](#)]
3. Bozkurt Ö, Ö.; Biricik, G.; Tayşi, Z.C. Artificial neural network and SARIMA based models for power load forecasting in Turkish electricity market. *PLoS ONE* **2017**, *12*, e0175915. [[CrossRef](#)] [[PubMed](#)]
4. Dong, X.; Deng, S.; Wang, D. A short-term power load forecasting method based on k-means and SVM. *J. Ambient. Intell. Humaniz. Comput.* **2022**, *13*, 5253–5267. [[CrossRef](#)]
5. Zhong, S.; Tam, K.S. Hierarchical classification of load profiles based on their characteristic attributes in frequency domain. *IEEE Trans. Power Syst.* **2014**, *30*, 2434–2441. [[CrossRef](#)]
6. Gilanifar, M.; Wang, H.; Sriram, L.M.K.; Ozguven, E.E.; Arghandeh, R. Multitask Bayesian spatiotemporal Gaussian processes for short-term load forecasting. *IEEE Trans. Ind. Electron.* **2019**, *67*, 5132–5143. [[CrossRef](#)]
7. Li, C. Designing a short-term load forecasting model in the urban smart grid system. *Appl. Energy* **2020**, *266*, 114850. [[CrossRef](#)]
8. Sajjad, M.; Khan, Z.A.; Ullah, A.; Hussain, T.; Ullah, W.; Lee, M.Y.; Baik, S.W. A novel CNN-GRU-based hybrid approach for short-term residential load forecasting. *IEEE Access* **2020**, *8*, 143759–143768. [[CrossRef](#)]
9. Tang, X.; Dai, Y.; Wang, T.; Chen, Y. Short-term power load forecasting based on multi-layer bidirectional recurrent neural network. *IET Gener. Transm. Distrib.* **2019**, *13*, 3847–3854. [[CrossRef](#)]
10. Zhang, M.; Yu, Z.; Xu, Z. Short-term load forecasting using recurrent neural networks with input attention mechanism and hidden connection mechanism. *IEEE Access* **2020**, *8*, 186514–186529. [[CrossRef](#)]
11. Yin, L.; Sun, Z.; Gao, F.; Liu, H. Deep forest regression for short-term load forecasting of power systems. *IEEE Access* **2020**, *8*, 49090–49099. [[CrossRef](#)]
12. Sun, Q.; Wang, X.; Zhang, Y.; Zhang, F.; Zhang, P.; Gao, W. Multiple load prediction of integrated energy system based on long short-term memory and multi-task learning. *Autom. Electr. Power Syst.* **2021**, *45*, 63–70.
13. Rafi, S.H.; Deeba, S.R.; Hossain, E. A short-term load forecasting method using integrated CNN and LSTM network. *IEEE Access* **2021**, *9*, 32436–32448. [[CrossRef](#)]
14. Xuan, Y.; Si, W.; Zhu, J.; Sun, Z.; Zhao, J.; Xu, M.; Xu, S. Multi-model fusion short-term load forecasting based on random forest feature selection and hybrid neural network. *IEEE Access* **2021**, *9*, 69002–69009. [[CrossRef](#)]
15. Munkhammar, J.; van der Meer, D.; Widén, J. Very short term load forecasting of residential electricity consumption using the Markov-chain mixture distribution (MCM) model. *Appl. Energy* **2021**, *282*, 116180. [[CrossRef](#)]

16. Tang, X.; Dai, Y.; Liu, Q.; Dang, X.; Xu, J. Application of bidirectional recurrent neural network combined with deep belief network in short-term load forecasting. *IEEE Access* **2019**, *7*, 160660–160670. [[CrossRef](#)]
17. Aly, H.H.H. A proposed intelligent short-term load forecasting hybrid models of ANN, WNN and KF based on clustering techniques for smart grid. *Electr. Power Syst. Res.* **2020**, *182*, 106191. [[CrossRef](#)]
18. Atef, S.; Eltawil, A.B. Assessment of stacked unidirectional and bidirectional long short-term memory networks for electricity load forecasting. *Electr. Power Syst. Res.* **2020**, *187*, 106489. [[CrossRef](#)]
19. Bian, H.; Zhong, Y.; Sun, J.; Shi, F. Study on power consumption load forecast based on K-means clustering and FCM–BP model. *Energy Rep.* **2020**, *6*, 693–700. [[CrossRef](#)]
20. Sokhanvar, K.; Karimpour, A.; Pariz, N. Electricity price forecasting using a clustering approach. In Proceedings of the 2008 IEEE 2nd International Power and Energy Conference, Johor Bahru, Malaysia, 24–26 November 2008; pp. 1302–1305.
21. Martínez-Álvarez, F.; Troncoso, A.; Riquelme, J.C.; Riquelme, J.M. Partitioning-clustering techniques applied to the electricity price time series. In *Intelligent Data Engineering and Automated Learning-IDEAL 2007, Proceedings of the 8th International Conference, Birmingham, UK, 16–19 December 2007*; Proceedings 8; Springer: Berlin/Heidelberg, Germany, 2007; pp. 990–999.
22. Lin, W.; Wu, D.; Boulet, B. Spatial-temporal residential short-term load forecasting via graph neural networks. *IEEE Trans. Smart Grid* **2021**, *12*, 5373–5384. [[CrossRef](#)]
23. Han, F.; Pu, T.; Li, M.; Taylor, G. Short-term forecasting of individual residential load based on deep learning and K-means clustering. *CSEE J. Power Energy Syst.* **2020**, *7*, 261–269.
24. Che, J.X.; Wang, J.Z. Short-term load forecasting using a kernel-based support vector regression combination model. *Appl. Energy* **2014**, *132*, 602–609. [[CrossRef](#)]
25. Ratnam, E.L.; Weller, S.R.; Kellett, C.M.; Murray, A.T. Residential load and rooftop PV generation: An Australian distribution network dataset. *Int. J. Sustain. Energy* **2017**, *36*, 787–806. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.