ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

Short-term Power Load Forecasting Based on a New Efficient Deep Learning Framework

Liguang Wang[1,a,*]

[1]*State Grid Zhenjiang Power Supply Comphany, Jiangsu, China*
[a]*liguang.wang@foxmail.com*
[*]*Corresponding author*

Abstract: *This research implemented a short-term power load forecasting model using a new deep learning framework called MindSpore. The new framework this research uses is more efficient than traditional deep learning frameworks such as TensorFlow. Firstly, the data is processed to meet the specific requirements of MindSpore. Subsequently, this research constructs the network architecture, comprising the LSTM layer, dropout layer, and fully connected layer. The effects of different parameters on the performance of the model are discussed in detail. The experiments unequivocally demonstrate the efficacy of the model in short-term power load forecasting.*

Keywords: *Power Load Forecasting, LSTM, Deep Learning Framework*

1. Introduction

Short-term power load forecasting involves the prediction of electricity consumption demand within a power system over a short period, usually ranging from a few hours to a few days [1]. This prediction is based on historical load data, weather forecasts, holiday schedules, and other relevant factors. The significance of short-term power load forecasting becomes apparent in its capacity to aid power companies in strategically allocating generation capacity, optimizing transmission lines, and managing energy storage resources to effectively meet user demand while reducing operational expenses. Moreover, accurate short-term power load forecasting provides power market participants with the necessary information to make well-informed trading decisions, thereby increasing profitability.

With the development of economy and society, the power generation sector has experienced substantial transformations. Historically, the dominant method of power generation was thermal power generation, which primarily involved the combustion of coal to produce steam for driving generators. Anticipating power demand in advance proves invaluable for these facilities to strategically plan their generation plans, conserve labor resources, and preserve precious coal reserves, ultimately contributing to reduced production costs. Considering the increasing concerns surrounding global warming [2], the generation of new energy has emerged as a pivotal player in electricity production. However, new energy power generation, such as solar power generation and wind power generation, is susceptible to variations caused by weather conditions and other factors, which can have an impact on the stability of the power system. Consequently, to ensure the stability of power supply, it becomes essential to engage in short-term power load forecasting, pre-determine user power demands, and harmonize thermal and renewable energy generation.

In terms of temporal granularity, power load forecasting can be classified into three main categories: short-term power load forecasting, medium-term power load forecasting, and long-term power load forecasting [1]. Short-term power load forecasting typically operates at a daily level, considering various factors such as temperature, weather conditions, holidays, and other relevant variables to predict power load in the immediate future. Medium-term power load forecasting expands its scope to encompass a more extended time horizon, typically ranging from one to several months. The primary objective is to ascertain the power generation strategy for the forthcoming period. In contrast, long-term load forecasting deals with a much greater time horizon, often encompassing power load projections several years into the future. For the purposes of this paper, primary research focus is on short-term power load forecasting.

Common methods for short-term power load forecasting primarily consist of traditional forecasting techniques and deep learning-based approaches. Power load forecasting employing regression analysis

ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

stands as a prevalent method in this domain [3]. It relies on a statistical analysis of historical load data to construct a regression model that correlates load with various influencing factors, facilitating the prediction of future power load patterns. The regression-based power load forecasting approach is lauded for its simplicity, comprehensibility, computational efficiency, and ease of implementation. However, it's imperative to acknowledge that regression-based forecasting methods come with high data prerequisites, susceptibility to outliers, and vulnerability to external influences.

With the advancement of artificial intelligence technology, power load forecasting based on deep learning has become a focal point of research. Deep learning methodologies, including convolutional neural networks, recurrent neural networks, and self-attention mechanisms, have demonstrated remarkable capabilities in various domains such as image recognition [4], object detection [5], and natural language processing [6]. As the domain of new energy power generation undergoes vigorous development, power companies are placing increasingly stringent demands on the accuracy of load forecasting. In contrast to traditional methods, deep learning boasts enhanced capabilities for modeling complex nonlinear relationships, making it a prominent option for achieving precise load predictions. At present, power load data is becoming more and more abundant, which provides sufficient data support for the training of deep learning models.

The implementation of a deep learning model relies on the robust support of a deep learning framework. A deep learning framework is a specialized software, which can expedite the development process for engineers by facilitating swift model creation and training. Deep learning frameworks typically encompass essential functionalities, including such as the design of model architecture, processing of data, training of models, and evaluation of models. Due to the limitation of computing power, deep learning framework is not widely used in the beginning. At, present, with the advancement of technology, deep learning frameworks have garnered significant attention and experienced rapid development.

Presently, the mainstream deep learning frameworks include TensorFlow[7], PyTorch[8]. TensorFlow, an open-source deep learning framework developed by Google, offers extensive platform support, and enables distributed computing. In contrast, PyTorch, an open-source deep learning framework developed by Facebook, also provides compatibility with multiple platforms and is renowned for its dynamic graph capabilities. In the field of short-term power load forecasting, most existing models primarily rely on these two leading deep learning frameworks.

In recent years, as research in deep learning has deepened, some novel and advanced deep learning frameworks have been developed. One notable framework is MindSpore, developed by Huawei [9], with a primary mission to achieve three crucial goals of easy development, efficient execution, and full scenario coverage. MindSpore offers distinct advantages that differentiate it from traditional frameworks. For instance, automatic differentiation plays a crucial role in deep learning frameworks. Currently, mainstream deep learning frameworks adopt automatic differentiation technology based on either static graphs or dynamic graphs. Static graphs leverage static compilation techniques to enhance network performance, yet they can be intricate to construct and debug. Dynamic graphs are more convenient but can be challenging to maximize performance. MindSpore introduces a new approach of automatic differentiation based on source code transformations, which has the advantages of both traditional methods.

Nevertheless, short-term power load forecasting models based on deep learning mainly rely on the development of traditional deep learning frameworks such as TensorFlow. Very few power load forecasting models have been developed using the emerging deep learning frameworks, which prevents the model from taking advantage of the new deep learning framework and fails to gain performance advantages from the underlying layer. Hence, this research is dedicated to crafting a short-term power load forecasting model utilizing the MindSpore framework. By fully capitalizing on the strengths of MindSpore, this research aims to realize the task of short-term power load forecasting.

2. Data Processing and Model Architecture

This research will construct a neural network dedicated to short-term power load forecasting. Initial step of this research involves the acquisition of historical power load data over a defined period, followed by data processing to align with the requirements of the MindSpore deep learning framework. Additionally, the selection of an appropriate neural network architecture is crucial. This choice allows the model to effectively distill the intricate patterns within the extensive dataset, enhancing the capacity to make highly accurate load predictions. The comprehensive model structure is depicted in Figure 1.

ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

Subsequently, this section will delve into various aspects, including data processing and network architecture construction.

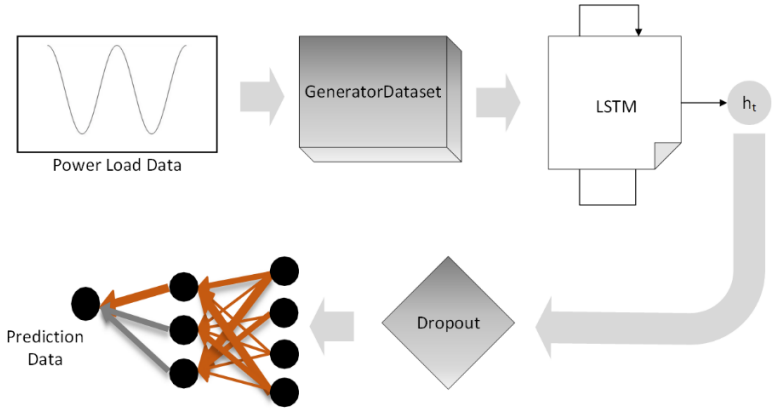

Figure 1: Model architecture

2.1. Data Processing

The deep learning model MindSpore needs to load data using a specified data structure for training and prediction. This research defines a class that encapsulates data using the $\_\_init\_\_$ function, reads data by index using the $\_\_getitem\_\_$ function, and returns sample totals using the $\_\_len\_\_$ function. Then this research uses the $GeneratorDataset$ provided by the MindSpore framework to load the data. Through the above processing, Mindspore can use the data that researchers provide. The whole data processing procedure is shown in the Figure 2.

Once the initial data processing is accomplished, there remains another crucial step: sample definition. In the original power load dataset, each data entry includes timestamp, temperature, power load, and other environmental factors. In this study, it is assumed that at the current moment denoted as i, this research utilizes the power load data from the preceding x moments as the input to predict the power load at time i. Consequently, for the training set, the input of each sample is defined as the power load data from time $i-x$ to time $i-1$, while the label corresponds to the power load at the current time i. Adhering to these defined rules, the data processing is finished.

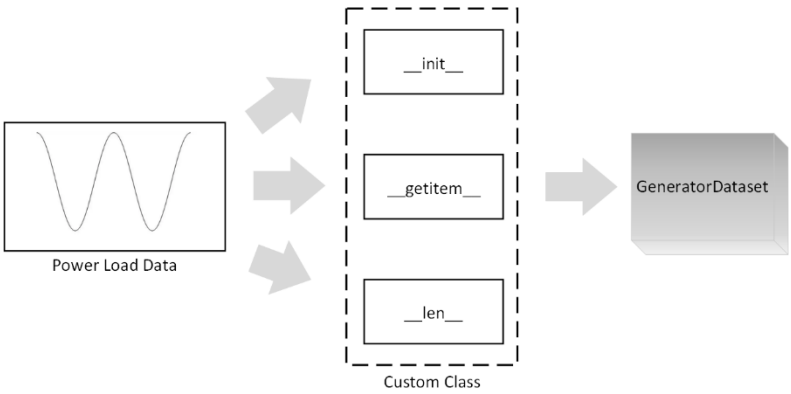

Figure 2: Data processing

2.2. LSTM Layer

In the process of power load forecasting, it is necessary to consider not only the current state, but also the impact of some previous moments. Simultaneously, the power load data at the current moment will also impact the subsequent moments. Therefore, when building neural networks, these factors must be considered. Long Short-Term Memory Network (LSTM) stands out as a type of recurrent neural network [10] that can handle long sequence data and solve the problem of gradient disappearance and gradient explosion. LSTM is well-suited for processing time series data, enabling the capture of long-term dependencies within the sequence, and can be used to predict future values. In this study, the LSTM layer is employed to extract the distinctive features from the power load data.

ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

The Cell state of the LSTM layer at time t is calculated using the gating mechanism. The formula is shown in formula (1), where $\otimes$ symbol represents the unit multiplication operation, $\boldsymbol{f_t}$ is called the forgetting gate, $\boldsymbol{i_t}$ is called the input gate, $\boldsymbol{c_{(t-1)}}$ indicates the cell state at time $\boldsymbol{t-1}$, $\boldsymbol{\tilde{c}_t}$ indicates the cell state update value. The calculation process of $\boldsymbol{f_t}$ is shown in formula (2), where $\boldsymbol{x_t}$ specifies the input at time t, $\boldsymbol{h_{(t-1)}}$ represents the hidden state at time $t-1$, $\boldsymbol{b_f}$ represents the bias, and w_{fhx} represents the weight of the neural network. The calculation process of the input gate $\boldsymbol{i_t}$ is shown in formula (3), where $\boldsymbol{x_t}$ specifies the input at time t, $\boldsymbol{h_{(t-1)}}$ represents the hidden state at time $t-1$, $\boldsymbol{b_i}$ represents the bias, and w_{ihx} represents the weight of the neural network. The cell state update value $\boldsymbol{\tilde{c}_t}$ is calculated by the formula (4), where $\boldsymbol{x_t}$ specifies the input at time t, $\boldsymbol{h_{(t-1)}}$ represents the hidden state at time $t-1$, $\boldsymbol{b_c}$ represents the bias, and w_{chx} represents the weight of the neural network. The hidden state $\boldsymbol{h_t}$ is calculated by the formula (5), where $\boldsymbol{o_t}$ is the output gate and $\boldsymbol{c_t}$ is the cell state at time t. The output gate $\boldsymbol{o_t}$ is calculated by the formula (6), where $\boldsymbol{x_t}$ specifies the input at time t, $\boldsymbol{h_{(t-1)}}$ represents the hidden state at time $t-1$, $\boldsymbol{b_o}$ represents the bias, and w_{ohx} represents the weight of the neural network.

In general, the LSTM layer uses the forgetting gate $\boldsymbol{f_t}$ to determine whether to retain or partially discard the information learned by $\boldsymbol{h_{(t-1)}}$. Output gate $\boldsymbol{o_t}$ determines which information is output. Finally, the current Cell status $\boldsymbol{c_t}$ and hidden status $\boldsymbol{h_t}$ are calculated using the forgetting gate, input gate and output gate.

$$\boldsymbol{c_t = f_t \otimes c_{(t-1)} + i_t \otimes \tilde{c}_t} \tag{1}$$

$$\boldsymbol{f_t = \sigma\left(w_{fhx} \cdot \left[h_{(t-1)}, x_t\right] + b_f\right)} \tag{2}$$

$$\boldsymbol{i_t = \sigma\left(w_{ihx} \cdot \left[h_{(t-1)}, x_t\right] + b_i\right)} \tag{3}$$

$$\boldsymbol{\tilde{c}_t = tanh\left(w_{chx} \cdot \left[h_{(t-1)}, x_t\right] + b_c\right)} \tag{4}$$

$$\boldsymbol{h_t = o_t * tanh(c_t)} \tag{5}$$

$$\boldsymbol{o_t = \sigma\left(w_{ohx} \cdot \left[h_{(t-1)}, x_t\right] + b_o\right)} \tag{6}$$

By using MindSpore, this research defines an LSTM layer directly using the $mindspore.nn.LSTM$ interface, and then set the number of hidden layer nodes to achieve the effect of LSTM.

2.3. Dropout Layer

Dropout serves as a valuable technique for combating overfitting. It effectively reduces overfitting by preventing correlations among neural nodes. More precisely, dropout combats overfitting by randomly setting a fixed percentage of hidden layer nodes to zero during each training batch. Within the MindSpore framework, this research directly establishes a dropout layer using the $mindspore.nn.dropout$ interface and specifies the drop rate to achieve the effect of dropout.

2.4. Fully Connected Layer

After the LSTM layer finishes extracting features from the power load data, the next step is to generate power load predictions using the defined model. To achieve this, this research introduces a fully connected layer responsible for performing power load predictions. By using MindSpore, this research utilizes the $mindspore.nn.dense$ interface to directly define the fully connected layer, where the number of nodes in the hidden layer is set to the same as that in the LSTM layer, and the number of nodes in the output layer is set to 1, indicating that there is only one output, which is the power load prediction result.

2.5. Loss Function

To quantify the disparity between the model-predicted power load data and the actual power load data

ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

at a specific moment, it is imperative to establish a loss function. In this study, the mean square error (MSE) is employed as the measure of loss, computed using the formula (7). Here, N represents the total number of samples, $\hat{y}_t$ signifies the predicted value at time t, y_t denotes the label value at time t, and t corresponds to the time for the current sample. In essence, the mean square error quantifies loss by squaring the differences between predicted and actual values, summing these squared values across all samples, and subsequently computing the average. Utilizing squares ensures that the mean square error calculation remains insensitive to sign variations. By using MindSpore, this research directly defines the mean square error loss function using the $mindspore.nn.MSELoss$ interface.

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_t - y_t)^2 \tag{7}$$

3. Experiments

3.1. Dataset

Since real power load data is considered proprietary information for power companies and difficult to obtain, this study utilizes simulated power load data instead. This research has chosen a subset of data for illustration purposes, as depicted in the Table 1. Within this dataset, each row corresponds to power load data at a specific moment, accompanied by various environmental factors. These factors encompass temperature, humidity, wind speed, general diffusion flow, and diffusion flow. In this dataset, for every data row, the first column denotes the timestamp, the subsequent five columns represent environmental factors, and the seventh column signifies the electrical load.

Table 1: Example of power load data

Date and Time	Temperature	Humidity	Wind Speed	General Diffuse Flow	Diffuse Flow	Total Electricity
1/1/2017 0:00	6.559	73.8	0.083	0.051	0.119	70425.54
1/1/2017 0:10	6.414	74.5	0.083	0.07	0.085	69320.84
1/1/2017 0:20	6.313	74.5	0.08	0.062	0.1	67803.22
1/1/2017 0:30	6.121	75	0.083	0.091	0.096	65489.23
1/1/2017 0:40	5.921	75.7	0.081	0.048	0.085	63650.45
1/1/2017 0:50	5.853	76.9	0.081	0.059	0.108	62171.34
1/1/2017 1:00	5.641	77.7	0.08	0.048	0.096	60937.36
1/1/2017 1:10	5.496	78.2	0.085	0.055	0.093	59566.75

3.2. Design of Experiments

In line with the methodology outlined in the second section of this paper, the MindSpore deep learning framework is employed to construct a short-term power load forecasting model. This model encompasses an LSTM layer, a dropout layer, and a fully connected layer. To optimize the performance of the model, a series of experiments were conducted to determine the optimal model parameters and evaluate the model's ability to predict power load.

This research centers on configuring the parameters for the LSTM layer firstly. The number of hidden layer nodes significantly impacts the feature extraction performance of LSTM. Thus, this research conducts a grid search in {32, 64, 128, 256, 512}. For the dropout layer, the drop rate affects the intervention degree of the model against overfitting. To this end, this research performs a grid search across the values {0.1, 0.2, 0.4, 0.6, 0.8}. Concurrently, other model parameters remain fixed throughout the grid search process. The batch size is set to 16, the learning rate is set to 0.001, and the number of training rounds is set to 30.

Upon determining the optimal model parameters, this research proceed to evaluate the performance of the model on the test dataset. Visual representations are created to illustrate the discrepancies between predicted results and actual values.

3.3. Evaluation Metric

To measure the prediction performance of the model, this research employs the Root Mean Square

ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

Error (RMSE) as evaluation metric. RMSE, widely acknowledged as a robust evaluation criterion [11] and referred to as the standard error, quantifies the disparity between predicted values and actual values. It is very sensitive to large or small errors in a set of measured values, so it can well reflect the precision of the measurement. RMSE is defined by the Formula (8), where N signifies the total number of samples, p_t represents the predicted value for sample t, and l_t denotes the label value for sample t.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_t - l_t)^2} \tag{8}$$

3.4. Results

In Table 2, the first column represents the drop rate of dropout layer, the second column represents the number of hidden nodes in the LSTM layer, the third column represents the RMSE on the test set, the fourth column represents the time to complete training, and the fifth column represents the average of RMSE on the test set with the same drop rate but different hidden nodes in LSTM layers. It can be seen from the results that with the increase of drop rate, the average of RMSE also increases. When the drop rate is 0.1, the RMSE of the model is the smallest, indicating that the deviation between the predicted value and the actual value is the smallest. Therefore, this research chooses the drop rate of 0.1 as the optimal parameter for the model.

Table 2: Experimental results of same drop rate but different hidden nodes

drop rate	hidden nodes	RMSE	Time	RMSE-AVG
0.1	32	1872.84	32.86	1738.06
	64	1749.40	34.27	
	128	1681.99	40.09	
	256	1722.12	67.94	
	512	1663.95	144.69	
0.2	32	2037.83	31.37	1787.48
	64	1771.09	35.36	
	128	1788.92	41.43	
	256	1681.01	63.81	
	512	1658.55	150.91	
0.4	32	2148.17	33.69	1843.00
	64	1967.15	33.61	
	128	1791.75	41.32	
	256	1666.39	64.45	
	512	1641.55	146.95	
0.6	32	2440.62	31.04	1983.67
	64	2107.61	35.51	
	128	1904.27	41.04	
	256	1720.01	64.81	
	512	1745.83	152.71	
0.8	32	2996.75	31.75	2284.02
	64	2595.42	35.15	
	128	2102.23	41.47	
	256	1874.07	64.20	
	512	1851.61	155.56	

In Table 3, the first column indicates the count of hidden nodes in the LSTM layer, the second column signifies the drop rate, the third column represents the RMSE observed on the test dataset, the fourth column denotes the training completion time, the fifth column represents the average of training completion time, and the sixth column indicates the average of RMSE on the test set with the same hidden nodes but different drop rates in LSTM layers.

The results indicate that an increase in the number of hidden nodes leads to a corresponding decrease in the average of RMSE. Notably, when there are 512 hidden nodes, the model attains the lowest RMSE, signifying minimal deviation between predicted values and actual values. Nevertheless, as demonstrated in Table 3, when the number of hidden nodes is fixed at 256, the average of the RMSE does not exhibit

ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

a significant deviation from the value obtained with 512 hidden nodes. Additionally, the average time of training is decreased by over 50%. Consequently, this research has chosen 256 hidden nodes in LSTM layer as the optimal parameter for the model.

Table 3: Experimental results of same hidden nodes but different drop rate

hidden nodes	drop rate	RMSE	Time	Time-AVG	RMSE-AVG
32	0.1	1872.84	32.86	32.15	2299.24
	0.2	2037.83	31.37		
	0.4	2148.17	33.69		
	0.6	2440.62	31.04		
	0.8	2996.75	31.75		
64	0.1	1749.40	34.27	34.78	2038.13
	0.2	1771.09	35.36		
	0.4	1967.15	33.61		
	0.6	2107.61	35.51		
	0.8	2595.42	35.15		
128	0.1	1681.99	40.09	41.07	1853.83
	0.2	1788.92	41.43		
	0.4	1791.75	41.32		
	0.6	1904.27	41.04		
	0.8	2102.23	41.47		
256	0.1	1722.12	67.94	65.04	1732.72
	0.2	1681.01	63.81		
	0.4	1666.39	64.45		
	0.6	1720.01	64.81		
	0.8	1874.07	64.20		
512	0.1	1663.95	144.69	150.16	1712.30
	0.2	1658.55	150.91		
	0.4	1641.55	146.95		
	0.6	1745.83	152.71		
	0.8	1851.61	155.56		

According to the parameters determined by the above two experiments, this research conducted tests on a dataset consisting of 600 samples, and the results are shown in Figure 3. The red curve represents the label values of the test set, while the blue curve signifies the predictions generated by the model. In the left panel of Figure 3, the visual representation indicates that the model's predictions closely align with actual values, and the trend is also close to the actual value. To further underscore the model's predictive capabilities, the right panel of Figure 3 is presented. In this figure, the first 2500 data points correspond to label values from the training set, while data points 2500 to 3000 represent model-generated predictions. It is evident that the model's forecast aligns with the overall power load trend.

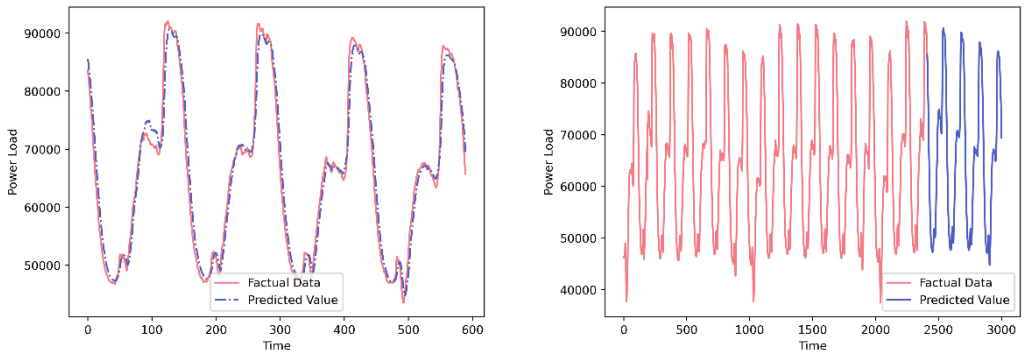

Figure 3: Comparison of predicted results with actual values

4. Conclusions

This research has employed the new deep learning framework, MindSpore, to implement a short-term

ISSN 2616-5775 Vol. 6, Issue 11: 9-16, DOI: 10.25236/AJCIS.2023.061102

power load forecasting model. The model consists of an LSTM layer, a dropout layer, and a fully connected layer. The experiments demonstrate that when the drop rate is set at 0.1 and the number of hidden nodes in the LSTM layer is set at 256, the model exhibits an appropriate balance between training time and performance. Furthermore, the prediction of model closely aligns with the actual power load variation trend. However, it's important to note that since only LSTM is employed for feature extraction, the prediction effect of the model is also insufficient. In the future, this research intends to use CNN and other technologies to extract power load data characteristics, thereby enhancing prediction accuracy.

References

[1] Din G M U, Marnerides A K. Short term power load forecasting using deep neural networks[C]. 2017 International conference on computing, networking and communications (ICNC). IEEE, 2017: 594–598.

[2] Al-Ghussain L. Global warming: review on driving forces and mitigation[J]. Environmental Progress & Sustainable Energy, 2019, 38(1): 13–21.

[3] Zheng J, Xu C, Zhang Z, et al. Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network [C]. 2017 51st Annual conference on information sciences and systems (CISS). IEEE, 2017: 1–6.

[4] Chauhan R, Ghanshala K K, Joshi R C. Convolutional neural network (CNN) for image detection and recognition[C]. 2018 first international conference on secure cyber computing and communication (ICSCCC). IEEE, 2018: 278–282.

[5] Zou Z, Chen K, Shi Z, et al. Object detection in 20 years: A survey[J]. Proceedings of the IEEE, IEEE, 2023.

[6] Khurana D, Koli A, Khatter K, et al. Natural language processing: state of the art, current trends and challenges [J]. Multimedia Tools and Applications, 2023, 82(3): 3713–3744.

[7] Pang B, Nijkamp E, Wu Y N. Deep Learning With TensorFlow: A Review[J]. Journal of Educational and Behavioral Statistics, 2020, 45(2): 227–248.

[8] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library [J]. Advances in neural information processing systems, 2019, 32.

[9] Huawei Technologies Co., Ltd. Huawei MindSpore AI Development Framework[A]. In: Artificial Intelligence Technology[M]. Singapore: Springer Nature Singapore, 2023: 137–162.

[10] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, MIT press, 1997, 9(8): 1735–1780.

[11] Hodson T O. Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not [J]. Geoscientific Model Development, Copernicus GmbH, 2022, 15(14): 5481–5487.