Comprehensive Review of Machine Learning-Based Methods for Electricity Load Forecasting

Zixu Zhao*

Jiangnan University, Wuxi 214122, China.

Abstract: With the improvement of data processing power and the continuous development of modern power grids, there is an increasing demand for accuracy in predicting power load. To study the field of power load forecasting, this article summarizes and categorizes different models into three types: traditional models, single machine learning models, and hybrid models, based on previous literature. Firstly, a general overview is provided of the application of different models in power load forecasting. Secondly, typical models from three categories are selected for a detailed introduction. In traditional models, the ARIMA model is chosen, while in single machine learning models, CNN, and LSTM are chosen. For the hybrid model, the ResNet-LSTM mixed neural network is selected for the introduction. Finally, four different datasets were used to test different models. The differences and patterns of the models were summarized, and suggestions were proposed for future research directions in the field of power load forecasting.

Keywords: Power load forecasting, ARIMA, CNN, LSTM, ResNet-LSTM.

1. Introduction

With the development of the economy, people's reliance on electricity is increasing. Due to the difficulty of storing electric energy, the output of power plants should be kept as consistent as possible with the power consumption. This means that accurate predictions of power loads can help relevant departments develop reasonable power dispatching plans. This ensures a reasonable arrangement of unit maintenance plans, guarantees normal production and life of society, effectively reduces power generation costs, and improves economic and social benefits. According to the predicted time, load forecasting can be divided into three types: short-term forecasting, medium-term forecasting, and long-term forecasting. Short-term forecasting can be provided one day to one week in advance, medium-term forecasting can be provided one week to several months or even a year in advance, and long-term forecasting can be provided more than one year in advance. The emphasis varies at different times. Short-term forecasts are mainly used to arrange power generation and transmission, medium-term forecasts focus on arranging fuel purchases. And long-term forecasts focus on overall improvements to power generation units, transmission systems, and distribution systems [1]. The electric power system requires strict control functions to produce high-quality electrical energy, which places high demands on the speed of correcting related parameters [2], therefore load forecasting is of great significance. This article will focus on discussing the relevant model methods for electric load forecasting, rather than excessively classifying them according to the length of the forecasting period.

There are roughly three types of models for electrical load forecasting: traditional models, single machine learning models, and hybrid models. Traditional models apply traditional mathematical, statistical, and operations research methods to electrical load forecasting, while making certain improvements on this basis. For example, Jin [3] et al. proposed a new grey correlation competition model based on the general GM (1,1) model, which improves the problem of decreased accuracy in traditional grey prediction models when dealing with significant load mutations. Nepal B [4] et al., on the other hand, combined clustering and ARIMA models for prediction. Furthermore, regression, multiple regression, exponential smoothing, and iterative re-weighted least squares techniques [5] are all traditional load forecasting methods. These methods are relatively simple, but at the same time, their drawbacks are also very obvious. They cannot accurately predict power load that exhibits non-linear changes and is influenced by multiple factors; thus the upper limit of predictive accuracy cannot be exceeded. With the improvement of data computing power and the continuous development of modern power grids, machine learning is gradually entering people's vision. Compared to traditional models, machine learning can better handle non-linear data. Various machine learning methods are also being applied to power load prediction.

Specifically, a single machine learning model includes models such as least squares support vector machine (LSSVM), artificial neural networks (ANN),

* Corresponding author: zzx_5656@163.com

convolutional neural networks (CNN), and so on. The following are examples. Yang et al. [6] combined the autocorrelation function (ACF) with least square support vector machines (LSSVM) to establish a model called AS-GCLSSVM for power load forecasting. The study aimed to select optimal input features (Feature Selection, FS) and uncover potential feature variables. Bakirtzis A G et al. [7] proposed a short-term load forecasting model based on an artificial neural network (ANN), which was applied to the energy control center of the Greek Public Power Corporation (PPC) using 63 input neurons, 24 hidden neurons, and 24 outputs for prediction. Imani M [8] mainly uses Convolutional Neural Networks (CNN) for extracting non-linear features of the load. Muzaffar S [9] et al. collected electric load data with external variables such as temperature, humidity, and wind speed, and used this data to train LSTM networks. Due to the high number of parameters and training costs of LSTM, a variant of LSTM called GRU was designed. For instance, Zhu [10] et al. applied the Gated Recurrent Unit (GRU) model to short-term electric vehicle load forecasting scenarios and achieved good performance. Whether using supervised learning support vector machines or neural networks such as GRU, CNN, and LSTM, each of them has its advantages. Compared to traditional models, they exhibit superior performance. However, each method also has its limitations to varying degrees. Therefore, to avoid their respective limitations, hybrid models that combine multiple methods have also begun to emerge continuously.

A hybrid model combines multiple neural networks or other algorithms to optimize the model. For example, as mentioned in the text, combinations of GRU, LSTM, and CNN have shown better results than using a single network. For instance, Tang et al. [11] proposed a multi-layer bidirectional recursive neural network model based on LSTM and GRU to predict short-term power load. The model was validated on two datasets, the European Intelligent Technology Network Competition data and Chongqing Power Company data, and achieved good results. Guo et al. [12] fused CNN and LSTM to create a multiscale CNN-LSTM hybrid neural network model for short-term load forecasting. Wu et al. [13] combined CNN and GRU for a similar purpose.

We can see that the combination of various models is flexible and diverse. Not only can networks mentioned in a single machine learning model be mixed, but other neural networks and algorithms can also be integrated. For example, Xie et al. [14] proposed a short-term power load forecasting method that combines the Elman neural network (ENN) and particle swarm optimization (PSO). Nie [15] proposed an RBF-GRNN-ELM composite prediction model. Dai et al. [16] transformed traditional support vector machines and proposed a hybrid model that combines intelligent feature selection and parameter optimization methods. Lv et al. [17] proposed a hybrid model based on the variational mode decomposition (VMD) and long short-term memory (LSTM), and seasonality elimination and error correction were also performed. Wang et al. [18] proposed an industrial user short-term load forecasting model based on Temporal Convolutional Network (TCN) and Light Gradient Boosting Machine (LightGBM), which was applied to data from multiple countries and validated for its accuracy compared to other models. Chen [19] et al. developed a hybrid model that combines Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN), Gated Recurrent Unit (GRU), and an improved Grey Wolf Optimizer (IGWO), named CEEMDAN-IGWO-GRU (CIG) hybrid algorithm. Huang [20] et al. proposed a comprehensive short-term power load prediction method based on Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN), Sample Entropy (SE), Back-Propagation Neural Network (BPNN), and Transformer Model.

Neural networks are rapidly evolving, and various new models combining different types of neural networks are constantly emerging. Chen et al. [21] proposed a hybrid model based on Residual Neural Network (ResNet) and Long Short-Term Memory (LSTM) to fully utilize the time characteristics. Liu et al. [22] proposed a short-term load forecasting method based on an improved Temporal Convolutional Network (TCN) and Dense Connection Convolutional Network (DenseNet), where parallel pooling was introduced based on traditional TCN to extract features from time series. The effectiveness of the model was demonstrated after applying it to four datasets, including Singapore and the United States. Some researchers have also attempted to incorporate other algorithms into models that have already combined multiple neural networks. Su et al. [23] proposed a variational mode decomposition optimization algorithm (PSVMD-CGA) that combines CNN, GRU, attention mechanism, and mahjong algorithm. Cai et al. [24] introduced a short-term load forecasting approach that integrates variational mode decomposition (VMD), gated recurrent unit (GRU), and temporal convolutional network (TCN) into a hybrid network. Yi et al. [25] optimized the CNN-LSTM model by introducing a self-attention mechanism (SAM). The hybrid models can be further divided into three types: combinations of multiple neural networks, combinations of a single neural network and other algorithms, and combinations of multiple neural networks and other algorithms.

However, after summarizing the literature, it was found that these models mainly have the following shortcomings:

1. The application scenarios are limited and only tested in a single scenario [8,10], not tested in multiple scenarios.
2. The computation time is comparatively longer and consumes more time than other models [15].
3. The testing and training data are insufficient [10], which can be further optimized.
4. To predict power load, it is necessary to add more irrelevant relevant variables, such as weather factors, temperature, holidays, and other events [6].
5. Optimizing the internal structure of the neural network, such as selecting the scale of the CNN convolution layer, affects the feature dimension and extraction accuracy, and further affects the accuracy of load forecasting [12].

In the next section of this article, we will explain typical models selected for traditional models, single machine learning models, and hybrid models separately. In section 3, we will compare and evaluate different models. Finally, we will summarize and offer suggestions based on multiple experiments.

2. Traditional Models

2.1 ARIMA Model based on K-means Clustering

The ARIMA model based on K-means clustering belongs to traditional models. This article will introduce the ARIMA model and K-means clustering method separately, and then introduce the ARIMA model incorporating K-means.

2.1.1 ARIMA

The ARIMA model is a famous time series prediction method proposed by Box and Jenkins in the early 1970s [26]. The ARIMA model is represented as ARIMA (p, d, q). ARIMA is essentially divided into three parts: AR, I, and MA. AR stands for autoregression, MA stands for moving average, and I stands for differencing. p is the order of autoregression, q is the order of moving average, and d is the order of differencing.

The ARIMA model requires data to be stationary. Before modeling, a stationarity test should be performed. If data is non-stationary, differencing can be performed to make it stationary (as mentioned earlier, d is used for differencing). If data is already stationary, then d is 0 and differencing is not needed.

The AR model is an autoregressive model that focuses on the relationship between current and past values, which is shown in formula (1):

$$y_t = \mu + \sum_{i=1}^{p} r_i y_{t-i} + \varepsilon_t \tag{1}$$

Where y_t is the current value, μ is the constant term, r_i is the autocorrelation coefficient, and ε_t is the error term. As for the autocorrelation function, it is shown in formula (2):

$$r_K = \frac{\sum_{t=1}^{n-k}[(X_t-\bar{X})(X_{t+k}-\bar{X})]}{\sum_{t=1}^{n}(X_t-\bar{X})^2}, k = 1,2,\cdots K \tag{2}$$

Where n is the sample size, and $\bar{X}$ is the sample mean. The MA model stands for moving average, which focuses on the accumulation of the error term, as shown in formula (3):

$$y_t = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \varepsilon_t \tag{3}$$

The ARMA model is a combination of the AR model and MA model, as shown in formula (4):

$$y_t = \mu + \sum_{i=1}^{p} r_i y_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{4}$$

The ARIMA model is an ARMA model that has been differenced d times. The values of p and q in the ARIMA model are determined based on the autocorrelation function (ACF) and partial autocorrelation function (PACF). Then, the model fitness is evaluated by checking the Akaike information criterion (AIC) and other criteria. Finally, the residuals ϵ is examined. If the residual ϵ is white noise, it indicates that all useful information in the time series has been extracted and the modeling process can be terminated.

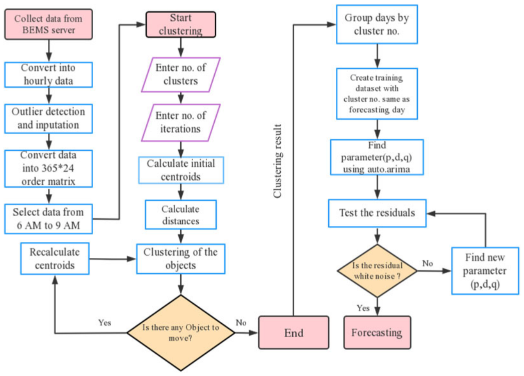
Collect data from BEMS server
Convert into hourly data
Outlier detection and inputation
Convert data into 365*24 order matrix
Select data from 6 AM to 9 AM
Start clustering
Enter no. of clusters
Enter no. of iterations
Calculate initial centroids
Calculate distances
Clustering of the objects
Recalculate centroids
Is there any Object to move?
Yes
No
End
Clustering result
Group days by cluster no.
Create training dataset with cluster no. same as forecasting day
Find parameter(p,d,q) using auto.arima
Test the residuals
Is the residual white noise ?
No
Find new parameter (p,d,q)
Yes
Forecasting

Fig.1 Flowchart of the model combining K-means and ARIMA. BEMS stands for the Building Energy Management System

2.1.2 Combination of K-means clustering and ARIMA

In reference [4], the ARIMA model is combined with K-means clustering[27] to predict the power consumption of buildings. The process is shown in Figure 1.

First, extract the existing data from the Building Energy Management System (BEMS) and process the data. Then, use K-means clustering to cluster the data into 6 classes as in [4]. After the clustering is completed, create a dataset that belongs to the same cluster as the predicted values, and then build an ARIMA model and select suitable p, d, and q values. Check whether the residuals ϵ are white noise during the process. And after the test, use the set ARIMA model for prediction.

2.2 CNN-Based Nonlinear Feature Extraction Model for Power Load

This model belongs to a single machine learning model, which accomplishes nonlinear feature extraction for power load prediction by constructing a convolutional neural network(CNN).

The first implementation of Convolutional Neural Networks (CNN) was carried out by LeCun Y [28] and his colleagues in 1989. CNN has shown excellent capability in capturing load data trends for load forecasting [29] and performs well in extracting and analyzing multidimensional data.

In literature [8], the prediction of electricity consumption for the next hour is finally made. To achieve this goal, two CNN networks are set up to extract load characteristics and load-temperature characteristics respectively. The sampling points for load and temperature are set to be collected once every hour, and the data collected for the previous two weeks for load is 24×7×2, while for temperature it is 24×7×1 for the previous week. The two CNN networks are named Load-CNN and Load-Temperature CNN respectively. The input for Load-CNN and Load-Temperature CNN is load data from the previous two weeks, as shown in Figure 2.

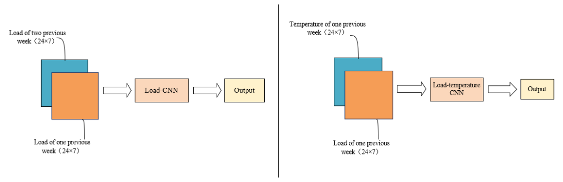
Load of two previous week（24×7）
Load of one previous week（24×7）
Load-CNN
Output
Temperature of one previous week（24×7）
Load of one previous week（24×7）
Load-temperature CNN
Output

Fig.2 Load-CNN, which extracts features by inputting the load data of the first two weeks into the Load-CNN neural network.And Load-Temperature CNN, which inputs the load and temperature data from the previous week into the Load-temperature CNN neural network for feature extraction.

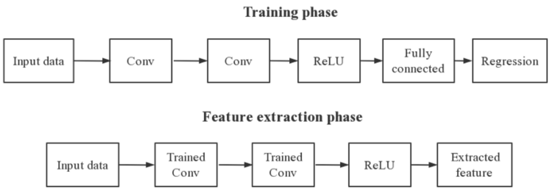

Fig.3 Phase diagram indicating the network architecture used during training stages and for feature extraction at the end.

The input of Load-CNN consists of 24 rows and 7 columns, corresponding to 24 hours and seven days within a week. The total input includes the load data from the first two weeks. The input of Load-temperature CNN includes the load data and temperature data from the previous week, also with 24 rows and 7 columns corresponding to 24 hours and seven days within a week. For two neural networks, training is required. During training, the output is the load value for the next hour, but in the end, the output becomes the load features to be extracted and the load-temperature features.

CNN has two main characteristics: weight sharing and local connections. A typical CNN consists of several parts, including convolutional layers, activation functions, pooling layers, fully connected layers, etc. Here, the max pooling layer is omitted in the design to avoid discarding some features. Additionally, the network used during the training stage and the final feature extraction stage may differ. The process is shown in Figure 3:

During training, two convolutional layers are used followed by an activation function using the ReLU function, then a fully connected layer and a regression layer. For feature extraction after training, two convolutional layers and ReLU activation function are used for feature extraction. For the convolutional layer, as shown in formula (5):

$$\mathrm{F}^{\mathrm{l}} = \mathrm{g}(\mathrm{F}^{\mathrm{l}-1} * \mathrm{W}^{\mathrm{l}} + \mathrm{b}^{\mathrm{l}}) \qquad (5)$$

Where $\mathrm{F}^{\mathrm{l}-1}$ represents the feature map of the (l-1)th layer, W^{l} and b^{l} are learnable parameters representing the weights and biases of the lth layer. $*$ represents the linear convolution operator. $\mathrm{g}(\cdot)$ represents the ReLU activation function defined as formula (6):

$$\mathrm{g}(\mathrm{x}) = \max(0, \mathrm{x}) = \begin{cases} \mathrm{x}, \mathrm{x} > 0 \\ 0, \mathrm{x} \leq 0 \end{cases} \qquad (6)$$

The initial input of the first layer has been mentioned previously. $\mathrm{F}^{\mathrm{l}-1}$ has a size of $W_d^{l-1} \times W_h^{l-1} \times W_n^{l-1}$, where W_d^{l-1} and W_h^{l-1} are the width (corresponding to the day) and height (corresponding to the hour) of the feature map of the (l-1)th layer, and W_n^{l-1} represents the number of outputs of the (l-1)th layer. The neural network uses filters composed of convolution kernels with a size of 3×3 and a stride of 1.

In order to perform feature extraction, neural networks need to be trained. Assuming there are M samples, and their next-hour load values are known. The actual values during training are set as y_i^a {i = 1, 2, ... M}, with predicted values as y_i^f {i = 1, 2, ... M}. Assuming there are L layers, the formula (7) is as follows:

$$y_i^f = \mathrm{W}^{\mathrm{L}}\mathrm{F}^{\mathrm{L}-1} + \mathrm{b}^{\mathrm{L}}; i = 1,2,\dots M \qquad (7)$$

Therefore, the mean square error loss function is established, as shown in formula (8):

$$\psi = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{M}} |y_i^a - y_i^f|^2 \qquad (8)$$

The backpropagation algorithm is widely used to optimize weight W^{l} and bias b^{l} by minimizing the loss function. Prediction errors propagate from the higher layer to the previous layer, and the parameters of each layer are modified based on the propagation error. Stochastic gradient descent and Adam optimizer can be used to implement weight W^{l} and bias b^{l}. Finally, the trained CNN network can be used for nonlinear feature extraction.

2.3 LSTM-based power load prediction model

This model belongs to a single machine learning model, which uses a constructed LSTM neural network to perform electric load forecasting. LSTM stands for Long Short Term Memory, first proposed by Hochreiter S [30], it is a variation of Recurrent Neural Network (RNN) [31] with the ability to learn long-term dependencies. LSTM overcomes the impact of short-term memory, characterized by cell and "gate" structures.

The cell structure of LSTM is shown in Figure 4, which provides memory capability through the gating structure in the cell. Multiple cells are connected to form the hidden layer. The symbol "×" in the figure represents pointwise multiplication, and the symbol "+" represents pointwise addition. The cell structure includes a forget gate, an input gate, a cell state, and an output gate.

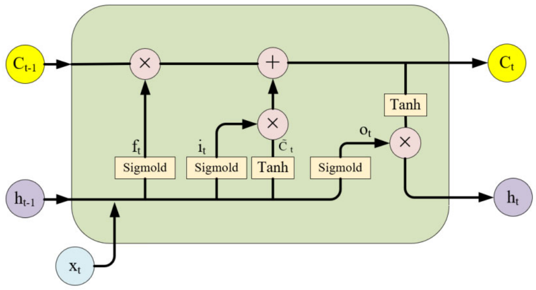

Fig.4 LSTM cell diagram, mainly composed of forget gate, input gate, output gate, and others.

The forget gate determines which information should be discarded or retained. Information from the previous hidden state h_{t-1} and the current input x_t are passed into a Sigmoid function, with the output value ranging between 0 and 1. The closer the output value is to 0, the more it should be discarded. This is expressed in formula (9):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \qquad (9)$$

The input gate is used to update the cell state. First, the information of the previous hidden state h_{t-1} and the current input information x_t are passed to the Sigmoid function. Then, the information of the previous hidden state h_{t-1} and the current input information x_t are passed to the tanh function to create a new candidate value vector. Finally, the output value of the Sigmoid function and the output value of the tanh function are multiplied. This is shown in equations (10) and (11).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{10}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{11}$$

The cell state C_t is obtained through the forget gate, input gate, and the previous layer's cell state. As shown in formula (12):

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{12}$$

The output gate is used to determine the value of the next hidden state, which contains information from previous inputs. First, the information of the previous hidden state h_{t-1} and the current input information x_t are passed to the Sigmold function, and then the newly obtained cell state C_t is passed to the tanh function. Finally, the output of the tanh function is multiplied by the output of the Sigmoid function to determine the information the hidden state should carry, which is then passed to the next cell. Formulas (13) and (14):

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{13}$$

$$h_t = o_t \times tanh(c_t) \tag{14}$$

In order to provide appropriate training data, two vectors are generated [9] for each sequence: PredTrain and RespTrain, where PredTrain represents the predicted value and RespTrain represents the expected value. PredTrain is the input TS with the last 5 values removed, while RespTrain is the input TS with the first 5 values removed. The optimal lag chosen here is 5, meaning that each response depends on the previous 5 values. The training data for both input TS is generated in the same way. Predictive test vectors are also generated using only 13 months of data. "end" represents the end of the data, "12MonthsData" represents the first 12 months of data, and "13MonthData" represents the 13th month's data. The format for training and predictive testing data is shown in Equations (15), (16), (17), and (18):

$$\text{PredTrain} = \text{12MonthsData}(1\text{: end} - \text{lag}) \tag{15}$$

$$\text{RespTrain} = \text{12MonthsData}(\text{lag} + 1\text{: end}) \tag{16}$$

$$\text{PredTest} = \text{13MonthData}(1\text{: end} - \text{lag}) \tag{17}$$

$$\text{RespTest} = \text{13MonthData}(\text{lag} + 1\text{: end}) \tag{18}$$

2.4 Hybrid Power Load Forecasting Model Based on Resnet and LSTM.

2.4.1 Resnet

Resnet was first proposed in [32] to solve the degradation, vanishing, and exploding gradients problems that occur as neural networks become deeper. Vanishing gradients refer to the problem where the gradient approaches zero as the network gets deeper, resulting in difficulty in backpropagation when the error gradient of each layer is less than 1. On the other hand, exploding gradients happen when the error gradient of each layer is larger than 1, resulting in the gradient getting larger as the network gets deeper. The characteristics of this network are: (1) an ultra-deep network structure with over 1000 layers, (2) the introduction of residual blocks, and (3) the use of Batch Normalization [33] to accelerate training.

A. Batch Normalization

For addressing the issues of vanishing gradients and optimizing gradients, Batch Normalization is mainly employed. This operation is performed between each fully connected layer and activation function. The Batch Normalization algorithm normalizes the feature maps of a batch of data to follow a distribution pattern with a mean of 0 and variance of 1. This is achieved mainly through equations (19), (20), (21), and (22).

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{19}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{20}$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{21}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{22}$$

Here, $\mathcal{B} = \{x_{1...m}\}$ is the set of input values x_i , ϵ is a small constant used to prevent division by zero, y_i is the input, $\sigma_{\mathcal{B}}^2$ represents variance, with each element representing the variance in each dimension, and similarly $\mu_{\mathcal{B}}$ represents the mean, with each element representing the mean in each dimension. $\sigma_{\mathcal{B}}^2$ and $\mu_{\mathcal{B}}$ are calculated, while γ and β are parameters learned in backpropagation. γ adjusts the variance of the value distribution, with an initial value of 1, while β adjusts the position of the value mean, with an initial value of 0.

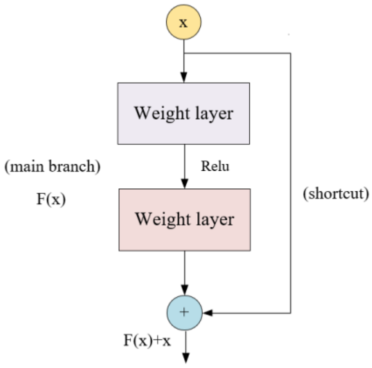

Fig.5 Residual block, where F(X) has two layers of weight layer and one layer of activation function.

B. residual block

To solve the degradation problem in deep neural networks, a residual block is applied to address it. The residual block adds a shortcut connection to weaken the strong correlation between each layer. Assuming the input is x and the expected output is H(x), which represents the actual mapping, the interpolation between the expected output and input values is F(x). F(x) is the residual mapping. Therefore, we can obtain formulas (23) and (24):

$$\text{F(x)} = \text{H(x)} - \text{x} \tag{23}$$

$$\text{H(x)} = \text{F(x)} + \text{x} \tag{24}$$

As shown in Figure 5, if the network tends to be optimal, continue to deepen the network. If the residual mapping F(x) becomes 0, H(x) = x; theoretically the performance of the network will not decrease with the increase of depth. If F(x) ≠ 0, but F(x) is close to 0, then x approximates the actual mapping H(x), and the problem of degradation is solved. However, it should be noted whether the shapes of the main branch and shortcut are the same, only when they are the same can they be added together. Therefore, in Resnet, the residual structure is divided into solid line residual structure and dotted line residual structure. When the shape of the main branch in the residual block is the same as the shape of the shortcut, the main branch and shortcut branches can be directly added together, which is the solid line residual structure. When the shapes are different, the shortcut branch needs to be processed by convolution operation to make the feature matrix shape of the shortcut branch consistent with that of the main branch, and then add them together.

2.4.2 Hybrid Model of Resnet and LSTM

The principle of LSTM has been mentioned in the previous text. Here we introduce a hybrid model of Resnet and LSTM based on literature [18]. This model uses Resnet as the pre-feature extraction unit and LSTM as the time series feature learning unit.

For the Resnet network part in the hybrid model, Resnet18 is adopted, consisting of five parts: Conv1, Conv2_x, Conv3_x, Conv4_x, and Conv5_x. Conv1 performs convolution and max pooling operations. The other four parts each consist of 2 residual blocks, with each residual block composed of two layers of a convolutional network. The feature matrices of the two branches of the residual block are added together and then activated by the ReLU function before being output. The data structure input to Resnet is (None, 56,1), where 56 is the number of feature parameters and the number of channels is 1. The number of convolutional kernels for each part is 32, 32, 64, 128, and 256, respectively. The size of the convolutional kernel is 3×3. In the max pooling layer, the pool size is also 3×3, and the step size is 2. After feature extraction by Resnet, a three-dimensional vector (None, 2, 256) is output to the LSTM network.

In this model, the residual block of the solid line residual structure performs a convolution operation with a stride of 1, while the dashed line residual structure performs a convolution operation with a stride of 2 on the shortcut branch, ensuring that the feature matrices of the short cut branch and the main branch have the same shape.

For the LSTM network, two layers are set here. The number of units in each layer is 1024 and 256. Dropout is used between the layers of the LSTM network to prevent overfitting of the model. Finally, two Dense layers are used to output the predicted load value. The first layer has 64 neuron nodes and the second layer has 1 neuron node.

3. Model Evaluation

In this section, we will select typical models from several categories mentioned earlier and compare their performance on different datasets to evaluate their strengths and weaknesses.

3.1 Evaluation Indicators.

Model evaluation metrics are diverse, and a single metric cannot fully reflect model performance. Therefore, this paper selects Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) [34] to compare and evaluate the models. The specific content and calculation formulas of each metric are shown in Table 1.

Table 1 Evaluation Indicators and Calculation Formulas.

Metric	Definition	Equation
MAPE	Average absolute percentage errors	$MAPE=\frac{1}{N}\sum_{n=1}^{N}\lvert\frac{\hat{y}_i-y_i}{y_i}\rvert\times 100\%$
MAE	Average absolute error	$MAE=\frac{1}{N}\sum_{n=1}^{N}\lvert\hat{y}_i-y_i\rvert$
RMSE	Root Mean Square Error	$RMSE=\sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{y}_i-y_i)^2}$

N represents the total number of samples, y_i represents the actual value of the sample, and $\hat{y}_i$ represents the predicted value of the sample. MAPE, MAE, and RMSE are all indicators where the smaller the value, the better.

3.2 Model Comparison

This article selected experimental results from four references [17,21,25,35] for model comparison. The source of each dataset is shown in Table 2.

Table 2 Dataset description and the dataset source

Dataset description	Dataset source
The publicly available online load data set published in Queensland, Australia from Jan. 1, 2006, to Dec.31, 2010	https://github.com/weiran4/AustraliaData
Real load data from the dataminer2 website of PJM in the US.	Data Miner 2 (pjm.com)
The building data set derived from hourly load data provided by the UTD from 1 January 2014 to 31 December 2015.	https://ieee-dataport.org/documents/short-term-load-forecasting-data-hierarchical-advanced-metering-infrastructure-and-weather
A pool of 69 customers out of thousands of customers' data, which were retrieved from the SGSC project initiated by the Australian Government	http://www.industry.gov.au/ENERGY/PROGRAMMES/SMARTGRIDSMARTCITY/Pages/default

3.2.1 Queensland Dataset in Australia

The publicly available online load dataset in Queensland, Australia from January 1, 2006, to December 31, 2010, contains 87649 rows of data. It has a sampling interval of 30 minutes with 48 sampling points per day. The dataset includes features such as load, dry-bulb temperature (drybt), dew point temperature (dewbt), wet-bulb temperature (wetbt), humidity, and price. The data from the first four years is used as the training set, data from January 1 to November 30, 2010, is used as the validation set, and the last month's data is used as the test set. Testing was conducted using data from December 1 to 7, 2010, and the results are shown in Table 3.

Table 3 Comparison of indicators of various models on the Australian dataset

Model	MAPE	MAE	RMSE
MLR	0.769	69.938	84.307
LSTM	0.822	65.982	89.396
CNN	0.815	67.747	86.494
Resnet	0.753	62.488	79.653
CNN-LSTM	0.830	69.501	89.264
Resnet-LSTM	**0.597**	**49.301**	**62.933**

*MLR, Mixed Logistic Regression, uses piecewise linear regression to fit nonlinear data.

It can be seen that Resnet-LSTM outperforms other models in terms of MAPE, MAE, and RMSE, demonstrating excellent performance. Among the six models compared, Resnet performs better than CNN and LSTM, as well as their hybrid network CNN-LSTM. However, the performance of Resnet is inferior to the Resnet-LSTM hybrid network. Resnet also outperforms the MLR algorithm.

3.2.2 Real load data from the dataminer2 website of PJM in the US

The data sets were collected from six load groups in the Mid-Atlantic, PJM_RTO, and Western regions of the United States, from November 22, 2020, to January 17, 2021, and from February 1, 2021, to March 28, 2021. They are represented as 2020 MID, 2020 PJM, 2020 Western, 2021 MID, 2021 PJM, and 2021 Western. These data sets were sampled 24 times a day, for a total of 1344 samples. The performance of models is shown in Table 4. Only the results of 2020 MID and 2021 MID are shown here.

Table 4 Model results based on the dataminer2 website dataset in PJM in the United States.

Distance	Model	Metrics		
		RMSE	MAE	MAPE(%)
2020 Mid	CEEMD+LSTM	430	335	1.1
	ARIMA	1015	725	2.3
	VMD+LSTM	**388**	**309**	**1**
	GRU+RNN	559	426	1.3
2021 Mid	CEEMD+LSTM	537	408	1.6
	ARIMA	939	690	2.7
	VMD+LSTM	**392**	**313**	**1.2**
	GRU+RNN	747	543	2.0

*CEEMD stands for Complementary Ensemble Empirical Mode Decomposition. VMD stands for Variational mode decomposition. GRU is a gated recurrent unit, and RNN stands for the recurrent neural network.

CEEMD and VMD are two different signal decomposition algorithms, while GRU and RNN are types of neural networks. Through comparison, it can be found that in these sections, the performance of the ARIMA model is significantly inferior to other neural network models. Among the three hybrid models involving neural networks, the VMD+LSTM model is the best.

3.2.3 The building data provided by the UTD

The dataset includes data from January 1st, 2014 to December 31st, 2015. The dataset was divided into training, validation, and test sets in a ratio of 0.6/0.2/0.2. The purpose of the test was to evaluate the performance of different neural networks with attention mechanisms. The results of the test are shown in Table 5.

Table 5 Results of the model on the building data provided by UTD.

Model	MAPE	MAE	RMSE
LSTM	0.070	2.991	3.904
LSTM-based SAM	0.067	2.884	3.826
CNN-GRU-based SAM	0.079	3.370	4.323
LSTM-CNN-based SAM	**0.027**	**1.144**	**1.589**

* SAM stands for self-attention mechanism, which is introduced to pay attention to the correlations among input data during the training process.

Comparing the results of the four models, the LSTM network fused with SAM outperformed the original LSTM network. However, the performance of the CNN-GRU-based SAM hybrid network was poor, performing worst among the four models. The LSTM-CNN-based SAM model was the best, with all numerical values significantly lower than the other models.

3.2.4 *The data retrieved from the SGSC project*

The dataset consists of 69 customers selected for the SGSC project initiated by the Australian government. These customers were selected based on the criteria of being energy users with a hot water system. The available data volume varies for each customer. Data were selected from June 1, 2013, to August 31, 2013. The data span for each customer is 92 days. The data is divided into training, validation, and testing data sets in a ratio of 0.7/0.2/0.1. At the same time, different time steps were set during the inspection, including 2, 6, and 12. However, the model metric used here is Average MAPE Individual Forecasts. The model metrics obtained by applying different models and time backstep lengths are shown in Table 6.

Table 6 Indicators using different models and time steps.

Model	Scenario(Lookback)	Average MAPE Individual Forecasts(%)
CNN-LSTM	2-time steps	**40.38**
	6-time steps	41.07
	12-time steps	42.85
LSTM	2-time steps	44.39
	6-time steps	44.31
	12-time steps	**44.06**
BPNN	2-time steps	49.62
	6-time steps	**49.04**
	12-time steps	49.49
KNN	2-time steps	74.83
	6-time steps	**71.18**
	12-time steps	81.13
ELM	2-time steps	**122.90**
	6-time steps	136.49
	12-time steps	123.45

*BPNN stands for Back Propagation Neural Network, KNN stands for k-nearest neighbor model, and ELM stands for Extreme Learning Machine, which is a method based on Feedforward Neuron Network (FNN).

Based on the above data, it can be seen that different parameters can lead to different results for the same model. For CNN-LSTM and ELM, 2-time steps are optimal; for LSTM, 12-time steps are optimal; for BPNN and KNN, 6-time steps are optimal. Overall, CNN-LSTM performs better than other single networks.

4. Conclusion and Outlook

Although it is not possible to compare the results obtained from the four datasets in Section 3 in a unified way, some characteristics related to the model can be summarized based on the results presented in each dataset. Therefore, the conclusions obtained in this article are summarized as follows:

1. Traditional model has a simple structure and performs well in special scenarios for fitting and prediction. However, overall, its performance is inferior to that of a single machine-learning model or a hybrid model.
2. A single machine learning model solves non-linear problems compared to traditional models. However, due to the use of only one approach, there are inevitably some shortcomings in some aspects.
3. Hybrid models combine multiple neural networks or other algorithms to optimize the model. Overall, hybrid models outperform single machine learning models, but hybrid models are more complex than single machine learning models. It cannot be said that the performance of a model after combining multiple neural networks is always better than the original model. The appropriate model should be selected based on the actual situation. Improper hybridization can degrade model performance.
4. Parameters also affect the performance of the model. After determining the network type and algorithm type, fine adjustment of the model parameters should also be done.

According to the experience in this article, there are the following optimization directions for future research:

1. Neural networks still have great potential for development, and can improve the shortcomings of a single network to form a new single network with better performance.
2. New combinations can be attempted based on existing neural networks and algorithms. Improve model performance by using new combination methods to compensate for their respective weaknesses.
3. Optimize the internal parameters of the model to improve its generalization ability.

References

1. Almeshaiei E, Soltan H. A methodology for electric power load forecasting[J]. Alexandria Engineering Journal, 2011, 50(2): 137-144.
2. Jahan I S, Snasel V, Misak S. Intelligent systems for power load forecasting: A study review[J]. Energies, 2020, 13(22): 6105.
3. Jin M, Zhou X, Zhang Z M, et al. Short-term power load forecasting using grey correlation contest modeling[J]. Expert Systems with Applications, 2012, 39(1): 773-779.
4. Nepal B, Yamaha M, Yokoe A, et al. Electricity load forecasting using clustering and ARIMA model for energy management in buildings[J]. Japan Architectural Review, 2020, 3(1): 62-76.
5. Singh A K, Khatoon S, Muazzam M, et al. Load forecasting techniques and methodologies: A review[C]//2012 2nd International Conference on Power, Control and Embedded Systems. IEEE, 2012:
6. Yang A, Li W, Yang X. Short-term electricity load forecasting based on feature selection and Least Squares Support Vector Machines[J]. Knowledge-Based Systems, 2019, 163: 159-173.
7. Bakirtzis A G, Petridis V, Kiartzis S J, et al. A neural network short term load forecasting model for the Greek power system[J]. IEEE Transactions on power systems, 1996, 11(2): 858-863.
8. Imani M. Electrical load-temperature CNN for residential load forecasting[J]. Energy, 2021, 227: 120480.
9. Muzaffar S, Afshari A. Short-term load forecasts using LSTM networks[J]. Energy Procedia, 2019, 158: 2922-2927.
10. Zhu J, Yang Z, Guo Y, et al. Short-term load forecasting for electric vehicle charging stations

based on deep learning approaches[J]. Applied sciences, 2019, 9(9): 1723.

11. Tang X, Dai Y, Wang T, et al. Short-term power load forecasting based on multi-layer bidirectional recurrent neural network[J]. IET Generation, Transmission & Distribution, 2019, 13(17): 3847-3854.
12. Guo X, Zhao Q, Zheng D, et al. A short-term load forecasting model of multi-scale CNN-LSTM hybrid neural network considering the real-time electricity price[J]. Energy Reports, 2020, 6: 1046-1053.
13. Wu L, Kong C, Hao X, et al. A short-term load forecasting method based on GRU-CNN hybrid neural network model[J]. Mathematical Problems in Engineering, 2020, 2020.
14. Xie K, Yi H, Hu G, et al. Short-term power load forecasting based on Elman neural network with particle swarm optimization[J]. Neurocomputing, 2020, 416: 136-142.
15. Nie Y, Jiang P, Zhang H. A novel hybrid model based on combined preprocessing method and advanced optimization algorithm for power load forecasting[J]. Applied Soft Computing, 2020, 97: 106809.
16. Dai Y, Zhao P. A hybrid load forecasting model based on support vector machine with intelligent methods for feature selection and parameter optimization[J]. Applied energy, 2020, 279: 115332.
17. Lv L, Wu Z, Zhang J, et al. A VMD and LSTM based hybrid model of load forecasting for power grid security[J]. IEEE Transactions on Industrial Informatics, 2021, 18(9): 6474-6482.
18. Wang Y, Chen J, Chen X, et al. Short-term load forecasting for industrial customers based on TCN-LightGBM[J]. IEEE Transactions on Power Systems, 2020, 36(3): 1984-1997.
19. Chen Z, Jin T, Zheng X, et al. An innovative method-based CEEMDAN–IGWO–GRU hybrid algorithm for short-term load forecasting[J]. Electrical Engineering, 2022, 104(5): 3137-3156.
20. Huang S, Zhang J, He Y, et al. Short-Term Load Forecasting Based on the CEEMDAN-Sample Entropy-BPNN-Transformer[J]. Energies, 2022, 15(10): 3659.
21. Chen X, Chen W, Dinavahi V, et al. Short-Term Load Forecasting and Associated Weather Variables Prediction Using ResNet-LSTM Based Deep Learning[J]. IEEE Access, 2023.
22. Liu M, Qin H, Cao R, et al. Short-Term Load Forecasting Based on Improved TCN and DenseNet[J]. IEEE Access, 2022, 10: 115945-115957.
23. Su J, Han X, Hong Y. Short Term Power Load Forecasting Based on PSVMD-CGA Model[J]. Sustainability, 2023, 15(4): 2941.
24. Cai C, Li Y, Su Z, et al. Short-Term Electrical Load Forecasting Based on VMD and GRU-TCN Hybrid Network[J]. Applied Sciences, 2022, 12(13): 6647.
25. Yi S, Liu H, Chen T, et al. A deep LSTM-CNN based on self-attention mechanism with input data reduction for short-term load forecasting[J]. IET Generation, Transmission & Distribution, 2023.
26. Ahmed M S, Cook A R. Analysis of freeway traffic time-series data by using Box-Jenkins techniques[M]. 1979.
27. MacQueen J. Classification and analysis of multivariate observations[C]//5th Berkeley Symp. Math. Statist. Probability. 1967: 281-297.
28. LeCun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition[J]. Neural computation, 1989, 1(4): 541-551.
29. Rafi S H, Deeba S R, Hossain E. A short-term load forecasting method using integrated CNN and LSTM network[J]. IEEE Access, 2021, 9: 32436-32448.
30. Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
31. Elman J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.
32. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
33. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. pmlr, 2015: 448-456.
34. Jin Y, Guo H, Wang J, et al. A hybrid system based on LSTM for short-term power load forecasting[J]. Energies, 2020, 13(23): 6241.
35. Alhussein M, Aurangzeb K, Haider S I. Hybrid CNN-LSTM model for short-term individual household load forecasting[J]. Ieee Access, 2020, 8: 180544-180557.