Article

Integration of Ensemble GoogLeNet and Modified Deep Residual Networks for Short-Term Load Forecasting

Aijia Ding ¹, Tingzhang Liu ^{1,*} and Xue Zou ^{2,*}

¹ School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China; dingaijia@shu.edu.cn

² Zibo Liming Electric Appliance Co., LTD., Zibo 255000, China

* Correspondence: liutzhcom@oa.shu.edu.cn (T.L.); xuexin0777@163.com (X.Z.)

Abstract: Due to the strong volatility of the electrical load and the defect of a time-consuming problem, in addition to overfitting existing in published forecasting methods, short-term electrical demand is difficult to forecast accurately and robustly. Given the excellent capability of weight sharing and feature extraction for convolution, a novel hybrid method based on ensemble GoogLeNet and modified deep residual networks for short-term load forecasting (STLF) is proposed to address these issues. Specifically, an ensemble GoogLeNet with dense block structure is used to strengthen feature extraction ability and generalization capability. Meanwhile, a group normalization technique is used to normalize outputs of the previous layer. Furthermore, a modified deep residual network is introduced to alleviate a vanishing gradient problem in order to improve the forecasting results. The proposed model is also adopted to conduct probabilistic load forecasting with Monte Carlo Dropout. Two acknowledged public datasets are used to evaluate the performance of the proposed methodology. Multiple experiments and comparisons with existing state-of-the-art models show that this method achieves accurate prediction results, strong generalization capability, and satisfactory coverages for different prediction intervals, along with reducing operation times.

Keywords: short-term load forecasting; GoogLeNet; modified deep residual network; convolution; probabilistic load forecasting

Citation: Ding, A.; Liu, T.; Zou, X. Integration of Ensemble GoogLeNet and Modified Deep Residual Networks for Short-Term Load Forecasting. *Electronics* **2021**, *10*, 2455. <https://doi.org/10.3390/electronics10202455>

Academic Editor: Christos J. Bouras

Received: 29 September 2021

Accepted: 8 October 2021

Published: 10 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. 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/>).

1. Introduction

Modern power systems are in a new era of coexistence with combined traditional and renewable energy power generation modes, with the latter especially generated by wind and photovoltaic (PV) sources. Due to design codes [1], most installed capacity of PV is from 1 to 30 MWp, while PV stations are generally connected to 10 or 35 kV, which belongs to the voltage level of distributed electrical networks. The integration of new energy sources brings challenges not only to regional energy forecasts, but also to total energy demand prediction.

Accurate load forecasting plays an important role in ensuring the safe and stable economic operation of the power system, which is conducive to not only maintaining the normal survival function of a city, but also to optimizing the allocation of resources and alleviating the increasingly tense energy pressure. Short-term load forecasting (STLF) is the basic work of power enterprise planning, scheduling, dispatching, power consumption, and other departments.

Particularly, STLF often denotes a prediction for a leading time of one hour to a few days, while one hour ahead forecasting is mostly studied by researchers [2–13]. The methodologies of STLF proposed in the literature can be divided into two categories, namely statistical and artificial intelligence (AI) methods. Many approaches for statistical methods have been proposed in recent decades, such as multiple regression models [14], exponential smoothing [15], Kalman filters [16], and the autoregressive moving average method [17]. Nevertheless, these traditional methods cannot yield high accuracy when

facing inevitable large uncertainties in load data and flexible meteorological data. By contrast, the performance of AI and hybrid methods outweighs statistical ones with the development of machine learning. One of the most prevalent machine learning methods, support vector regression (SVR), has great generalization ability and allows for a rapid global solution. It does not perform well, however, when dealing with large data. The typical back-propagation neural network (BPNN) is proposed to handle large data due to its powerful mapping ability. However, BPNN is very easily trapped in a local optimal solution [18]. Meanwhile, gradient vanishing is more likely to occur when the network gets deeper. To address this, Chen et al. proposed a deep residual neural network [19] to forecast 24 h load for the next day using eleven inputs including historical load and temperature and one-hot codes of weekday, season, and festival. Another problem in BPNN is that the intrinsic characteristics existing in the time series are neglected. To resolve this, a recurrent neural network (RNN) [20] is proposed to take into account the short memory of previous time series. The disadvantages of gradient explosion and vanishing, however, restrict RNN's scalability. To make up for this deficiency, long short-term memory (LSTM) is obtained by adding a state vector and introducing a gate mechanism to RNN. A hybrid model [3] combining multivariable linear regression (MLR) and a LSTM neural network forecasted low and high frequency components, respectively, based on ensemble empirical-mode decomposition (EEMD) technology. This model makes the most of the advantages of MLR and LSTM, while the LSTM is sensitive to the fine part and the MLR is excellent in fitting low-frequency time series. LSTM, however, is relatively complicated and results in high computing costs. Therefore, a gated recurrent unit (GRU), one of the most popular LSTM variants, was proposed to combine a state vector and output vector and to reduce the number of gates to two. A model EGM [21] combined EEMD, GRU, and MLR to forecast the load one hour ahead with only one variable, namely the historical load. The EGM method can accurately predict the local characteristics with strong randomness. A hybrid model [22] combining variational mode decomposition and long short-term memory networks and taking relevant factors with Bayesian optimization algorithms into consideration obtained high forecasting accuracy. An ensemble method [23] exploited the full wavelet packet transform model to decompose features into various components and then fed them into a trained neural network. This method reduced the MAPE by 20% with regards to traditional neural network methods. The literature [24] proposed a fuzzy theory-based machine learning model for workdays and short-term load forecasting for weekends, which enhanced the forecasting stability and accuracy.

The previous methods only take temporal features into consideration, while spatial and shape characteristics can also affect the real-time load. A convolutional neural network (CNN) was further developed into LeNet by Yann Lecun to address issues in the field of computer vision, including handwritten digit recognition in 1989. The core concept existing in CNN involves local correlation and weight sharing. A hybrid model [4] combined CNN, LSTM, and GRU, where the LSTM and GRU both have the ability to store short-term and long-term memories while CNN has a significant ability to extract the input variable features. A CNN-LSTM neural network [25] is proposed to predict residential energy consumption. This novel network extracts temporal and spatial features from input variables that influence the energy consumption. An improved CNN [26] for probabilistic load prediction was put forward by discretizing continuous load range. A hybrid model combining CNN with multiple wavelets [27] is proposed for deployment in practice. However, the problems of overfitting together with huge computational effort and time consumption occur. To resolve this, GoogLeNet was proposed as a new deep learning structure by Szegedy et al. in 2014 and extracts more characteristics with the same amount of computation, while previously proposed structures obtain decent results with numerous hidden layers. A 22-layer GoogLeNet [28] is proposed and evaluated in the classification and detection of context. GoogLeNet's inception structure strengthens the computing power of the entire network, and the whole model is lightweight. GoogLeNet has emerged as one of the most influential improved network structures which adopted inception

mechanism that utilizes multi-scale processing to image. Simultaneously adopted as a forecasting methodology, GoogLeNet has been utilized in various fields, such as wind power generation forecasting and sun coverage forecasting. A GoogLeNet based CNN [29] was proposed to forecast solar irradiance. However, GoogLeNet has weak expandability and cannot be adopted as a potential model.

1.1. Contributions and Findings

In this paper, we aimed to expand the existing neural network structure of CNN by utilizing dense blocks, residual network structure, and realization techniques. We learned from the GoogLeNet [28] structure and proposed an end-to-end neural network instead of simply stacking convolutional and hidden layers.

Three points cover the contributions of this work. First, a hybrid model consisting of ensemble GoogLeNet and modified deep residual neural networks for short-term load forecasting is proposed. The model we propose involves no external variables except temperature and applies no complicated feature extraction technology except for Pearson correlation analysis. It is the first time that GoogLeNet is combined with dense blocks and a deep residual network structure. The input of each inception module comes from the concatenation of previous inception modules' outputs and the inputs of the first inception module. Second, the output of the last inception module, together with one-hot codes, is added to a deep residual network in order to enhance the fitting ability and to avoid gradient vanishing. Third, we obtained a probabilistic load forecast by applying Monte Carlo (MC) dropout to the proposed model trained for point forecasting. The indices of the Winkler score of different confidence coefficients and pinball loss are calculated to estimate the performance of probabilistic load forecasting.

To assess the model's generalization ability, two widely recognized datasets are utilized. Then, compared with various methods, including ANN + RF, HEFM, LWSVR-MGOA and WT-ELM-MABC, the proposed model obtained the best performance.

To evaluate the model's robustness, two measures were adopted. One is that we added Gaussian noise with 1 °F standard variation to the observed temperature at the forecast time so that we could see how the model performs when we use modified temperature. The other measure we used is the predicted temperature forecasted by the proposed model to replace the actual temperature. Experiments using both measures showed that the proposed model has strong robustness against great uncertainty of temperature.

The comparison of training time and forecasting performance among various models together with the forecasting performance is also implemented.

1.2. Structure

The remainder of this article is organized as follows. In Section 2, we present the framework of the proposed model based on an ensemble GoogLeNet with dense block structure and modified deep residual neural networks. The input variable selection and Pearson correlation coefficient are also provided. In Section 3, the experiments of the STLF conducted by adopting the proposed model are illustrated. We then analyze the test results compared with other existing state-of-the-art methods. Section 4 concludes the paper.

2. Proposed Hybrid STLF Method

In this work, an hourly load forecasting model is proposed based on ensemble GoogLeNet with dense block structure and deep residual neural networks. The fundamental structure of subsections of the hybrid model is presented first. The inputs of the overall model are selected by correlation analysis and then processed prior to the introduction of elementary structure. The preliminary forecast produced by ensemble GoogLeNet is then passed through a modified deep residual neural network to improve prediction accuracy. During the training process, we find that some modifications, like learning rate, etc., should be made to enhance the model's learning power. The implementation and evaluation indices of probabilistic load forecasting by MC dropout are also presented.

2.1. Model Input Selection

The aim of this part is to appropriately select the historical time range of inputs and to reduce the computation amount as much as possible. The dimension of inputs rests with its historical time range and directly determines the complexity of the matrix produced during the computation of the network. The Pearson correlation coefficient, which denotes the linear correlation relationship between two time series, is applied to select historical data width.

If the linear correlation relationship between X and Y is strong, then the Pearson correlation coefficient is very close to 1. Sliding window technology is utilized to obtain time-lag data. Figure 1 shows the Pearson correlation coefficients between load data of different time lags and Figure 2 shows the Pearson correlation coefficients between load and lagged temperature data of different time lags. The raw hourly load and temperature data used in Figure 1 and Figure 2 are from ISO New England (ISO-NE) data [30]. As seen from Figure 1, the periodicities in the raw load data are 24 h, seven days, 52 weeks, and 12 months, according to different time lags. Larger time lags can reveal more information existing in input time series, but more time and computation are required. To avoid parameter explosion resulting from too many inputs and to preserve the most influential information, only the previous 48 h load data are taken into consideration. Obviously, temperature is a significant and influential external variable while the linear correlation relationship between load and temperature data is weak, as can be seen from Figure 2, which shows that the coefficient is less than 0.3. Nevertheless, the high temperature in summer and low temperature in winter contribute to great increases in electricity demand of air conditioners, which accounts for a large proportion of energy consumption. As can be seen from Figure 3, the strong nonlinear correlation between daily load and daily temperature is revealed; thus, the previous 48 h temperature data are also used to tune the prediction results. Note that the size of these input time variables is small.

Figure 1. Pearson correlation coefficient of load data corresponding to hour, day, week, month, and year.

Figure 2. Pearson correlation coefficient between load and lagged temperature data corresponding to hour, day, week, month, and year.

Figure 3. Nonlinear correlation with respect to load, temperature, and hour.

Additionally, the objective information concerning time, including weekday, weekend, season, holiday and nonholiday, is turned into one-hot codes which are added as binary inputs to assist the model in extracting the periodic and unusual time features of electric demand time series.

Linear interpolation is applied in the preprocessing stage to cope with missing data. The normalization of the data is conducted as given in the following formula:

$$Load_{Norm} = Load_{act} / Load_{max} \quad (1)$$

$$T_{Norm} = T_{act} / 100 \quad (2)$$

where $Load_{Norm}$ and T_{Norm} denote normalized load and temperature data, respectively. $Load_{act}$ and T_{act} denote actual load and temperature data, respectively, and $Load_{max} = \max_i Load_i$.

2.2. GoogLeNet Architecture

A great amount of improved network structure with respect to image processing was developed after 2012 when AlexNet was first introduced to implement tasks of image classification. The original intention of GoogLeNet was to dedicate it to solving two problems: (1) overfitting in the circumstance of too many parameters, and (2) increased calculation [8] due to a sharply increasing number of layers.

GoogLeNet consists of two different layers. First, the convolution layer plays a critical role in extracting various features from the input data. The convolution layer sustains the spatial relationship hidden in the data by acquiring data features by calculating the tiny space of input data. Each input datum is viewed as an observed matrix. The “tiny space” directly depends on the filter, which is also a matrix that is implemented on input data. The output of the convolution layer is a matrix, namely a feature map, arranged by sliding the filter through the entire input matrix and computing the values at each corresponding window. After the convolution operation, a padding operation is also needed for successful concatenation. Second, the pooling layer sustains the most favorable features while decreasing the dimension of input representations. The pooling layer is divided into three types: max pooling, average pooling and sum pooling. In this work, we adopt max pooling, where the largest value is taken in each sliding window. The pooling operation can make the input of the pooling layer smaller and the number of parameters less, reduce the amount of calculation, and improve the overfitting problem.

The nuclear idea of GoogLeNet lies in the inception mechanism, which is adopted through combining with bidirectional LSTM to predict 48 points of the next day [31]. First, the current simple representative of the inception module is limited to three filter sizes, namely 1×1 , 3×3 , and 5×5 ; this criterion was set, by default, out of commodiousness rather than requirement. In addition to the convolution layer, there is a max pooling layer with filter size 3×3 in a simple inception module as it is essential for the success of the current convolutional network, as seen in Figure 4a. The input image can be processed in parallel with these different operations. The suggested module is then an integration of all previous layers by concatenating their output into a separate output vector as the input of the subsequent layer.

Second, 1×1 convolutions brilliantly reduce the dimension wherever the computational requirements become greater. This idea comes from the success of embeddings, which mean 1×1 convolutions are used to reduce computation before 3×3 and 5×5 convolution layers. In addition to the dimension reductions, the adoption of rectified linear activation ReLU has also improved the performance of the inception module. Specifically, the ReLU has the form of Equation (3), where x_i is the input of the i -th node of a layer. The modified inception module is depicted in Figure 4b. The inputs of the inception module derive from the preprocessed data, as described in Section 2.1.

$$\text{ReLU}(x_i) = \max(x_i) \quad (3)$$

Instead of simply stacking the inception module layer by layer, we modified the connection between different inception modules based on the structure of dense block. Dense blocks exist in the DenseNet structure. The advantages of this structure involve alleviating gradient vanishing and strengthening the transmission of features. In traditional CNN, the number of layers is equal to the number of connections. Nevertheless, the number of connections in DenseNet is $L(L + 1)/2$, where L denotes the number of layers. Generally, the inputs of the layer come from the concatenation of the outputs of all the previous layers. A transition layer between different inception modules is adopted in case of increasing features with more inception modules.

(a) Simple inception module

(b) Inception module with dimension reduction

Figure 4. Inception module.

In the training period of a random neural network, the parameters of each layer will continue to change with the iteration processes, which will result in the continuous change in the input data distribution of the next layer, which creates the problem of internal covariate shift. The distribution of different input data requires adjustment of the learning rate and fine initialization weight parameters during the iterative process. To address this problem, batch normalization is adopted to normalize the output of the previous layer. This practice is likely to accelerate the training process. Nevertheless, BN's error is very sensitive to the batch size. Group Normalization (GN) is adopted due to its independence of batch size and satisfactory accuracy. The data can be converted into data with zero mean and unit variance by GN. The fundamental conversion formula of normalization methods, including BN and GN, is obtained by:

$$\mathbf{X}'_i = \frac{\mathbf{X}_i - \mu_i}{\sigma_i} \quad (4)$$

where $\mathbf{X}'_i$ denotes the normalized output of previous layer, μ_i represents the mean value of vector $\mathbf{X}_i$, and σ_i stands for the standard deviation (std) of vector $\mathbf{X}_i$. The details of computing μ_i and σ_i is illustrated in Appendix A.

2.3. Deep Residual Neural Network

The layer number fallback mechanism is implemented by the residual neural network shown in Figure 5a, which adds a shortcut between the input and output of the network and improves the prediction accuracy of the overall deep learning model. As shown in Figure 5a, a mapping which is from X to $\mathcal{H}(X)$ is learned in place of training a map from X

to $\mathcal{G}(X)$. The overall formula of the residual block turns into Equation (5). Note that $\mathcal{G}(X)$ and X must possess the same dimension.

$$\mathcal{H}(X) = \mathcal{G}(X) + X \quad (5)$$

Figure 5. (a) The residual block (ReLU is adopted as an activation function in each hidden layer). (b) RoR-3 has a three-level shortcut.

A deep residual neural network could be readily achieved through stacking ample residual blocks. Simply stacking residual blocks, however, inevitably limits its optimization ability. Thus, residual networks of residual networks (RoR) [32], a novel residual network structure, is proposed to improve the optimization ability of residual networks. A three-level residual network, RoR-3, illustrated in Figure 5b, contains a shortcut above all residual blocks, a shortcut above each residual block group which holds two residual blocks, and a shortcut in the residual block; respectively, these are referred to as a first-level shortcut, a second-level shortcut, and an inner-level shortcut. Specifically, if the layer in the first-level shortcut is set to 1 and K residual block groups are stacked, the forward propagation can be described by

$$X_K = X_0 + \mathcal{F}(X_{K-1}) \quad (6)$$

where X_0 is regarded as the input source of the residual neural network and the i -th residual block group yields X_i . The total loss of the residual network back propagated to X_0 can be obtained by

$$\frac{\partial Q}{\partial X_0} = \frac{\partial Q}{\partial X_K} \left[1 + \frac{\partial \mathcal{F}(X_{K-1})}{\partial X_0} \right] \quad (7)$$

where Q denotes the total loss of the RoR network. The “1” in Formula (7) implies that the gradient at the final layer can be straight back propagated to the input layer. This indicates that the gradient vanishing problem likely can be alleviated.

2.4. Proposed Model Structure

The diagram of the hybrid model proposed in this work can be seen in Figure 6. As shown, we use the hourly electrical load data and one-hot codes together with hourly temperature data as the input data sources of our model.

Figure 6. The structure of the proposed model.

First, preprocessing is conducted on the electrical load data and temperature data, which may have a singular value due to measurement error of the electrical device. In the next stage, the corrected data set is normalized to facilitate the learning model.

Second, loads and temperature values of the most recent 48 h are delivered to the ensemble GoogLeNet. The input of the modified GoogLeNet unit is the concatenation of output of all previously modified GoogLeNet units and input of the ensemble GoogLeNet. The output of the ensemble GoogLeNet with dense block structure is concatenated with the one-hot codes of weekday, weekend, season, holiday, and nonholiday distinction as input to go across the fully connected (FC) layers.

Last, the output of the FC layers is viewed as the input of the deep residual networks that contain the right and left columns of residual blocks. The input of the right column of residual block groups is what the FC layers yield (except for the second one in the right column of residual block groups, whose input is connected with the first blue spot in the

left column). What the residual block in the same layer yields is averaged (represented by gray spots on the right). The output of the gray spots is attached to all blue spots in following layers. The input of the first residual block of the left column is also what the FC layers yield. Beginning from the second layer, what imports the residual block in the left column is acquired via averaging all links from the gray spots with the connection from the output of the FC layers. What the modified deep residual network yields is considered as predicted hourly load. It is believed that the added column of residual block groups together with the shortcut connections can promote forecasting ability and alleviate the gradient vanishing problem of back-propagated error.

2.5. Probabilistic Load Forecasting

Probabilistic load forecasting is likely to have more practical significance than point forecasting. Due to the uncertainties of both model and data, the proposed model has potential to be used for probabilistic load forecasting. In this work, MC dropout [33] is adopted to quantify the prediction uncertainty. Generally speaking, dropout is a technique that randomly drops out units with probability p in a neural network. In practice, this is comparable to performing T stochastic forward passes through the network and averaging the results. We use the training dataset containing x' and y' to acquire the hidden knowledge in $f^{\mathcal{W}}(\cdot)$, an unknown neural network with parameters $\mathcal{W}$ to be trained. The prediction uncertainty [19] can be quantified as follows:

$$\begin{aligned}\text{Var}(y'|x') &= \text{Var}[\mathbb{E}(y'|\mathcal{W}, x')] + \mathbb{E}[\text{Var}(y'|\mathcal{W}, x')] \\ &= \text{Var}[f^{\mathcal{W}}(x')] + \sigma^2 \\ &= \frac{1}{M} \sum_{m=1}^M (\hat{y}'_m - \mu_y)^2 + \sigma^2\end{aligned}\quad (8)$$

where M denotes the test times of dropout, $\hat{y}'_m$ is the m -th output that we observe, μ_y denotes the mean of all M outputs of test dataset corresponding to each hour and $\mathcal{W}$ is the trained parameters. The first equation in (8) derives from canonical conditional variance formula. The other term σ^2 measures the inherent noise existing in dataset which can be estimated by a validation dataset.

After obtaining the prediction uncertainties, the comprehensive standard variation is obtained:

$$\text{Var}(y'|x') \approx \sqrt{\alpha^2 * \text{Var}(f^{\mathcal{W}}(x')) + \beta^2 * \sigma^2} \quad (9)$$

where α and β are parameters to be estimated; α represents the ratio that model uncertainty occupies the comprehensive standard variation while β indicates the percentage that data uncertainty occupies the comprehensive standard variation. The details of computation for $\text{Var}(f^{\mathcal{W}}(x'))$ and σ^2 are illustrated in Appendix B.

When coping with various methods with equally accurate degrees of coverage, we prefer to select the method that yields the tightest intervals. The generally acknowledged score formula proposed by Winkler, namely the Winkler score [34], allows for jointly estimating the coverage and interval width. For a central $(1 - \alpha) \times 100\%$ prediction interval, the Winkler score is defined as follows:

$$\text{Winkler - score} = \begin{cases} \delta_t, & L_t \leq y_t \leq U_t \\ \delta_t + \frac{2(L_t - y_t)}{\alpha}, & y_t < L_t \\ \delta_t + \frac{2(y_t - U_t)}{\alpha}, & y_t > U_t \end{cases} \quad (10)$$

where L and U denote lower and upper bounds, respectively, and y_t is the actual value at time t . The parameter δ_t represents the variance between L_t and U_t . Obviously, α is similar to confidence level.

The pinball loss function [34] is an error estimation for quantile forecasts. The pinball loss is obtained as:

$$\text{Pinball-loss}(y'_{t,q}, y_t, q) = \begin{cases} (1-q)(y'_{t,q} - y_t), & y_t > y'_{t,q} \\ q(y_t - y'_{t,q}), & y_t < y'_{t,q} \end{cases} \quad (11)$$

where y_t is the actual load at time t , q denotes quantiles ($q = 1, 2, \dots, 99$), and $y'_{t,q}$ is the quantile forecast at the q -th quantile. We can secure the pinball loss of homologous probabilistic load forecasts by summing up the pinball loss of each quantile forecast.

3. Results and Discussion

The models in each experiment are trained by the Adam optimizer with default parameters as mentioned in [35]. The models are accomplished by adopting Keras 2.2.4 with Tensorflow 1.11.0 as backend in the Python 3.6 environment. Note that adaptive adjustment of the learning rate during the training process is used. Two different public data sets are utilized, both of which contain hourly load and temperature data. The first data set uses the ISO-NE dataset [30] from New England. The time range of the first dataset is from March 2003 to December 2014. The second dataset is the North American Utility dataset [36]. The time scope of the second dataset is from 1 January 1985 to 12 October 1992.

3.1. Comparison Criteria

Mean absolute percentage error (MAPE), root mean square error (RMSE) [2], R-Square and mean absolute error (MAE) are among the most significant indices to apply when comparing the results of various models in STLF issues. They are described as follows:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left(\left| \frac{\text{actual}(i) - \text{predicted}(i)}{\text{actual}(i)} \right| \times 100 \right) \quad (12)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\text{actual}(i) - \text{predicted}(i))^2} \quad (13)$$

$$\text{R-Square} = 1 - \frac{\sum_i (\text{actual}(i) - \text{predicted}(i))^2}{\sum_i (\overline{\text{actual}} - \text{actual}(i))^2} \quad (14)$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\text{actual}(i) - \text{predicted}(i)| \quad (15)$$

where $\overline{\text{actual}}$ denotes the average of the actual time series.

3.2. Experiment 1: Parameter of Filters Decision

Two inception modules are adopted in the ensemble GoogLeNet and the filter size consists of 1×1 , 3×3 , and 5×5 . The ISO-NE data in 2014 are used as the test set and the previous five-year data is used as the training set. Ten percent of the training data set is used as a validation set to tune the hyperparameters. The North American data of the one-year period before 12 October 1992 is utilized as the test set and the data ranging from 12 October 1986 to 12 October 1991 is treated as the training set. Ten percent of the training data set is also used as a validation set to tune the hyperparameters. The magnitude of load data in the ISO-NE data set and the North American data set is 1×10^4 and 1×10^3 respectively. The inputs of both tests contain two-day load and temperature data. We change the learning rate of the first training phase from 0.001 to 0.0012 on the second test of the North American dataset. The rest of the hyperparameters are unchanged except for the parameter of filters, which exists in the layer of Conv1D. The parameter of filters denotes the output dimension of Conv1D.

As is illustrated in Table 1, the method used is susceptible to the parameter of filters in

each convolutional layer. As an example, with 64 filters, the MAPE in the North American test set is 1.41 for year 1992, which has a deviation of 0.84 from the results when the number of filters is 128. The MAPE of the ISO-NE dataset is the lowest when the number of filters is set 128. Figure 7 presents the illustration of the prediction results for two different weeks in 2014 for the ISO-NE data set. The larger parameter of filters in each convolutional layer, then the more trainable parameters there are more likely to extract more features hidden in the dataset. Once the parameter of parameters exceeds a potential number, however, overfitting is easy to occur and the performance on the test set worsens.

Table 1. MAPE, RMSE, and number of trainable parameters in two datasets corresponding to different parameter of filters.

Parameter of Filters		32	48	64	128
Trainable parameters		74,965	83,733	92,501	127,573
ISO-NE	MAPE	1.95	1.85	1.49	1.32
	RMSE	376	346	336	299
North American	MAPE	1.93	1.82	1.41	2.25
	RMSE	54	50	43	59

Figure 7. Actual and predicted load for a summer week (**left**) and a winter week (**right**) of 2014 for the ISO-NE dataset; 12 August 2014 and 12 January 2014 begin the separate two-week periods.

As seen from Table 1, the prediction performance gets worse when the parameter of filters becomes 128 for North American Utility data set. The results also show that the proposed hybrid model with ensemble GoogLeNet and deep residual networks has strong generalization ability. Consequently, the parameter of filters for the ISO-NE dataset is 128; for the North American Utility dataset it is 64. The subsequent experiments are based on this fundamental criterion.

3.3. Experiment 2: Validation of the Integrated Method

To demonstrate the effectiveness of the integrated method, we compared the performance of the proposed method with typical GoogLeNet and ResNetPlus [19] model.

The ISO-NE data is split into training set, validation set and test set. Five-year data before test data is used to train the three diverse models. Ten percent of the training data set is adopted as a validation set to adjust the hyperparameters. We tested the three models on the test set containing the data in 2014 and achieved the results of each model depicted in Table 2. The proposed model shows the best fitting ability and reaches the highest R-Square, namely 0.9898, which represents the fitting degree of a model. Meanwhile, the raised model obtains lower values of MAE (168.47 MW) and RMSE (271.23 MW) than the other two basic models. It is clear that the MAPE of the proposed model is reduced by 0.23% with respect to the typical GoogLeNet model.

Table 2. Yearly R-Square, MAE, RMSE, and MAPE of proposed model compared with basic model.

Models	R-Square	MAE/MW	RMSE/MW	MAPE/%
GoogLeNet	0.9861	218.57	315.81	1.55
ResNetPlus [19]	0.9855	216.58	313.8	1.52
Proposed model	0.9898	168.47	271.23	1.32

Additionally, the monthly comparison of R-Square, MAE, RMSE and MAPE for typical GoogLeNet, ResNetPlus [19] and the proposed model is exemplified in Table 3. Specifically, the average value of R-Square for ResNetPlus [19] is 0.971, while the proposed model increases by 0.014. Similarly, the MAE and RMSE of the proposed model fluctuate from 139.1 to 228.9 and from 189.3 to 546.3 respectively. The upper and lower bounds of the fluctuation intervals is lower than typical GoogLeNet and ResNetPlus. Besides, the proposed method provides the lowest MAPE among the tested models. Therefore, the integration of ensemble GoogLeNet and modified residual network is shown to have more effectiveness in STLF tasks.

Table 3. Monthly R-Square, MAE, RMSE and MAPE for typical GoogLeNet, ResNetPlus and proposed model.

Month	R-Square			MAE/MW			RMSE/MW			MAPE/%		
	GoogLeNet	ResNetPlus [19]	Proposed Model	GoogLeNet	ResNetPlus [19]	Proposed Model	GoogLeNet	ResNetPlus [19]	Proposed Model	GoogLeNet	ResNetPlus [19]	Proposed Model
Jan	0.972	0.962	0.981	302.4	305.2	228.9	388.5	387.2	319.6	1.85	1.72	1.47
Feb	0.979	0.971	0.988	237.2	239.4	162.9	298.1	280.5	227.1	1.56	1.58	1.14
Mar	0.968	0.952	0.988	289.8	295.6	151.0	352.4	357.3	219.5	2.01	1.96	1.13
Apr	0.987	0.981	0.989	155.5	158.3	139.1	200.1	196.4	189.3	1.21	1.23	1.21
May	0.985	0.986	0.988	196.4	200.1	150.6	241.6	242.9	211.3	1.63	1.61	1.39
Jun	0.993	0.973	0.994	196.1	199.8	162.4	240.9	229.8	214.5	1.47	1.49	1.31
Jul	0.994	0.986	0.993	214.6	210.3	214.7	269.3	264.9	273.8	1.32	1.33	1.45
Aug	0.994	0.983	0.994	188.8	186.3	172.5	232.1	238.0	216.6	1.30	1.34	1.30
Sep	0.993	0.978	0.994	202.8	199.6	171.9	254.1	250.1	232.3	1.50	1.40	1.38
Oct	0.990	0.982	0.990	165.9	164.2	144.6	209.0	210.9	202.7	1.35	1.32	1.30
Nov	0.912	0.921	0.928	253.9	252.8	176.8	605.6	593.9	546.3	1.86	1.79	1.43
Dec	0.984	0.980	0.992	219.1	218.7	145.0	279.5	285.2	203.5	1.50	1.44	1.11
Average	0.979	0.971	0.985	218.5	219.2	168.4	297.6	294.8	254.7	1.55	1.52	1.302

In this experiment, it is obvious that the typical GoogLeNet has considerable performance with regards to ResNetPlus. The features extracted from the input matrix through GoogLeNet, which consists of convolutional layers, represent temporal and spatial relationships hidden in the load and temperature time series. The hidden temporal and spatial relationships together with other temporal information are viewed as inputs of deep residual networks which enhance the learning ability of the architecture. Therefore, the better performance presented clarifies the superiority of the integrated model.

3.4. Experiment 3: Comparison with Existing Models

Test 1: In this test, ANN + RF [37], HEFM [37], and LWSVR-MGOA [38] are taken into consideration for comparison with the proposed methodology. To evaluate the validity of the proposed model, the time range of historical data is from January 2013 to July 2015 for the ISO-NE dataset. In this test, a winter month (January 2015) and a summer month (July 2015) are tested. The results in Table 4 imply that the MAPEs of the proposed model decrease by 19.28% and 17.17% in winter months and summer months, respectively. It is

notable that the load in the summer months is difficult to forecast due to the effect of other meteorological factors, such as humidity and wind speed.

Table 4. Comparison to the existing methods—MAPE for load in the winter months and summer months of the ISO-NE dataset.

	MAPE (Winter Month)	MAPE (Summer Month)
ANN + RF [37]	3.1	2.95
HEFM [37]	1.95	2.81
LWSVR-MGOA [38]	1.67	2.26
Proposed model	1.348	1.872

Test 2: This test is to forecast daily loads of the ISO-NE dataset in the year 2006. The training set is from 1 March 2003 to 31 December 2005. The models of WT-ELM-MABC proposed in [39] and ResNetPlus proposed in [19] are also trained with data from 2003 to 2005. The MAPE of each month in 2006 for the three models are listed in Table 5. Nevertheless, the WT-ELM-MABC model produced better results than the proposed model for four months. However, the overall MAPE of the proposed model is reduced by 3.11% upon the dataset from ISO-NE.

Table 5. Comparison to the existing methods—MAPE for load in ISO-NE of 2006 using actual temperature.

Month	WT-ELM-MABC [39]	ResNetPlus [19]	Proposed Model
January	1.52	1.619	1.340
February	1.28	1.308	1.233
March	1.37	1.172	1.162
April	1.05	1.34	1.428
May	1.23	1.322	1.302
June	1.54	1.411	1.462
July	2.07	1.962	1.936
August	2.06	1.549	1.469
September	1.41	1.401	1.236
October	1.23	1.293	1.421
November	1.33	1.507	1.337
December	1.65	1.465	1.472
Average	1.48	1.447	1.402

3.5. Experiment 4: Robustness Estimation

Actual temperature data are used in the previous experiments. We utilize the North American Utility dataset in this experiment to validate the robustness of the proposed model. The division of the training set and test set is the same as mentioned in [8]. The time range of the data in this experiment is from 1 January 1988 to 12 October 1992. For the sake of evaluating the proposed model, the hourly loads for the two-year period, which is before 12 October 1992, have been predicted. The data prior to the current forecasting point is used for training. The validation dataset occupying 10% of the training set is used to adjust the hyperparameters of the proposed model. The Gaussian noise with 1 °F standard deviation is added to the measured temperature, as is done in [8,19,39,40] in order to obtain the predicted temperature. The comparison of the results is listed in Table 6. The results from Table 6 show that the MAPE of the proposed model is smaller than the other three available methods on the given test. The proposed strategy greatly reduces the MAPE by at least 4.91% and 6.34% in the two scenarios. Meanwhile, the MAPE of the proposed model increases by 0.315% when the actual temperature is replaced by the modified temperature.

Table 6. Comparison to the existing methods—MAPE for load in the North American Utility data set.

Model	Predictions Using Actual Temperature	Predictions Using Modified Temperature
ResNetPlus [19]	1.665	1.693
WT-ELM-MABC [39]	1.87	1.95
SSA-SVR [8]	1.99	2.03
WT-ELM-LM [40]	1.67	1.73
Proposed model	1.587	1.592
Improvement of proposed model	4.91%	6.34%

The actual temperature can also be replaced in another version. We use predicted temperature of the predicting time instead of actual temperature. One-hot codes for seasons, months, and hours are utilized as inputs to predict temperature. The proposed load forecasting model is simplified and modified slightly to predict temperature. The results of the temperature forecasting model are RMSE = 2.213 and MAPE = 3.723%, which are illustrated in Figure 8. The predicted temperature then replaces the actual load and MAPE for the load forecast is 1.589%, which is comparable to the results (1.587%) of the predictions using actual temperature. It is rational to come to the conclusion that the proposed method has strong robustness against temperature uncertainty.

Figure 8. Actual and predicted temperature for a summer week (left) and a winter week (right) of 1991 for the North American Utility dataset; 1 August 1991 and 12 January 1991 begin the separate two-week periods.

3.6. Experiment 5: Training Time Comparison

In this experiment, we train the model with ISO-NE data under the same circumstances where time span is six years, namely from 2004 to 2009. Various models listed in [41], including modified ELM, modified SVR, modified error correction (ErrCor), modified improved second order (ISO), and ResNetPlus [18] are compared with the proposed model. The training time results of the various state-of-the-art models are compared in Table 7. Meanwhile, we also take the MAPEs of the year 2010 and 2011 into consideration for these methods. The training time of the proposed method is reduced by 44.54% compared with the Modified ELM. The MAPEs of the proposed model are reduced by 3.07% and 7.38% for the year 2010 and 2011 respectively. The results showed that the proposed model not only needs less time to train, but also achieved better performance than the other five published models.

Table 7. Training time of various models.

Model	Training Time/Hour	MAPE/%	
		Predict 2010	Predict 2011
Modified ISO [41]	25.03	1.95	2.20
Modified ErrCor [41]	18.77	1.75	1.98
Modified SVR [41]	11.917	1.79	2.07
Modified ELM [41]	0.613	1.81	2.17
ResNetPlus [18]	1.50	1.50	1.64
Proposed model	0.34	1.454	1.519

3.7. Experiment 6: Probabilistic Load Forecasting

We used the ISO-NE dataset to illustrate the probabilistic load forecasting with MC dropout technology. The dropout rate ($p = 0.1$) is put into the previously proposed model, except for the inception module, input layer, and output layer. The parameters in (9) are estimated by the proposed hybrid model, which is trained via 30 epochs (with $M = 30$ for Formula (8)), and the estimated value of α is 0.78 and β is 0.47. An instance of the 95% coverage rate for two different weeks in 2014 is shown in Figure 9. The tested coverage rate results, generated by the proposed method with regards to various z-scores which stand for the number of standard variances, are listed in Table 8. The results show that the proposed model gives satisfactory tested coverages for different intervals.

Figure 9. Actual load and 95% prediction intervals of a summer week (beginning from 8.8) and a winter week (beginning from 1.9) for 2014 for the ISO-NE dataset.**Table 8.** Tested coverages of the proposed model with MC dropout.

Z-Score	Expected Coverage Rate	Tested Coverage Rate
1.000	68.27%	69.9%
1.280	80%	81.7%
1.645	90%	90.5%
1.960	95%	94.6%

In addition, as indicated in Formula (10), the Winkler scores in this experiment of the ISO-NE dataset are 16.7 ($\alpha = 0.5$) and 28.7 ($\alpha = 0.1$). From Formula (11), the pinball loss that we can obtain is 1.828. The results of both indices can be regarded as the benchmark performance of this dataset.

4. Conclusions

A novel STLF model based on ensemble GoogLeNet and modified deep residual neural networks was proposed in this paper. The ensemble GoogLeNet with dense block

structure and modified deep residual neural networks facilitated high accuracy of the proposed model and enabled suitable generalization ability. Various experiments conducted on two commonly accepted public datasets proved the validity of the methodology. The selection of the parameter “filters” were studied for the ISO-NE and North American Utility datasets. Comparisons with models in existence revealed that the proposed methodology outweighs other models in both prediction accuracy and robustness. Furthermore, the time spent in the training process was largely reduced compared with other models. It is also worth noting that the proposed methodology can be applied directly to probabilistic forecasting and achieve satisfactory results.

Much further work can be done. As we have only briefly considered deep neural networks, we may combine many other structures of deep neural networks (e.g., LSTM or GRU) with the model to strengthen its performance. Additionally, the implementation of probabilistic load forecasting can be further investigated. Moreover, more meteorological variables can be taken into consideration to improve the prediction accuracy.

Author Contributions: Conceptualization, A.D.; methodology, A.D.; software, A.D.; validation, A.D.; Formal analysis, A.D.; investigation, A.D.; resources, A.D. and X.Z.; data curation, X.Z.; writing—original draft preparation, A.D.; writing—review and editing, A.D.; visualization, A.D.; supervision, T.L.; project administration, X.Z.; funding acquisition, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available in <https://www.iso-ne.com/isoexpress/web/reports/load-and-demand> (accessed on 29 September 2021) and https://class.ece.uw.edu/555/el-sharkawi/index_files/Page3404.htm (accessed on 29 September 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Computing of μ_i and σ_i

To address the issue of internal covariate shift, we adopt GN layer which is independent of batch size. To explain the difference between the two methods, the feature map tensor is briefly illustrated in Figure A1 where (H, W) denote the spatial axes, namely height and width, C represents channel axis and N implies batch axis. μ_i and σ_i in Equation (4) can be obtained by:

$$\mu_i = \frac{1}{m} \sum_{k \in \mathcal{M}_i} x_k, \quad \sigma_i = \sqrt{\frac{1}{m} \sum_{k \in \mathcal{M}_i} (x_k - \mu_i)^2 + \delta} \quad (\text{A1})$$

where δ denotes a small constant, $\mathcal{M}_i$ represents the set of feature map whose size is m . The main difference between the two methods lies in the definition of set $\mathcal{M}_i$.

In BN, the set $\mathcal{M}_i$ is defined as:

$$\mathcal{M}_i = \{k | k_C = i_C\} \quad (\text{A2})$$

where k_C (and i_C) is the index of k (and i) along the C axis. This implies that the features, which have the same channel are normalized altogether. In other words, μ_i and σ_i are computed along the (N, H, W) axes.

In GN, the set $\mathcal{M}_i$ is defined as:

$$\mathcal{M}_i = \left\{ k \left| k_N = i_N, \left\lfloor \frac{k_C}{C/Z} \right\rfloor = \left\lfloor \frac{i_C}{C/Z} \right\rfloor \right. \right\} \quad (\text{A3})$$

where the hyper-parameter Z denotes the number of groups ($Z = 32$ or 64), C/Z represents the number of groups in each channel. $\lfloor \cdot \rfloor$ is floor operation. The second equation in (A3) means that the indices k and i share the same group of channels, presuming each group of channels are stored in a sequential order along the C axis. μ_i and σ_i are computed along a group of C/Z channels. As illustrated in Figure A1, a simplified case for GN has 3 groups

($Z = 3$) each of which owns 2 channels.

In both normalization methods, a single channel linear transformation is learned to make up for potential loss in describing capability:

$$y_i = \varsigma \hat{x}_i + \eta \quad (\text{A4})$$

where ς and η are trainable parameters of dilation and translation.

Once the $\mathcal{M}_i$ is determined by Equation (A3), we can define the GN layer by Equations (4), (A1) and (A4). Similarly, the features in the same group are normalized together by the same ς and η .

Figure A1. Two Normalization methods. The feature map tensor is shown in each subplot.

Appendix B. Computation of $\text{Var}(f^{\mathcal{W}}(x'))$ and σ^2

$\text{Var}(f^{\mathcal{W}}(x'))$ represents the model uncertainty caused by insufficient learning of the parameters $\mathcal{W}$. To alleviate the inevitable influence, dropout rate (p) is applied M times and the validation set contains d_1 days' datum. Then m -th overall outputs of the model are obtained as (A5) where $(1 - p)N_{\mathcal{W}}$ denotes the remaining nodes number of each layer.

$$\hat{y}'_{mo} = f_m^{\mathcal{W}}(x', (1 - p)N_{\mathcal{W}}) \quad (\text{A5})$$

We achieve the variance for j -th hour of i -th day as Equation (A6) where $\hat{y}'_{mij}$ (and $\hat{y}'_{moij}$) represents actual value (and predicted value) for j -th hour of i -th day at m -th for validation. Then we get the variance matrix of $[\text{Var}(f^{\mathcal{W}}(x'))]_{d_1 \times 24}$.

$$\text{Var}(f^{\mathcal{W}}(x'))_{ij} = \frac{1}{M} \sum_{m=1}^M (\hat{y}'_{mij} - \hat{y}'_{moij})^2 \quad (\text{A6})$$

To acquire σ^2 , select the first d_2 days from aforementioned d_1 days. The output of the trained model is $\hat{y}'_o$ as calculated in Equation (A7). The variance representing inherent noise at the time of daily j -th hour can be obtained in Equation (A8) where $\hat{y}'_{ij}$ (and $\hat{y}'_{oij}$) represents actual value (and predicted value) for j -th hour of i -th day. Therefore, the variance vector $[\sigma^2]_{24 \times 1}$ is acquired.

$$\hat{y}'_o = f^{\mathcal{W}}(x', (1 - p)N_{\mathcal{W}}) \quad (\text{A7})$$

$$\sigma_j^2 = \frac{1}{d_2} \sum_{i=1}^{d_2} (\hat{y}'_{ij} - \hat{y}'_{oij})^2 \quad (\text{A8})$$

To combine the aforementioned two variances, the formula of $[\text{Var}(y'|x')]_{d_1 \times 24}$ can be claimed as:

$$[\text{Var}(y'|x')]_{ij} = \alpha^2 \times \text{Var}(f^{\mathcal{W}}(x'))_{ij} + \beta^2 \times \sigma_j^2 \quad (\text{A9})$$

References

- Guo, J.; Xu, Y. Main wiring system. In *Code for Design of Photovoltaic Power Station*, 1st ed.; China Planning Press: Beijing, China, 2012; p. 32.
- Sadaei, H.; Silva, P.; Guimaraes, F.; Lee, M. Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series. *Energy* **2019**, *175*, 365–377. [\[CrossRef\]](#)
- Li, J.; Deng, D.; Zhao, J.; Cai, D.; Hu, W.; Zhang, M.; Huang, Q. A novel hybrid short-term load forecasting method of smart grid using MLR and LSTM neural network. *IEEE Trans. Ind. Informat.* **2021**, *17*, 2443–2452. [\[CrossRef\]](#)
- Eskandari, H.; Imani, M.; Moghaddam, M. Convolutional and recurrent neural network based model for short-term load forecasting. *Electr. Power Syst. Res.* **2021**, *195*, 107173. [\[CrossRef\]](#)
- Deihimi, A.; Showkati, H. Application of echo state networks in short-term electric load forecasting. *Energy* **2012**, *39*, 327–340. [\[CrossRef\]](#)
- Chitalia, G.; Pipattanasomporn, M.; Garg, V.; Rahman, S. Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks. *Appl. Energy* **2020**, *278*, 115410. [\[CrossRef\]](#)
- 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. [\[CrossRef\]](#)
- Ceperic, E.; Ceperic, V.; Baric, A. A strategy for short-term load forecasting by support vector regression machines. *IEEE Trans. Power Syst.* **2013**, *28*, 4356–4364. [\[CrossRef\]](#)
- Hu, Z.; Bao, Y.; Xiong, T. Comprehensive learning particle swarm optimization based memetic algorithm for model selection in short-term load forecasting using support vector regression. *Appl. Soft Comput.* **2014**, *25*, 15–25. [\[CrossRef\]](#)
- Imani, M. Long short-term memory network and support vector regression for electrical load forecasting. In Proceedings of the 5th International Conference on Power Generation Systems and Renewable Energy Technologies (PGSRET), Istanbul, Turkey, 26–27 August 2019; pp. 1–6.
- Hamed, 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.
- Jung, H.; Song, K.; Park, J.; Park, R. Very short-term electric load forecasting for real-time power system operation. *J. Electr. Eng. Technol.* **2018**, *13*, 1419–1424.
- Guan, C.; Luh, P.; Michel, L.; Wang, Y.; Friedland, P. Very short-term load forecasting: Wavelet neural networks with data pre-filtering. *IEEE Trans Power Syst.* **2013**, *28*, 30–41. [\[CrossRef\]](#)
- Kumar, S.; Mishra, S.; Gupta, S. Short term load forecasting using ANN and Multiple Linear Regression. In Proceedings of the 2nd International Conference on Computational Intelligence & Communication Technology, Ghaziabad, India, 12–13 February 2016; pp. 184–186.
- Mi, J.; Fan, L.; Duan, X. Short-Term Power Load Forecasting Method Based on Improved Exponential Smoothing Grey Model. *Math. Probl. Eng.* **2018**, *2018*, 3894723. [\[CrossRef\]](#)
- Sharma, S.; Majumdar, A.; Elvira, V.; Chouzenoux, É. Blind Kalman filtering for short-term load forecasting. *Electr. Power Syst. Res. IEEE Trans. Power Syst.* **2020**, *35*, 4916–4919. [\[CrossRef\]](#)
- Wu, F.; Cattani, C.; Song, W.; Zio, E. Fractional ARIMA with an improved cuckoo search optimization for the efficient Short-term power load forecasting. *Alex. Eng. J.* **2020**, *5*, 3111–3118. [\[CrossRef\]](#)
- Liu, Y.; He, G.; Tan, M.; Nie, F.; Li, B. Artificial neural network model for turbulence promoter-assisted crossflow microfiltration of particulate suspensions. *Desalination* **2014**, *338*, 57–64. [\[CrossRef\]](#)
- Chen, K.; Chen, K.; Wang, Q.; He, Z.; Hu, J.; He, J. Short-term load forecasting with deep residual networks. *IEEE Trans. Smart Grid* **2019**, *10*, 2943–2952. [\[CrossRef\]](#)
- Napoli, C.; Pappalardo, G.; Tina, G.; Tramontana, E. Cooperative strategy for optimal management of smart grids by wavelet RNNs and cloud computing. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 1672–1685. [\[CrossRef\]](#)
- Deng, D.; Li, J.; Zhang, Z.; Teng, Y.; Huang, Q. Short-term electric load forecasting based on EEMD-GRU-MLR. *Power Syst. Technol.* **2020**, *44*, 593–602.
- He, F.; Zhou, J.; Feng, Z.; Liu, G.; Yang, Y. A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with Bayesian optimization algorithm. *Appl. Energy* **2019**, *237*, 103–116. [\[CrossRef\]](#)
- El-Hendawi, M.; Wang, Z. An ensemble method of full wavelet packet transform and neural network for short term electrical load forecasting. *Electr. Power Syst. Res.* **2020**, *182*, 106265. [\[CrossRef\]](#)
- Li, C. A fuzzy theory-based machine learning method for workdays and weekends short-term load forecasting. *Energy Build.* **2021**, *245*, 111072. [\[CrossRef\]](#)
- Kim, T.; Cho, S. Predicting residential energy consumption using CNN-LSTM neural networks. *Energy* **2019**, *182*, 72–81. [\[CrossRef\]](#)
- Huang, Q.; Li, J.; Zhu, M. An improved convolutional neural network with load range discretization for probabilistic load forecasting. *Energy* **2020**, *203*, 117902. [\[CrossRef\]](#)
- Liao, Z.; Pan, H.; Fan, X. Multiple wavelet convolutional neural network for short term load forecasting. *IEEE Internet Things J.* **2021**, *8*, 9730–9739. [\[CrossRef\]](#)

28. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
29. Teerakawanich, N.; Leelaruij, T.; Pichetjamroen, A. Short term prediction of sun coverage using optical flow with GoogLeNet. *Energy Rep.* **2020**, *6*, 526–531. [[CrossRef](#)]
30. ISO New England Data. Available online: <https://www.iso-ne.com/isoexpress/web/reports/load-and-demand> (accessed on 12 July 2021).
31. Kim, J.; Moon, J.; Hwang, E.; Kang, P. Recurrent inception convolution neural network for multi short-term load forecasting. *Energy Build.* **2019**, *194*, 328–341. [[CrossRef](#)]
32. Zhang, K.; Sun, M.; Han, T.; Yuan, X.; Guo, L.; Liu, T. Residual networks of residual networks: Multilevel residual networks. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 1303–1314. [[CrossRef](#)]
33. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the 33rd International Conference on Machine Learning, Boston, MA, USA, 19–24 June 2016; Volume 48, pp. 1050–1059.
34. Liu, B.; Nowotarski, J.; Hong, T.; Weron, R. Probabilistic load forecasting via quantile regression averaging on sister forecasts. *IEEE Trans. Smart Grid* **2017**, *18*, 730–737. [[CrossRef](#)]
35. Kingma, D.; Ba, L. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
36. North-American Utility Dataset. Available online: https://class.ece.uw.edu/555/el-sharkawi/index_files/Page3404.htm (accessed on 12 July 2021).
37. Zhou, M.; Jin, M. Holographic ensemble forecasting method for short-term power load. *IEEE Trans. Smart Grid.* **2019**, *10*, 425–434. [[CrossRef](#)]
38. Elattar, E.; Sabiha, N.; Alsharef, M.; Metwaly, M. Short term electric load forecasting using hybrid algorithm for smart cities. *Appl. Intell.* **2020**, *50*, 3379–3399. [[CrossRef](#)]
39. Li, S.; Wang, P.; Goel, L. Short-term load forecasting by wavelet transform and evolutionary extreme learning machine. *Electr. Power Syst. Res.* **2015**, *122*, 96–103. [[CrossRef](#)]
40. Li, S.; Wang, P.; Goel, L. A Novel Wavelet-Based Ensemble Method for Short-Term Load Forecasting with Hybrid Neural Networks and Feature Selection. *IEEE Trans. Power Syst.* **2016**, *31*, 1788–1798. [[CrossRef](#)]
41. Cecati, C.; Kolbusz, J.; Rózycki, P.; Siano, P.; Wilamowski, B. A novel RBF training algorithm for short-term electric load forecasting and comparative studies. *IEEE Trans. Ind. Electron.* **2015**, *62*, 6519–6529. [[CrossRef](#)]