Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Heating Load Forecasting for Combined Heat and Power Plants via Strand-Based LSTM

Junyu Liu*[1], Xiao Wang*[1], Yan Zhao*[2], Bin Dong[3], Kuan Lu[2], Ranran Wang[4]

[1]Academy of Advanced Interdisciplinary Studies, Peking University, Beijing 100871, China
[2]State Grid Shandong Electric Power Research Institute, Jinan 250001, China
[3]Beijing International Center for Mathematical Research, Peking University, Beijing 100871, China
[4]Beijing Institute of Big Data Research, Beijing 100871, China
*Joint first author.

Corresponding authors: Ranran Wang (ranranw@bibdr.org), Kuan Lu (lk83@163.com), Bin Dong (dongbin@math.pku.edu.cn).

Bin Dong is supported in part by Beijing Natural Science Foundation (Z180001) and NSFC grant 11831002.

ABSTRACT Heating load forecasting is the premise for guiding heating operation management and dispatching. Heating load forecasting is a time series prediction problem which requires us to predict the real-time heating loads in the next 24 hours using available historical records and weather information. In this paper, we propose a model for short-term heating load forecasting based on a properly designed strand-based long short term memory (LSTM) recurrent neural network. We present how the data are pre-processed, and the loss function is designed to improve the model's performance. Furthermore, an ensemble strategy is incorporated with the LSTM model to enhance its generalization and robustness. On offline (historical) testing data, the proposed model performs satisfactory predictions which meet the requirements of the local power plant. In addition to offline tests, we also implement the model to an online system of a power plant in Shandong province, China. The model made continuous forecasting without human interference for four months during the heating season of 2018. The model reported satisfactory online testing results that were comparable with the offline experiments using historical data.

INDEX TERMS Deep learning, load forecasting, recurrent neural networks, time series.

I. INTRODUCTION

Load forecasting is essential for planning and optimizing operations for large energy systems. It can assist in a reasonable dispatch of resources. Because of the significant economic and environmental benefits, short-term forecasting has been widely used in energy fields, such as electric load forecasting [1], heating load forecasting [2][3], etc. This paper is focused on heating load forecasting.

Heating load forecasting is the premise for accurate guidance of heating operation management and heat dispatching. During the heating season in China, which usually starts in the middle of November and ends in March of next year, an accurate heating load forecast can significantly improve the stability and production efficiency of central heating systems. On the one hand, heating load forecasting is essential to the stability of the power system. Renewable energy, such as wind power, is severely dependent on weather and thus unstable. Therefore, power generation of other power sources (mainly thermal power) should be adjusted to maintain the stability of the power grid. Furthermore, the ability of the thermal units to adjust power generation can be predicted if the heating demand is accurately forecasted. On the other hand, heating load forecasting is vital for power plants to work economically and efficiently. Based on short-term heating load forecasting, power plants can make detailed schedules to meet the heating demand. The resources, such as fuels and human resources, can be reasonably allocated. Accurate forecasts enable power plants to receive excellent control of thermal units and reduce operating costs, which brings substantial economic benefits for heating systems.

Data studied in this research are the historical data provided by the Electric Power Academy of Shandong Province. The data set is composed of weather information and the heating load, both of which are sampled every ten minutes. Our objective is to perform next day forecasting of the heating load based on historical weather and heating load

data, as well as the weather forecasting data of the next day.

Most simply, we can regard the problem as a static regression problem. To be more precise, we can learn a regression model that approximates the mapping between weather information and heating load at each given time instance. Traditional machine learning methods have been applied to short-term load forecasting problems, such as linear regression [4], Support Vector Machine (SVM), [5][6][7], artificial neural networks (ANNs) [8][9] and some modified machine learning methods such as SSA-SVR [10][11]. However, Hippert et al. [12] suggested that most of the works based on ANNs had overfitted the data after examining a collection of ANN-related papers. Moreover, most of the studies about heating load forecasting were focused on machine learning methods. In [13], authors found that bagging trees, boosting trees and neural networks are efficient models and they outperformed multivariate linear regression method. In [14], authors proposed a hybrid model which combines multivariate adaptive regression splines (MARS) and extreme learning machine (ELM). In [15], authors found that the 4-layer neural network outperformed ridge regression, Lasso and gradient boosting methods.

Despite the popularity of the regression methods above, they do not work well for our problem, as will be demonstrated in Section IV. This is mainly caused by the negligence of the temporal dependence of the data. Therefore, time series analysis models are more suitable. Popular time series analysis models include the automatic regressive moving average (ARMA) [16][17][18][19] and the autoregressive integrated moving average (ARIMA) models [20][21][22]. However, these methods require either the time series to be stationary or its difference sequence to be stationary. Furthermore, these methods mostly do not utilize long-term historical information of the time series data.

To resolve the aforementioned problem, we consider the Long Short Term Memory (LSTM) network [23], which is effective on non-stationary time series data and capable of incorporating long-term dependencies in the data. LSTM is a Recurrent Neural Network (RNN) with a special structure. Since its repeating unit is composed of four gates (input gate, output gate, cell gate and forget gate), it can automatically select useful long-term information, discard irrelevant information, and incorporate short-term information. Consequently, LSTM works well on a large variety of problems, especially processing and analyzing non-stationary time series data, such as processing natural languages [24], machine translation [25][26], speech recognition [27], etc. Although LSTM model is popular in the fields above, to our best knowledge, it is new to heating load forecasting. In [28] , authors compared the performance of nonlinear autoregressive exogenous RNN and SVM, and found RNN yields higher accuracy than SVM. However, the performance of LSTM in heating load forecasting is still waiting for exploration.

This paper is focused on developing a 24-hours ahead heating load forecasting model based on LSTM. As will be shown in Section IV that a direct application of LSTM (called base LSTM) is not ideal for our problem, and modifications have to be made. Furthermore, to make the forecasting more stable, an ensemble strategy is proposed. Meanwhile, given that the heating load contain large fluctuates especially at the end of the heating season, we introduce proper smoothing and local scaling to solve such issues.

TABLE 1. Means (and standard deviations) of the measured weather data of the two cities.

Mean(std)	City A	City B
Temperature	3.932 (5.459)	2.071 (6.060)
Pressure	1015.233 (6.118)	1012.390 (6.093)
Humidity	53.730 (22.432)	55.150 (23.382)
WindSpeed	2.195 (1.523)	1.247 (0.917)

The paper is organized as follows. We provide detailed descriptions of our datasets in Section II. Section III presents the proposed LSTM model. Section IV presents experimental details and results. We conclude this paper in Section V.

II. DATA

A. DATA SOURCE

The data are provided by the Electric Power Academy of Shandong Province in China, and they are historical data of City A and B, two cities in Shandong Province. The data are composed of historical weather information and heating load. The historical weather data include the measured weather data and weather forecast data. The measured weather data are taken every ten minutes, while the weather forecast data are taken by the hour. The data of two cities cover two heating seasons: one is from 20th December of 2016 to 15th March of 2017; the other is from 15th November of 2017 to 15th March of 2018.

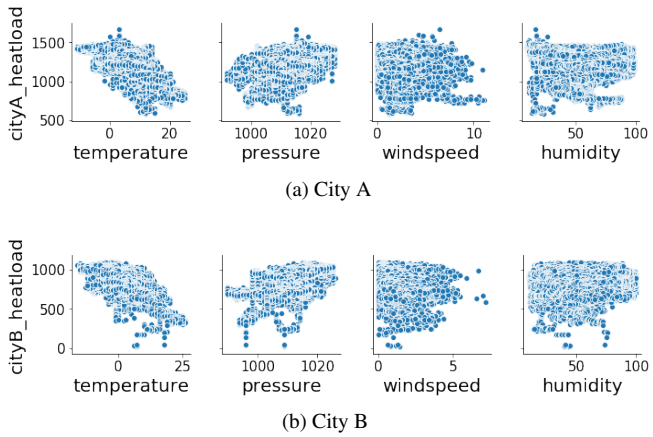
B. DATA ATTRIBUTES

The weather conditions, especially temperature, have a significant impact on the heating demand [2]. The given weather information includes temperature, pressure, humidity, and wind speed. Some statistics of the measured weather data are listed in Table 1.

Weather conditions vary between cities, as shown in Table 1. The temperature of the heating season in city A was on the average higher than that of city B, but with a lower variation. The average wind speed and pressure of heating season in city A were higher than those of city B, while the average humidity was lower in city A.

The relations of the weather variables with the heating load are shown in Fig. 1. In the heating seasons, the temperature has a negative correlation with the heating load. The pressure seems to have a positive correlation with the heating load. As for wind speed and humidity, their correlation with the heating load is hardly seen. However, they still have some influence on the heating load, since empirically the prediction accuracy would drop if we excluded these variables.

In practice, since our task is to predict the heating load of the next 24 hours, we can only use the weather forecast data



(a) City A

(b) City B

FIGURE 1. Pair-plots of the variables in city A vs city B

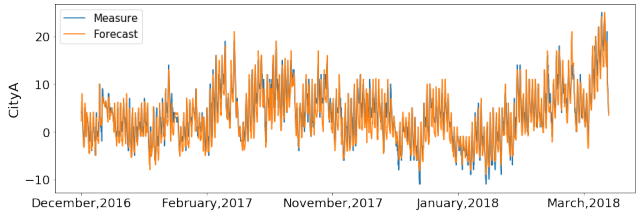


FIGURE 2. The measured and forecast temperature data of city A

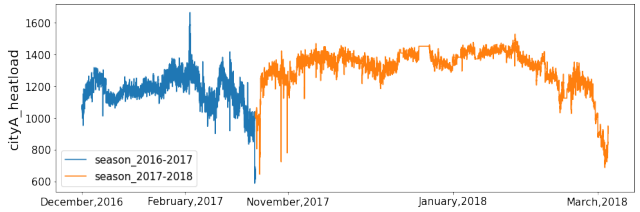


FIGURE 3. Heating load of city A

for future information. The weather forecast data are provided by the Meteorological Bureau of Shandong Province and are given on an hourly basis. In order to make a heating load forecast every ten minutes for the next 24 hours, linear interpolation for all the hourly weather forecast results is deployed in this paper. Fig. 2 presents the comparison of measured and forecast temperature data of city A. we can see that the measured and forecast data match well, which means the forecast data are reliable. Therefore, the interpolated weather forecast data are used during the training and testing of the models.

C. HEATING LOAD

The plots of the two cities' heating load are shown in Fig. 3 and Fig. 4 respectively. We can see that although the general trend of the heating load is similar, the magnitude of the heating load in city A is much greater than that of city B. The average heating load in city A is 1267.258. In contrast, it is 831.112 in city B. This indicates that the demands of

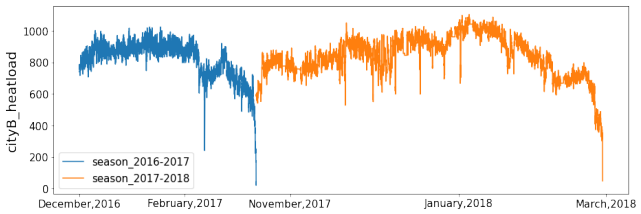


FIGURE 4. Heating load of city B

heating can vary significantly among cities. It also suggests that it may be better to train models separately for two cities, or train a model on one city and transfer to another city by fine-tuning.

Besides, the heating load increases and decreases drastically at the beginning and the end of the heating season, respectively. Such large fluctuation of the heating load poses difficulty in the training of LSTM. Therefore, data pre-processing methods, such as smoothing and scaling, are needed.

III. METHOD

This section shows how we exploit our data to build a practical model for accurate heating load forecasting. First, we introduce a base model, and then several modifications are proposed to improve its performance.

A. BASE MODEL

As for time series prediction, it is natural to adopt the Long Short Term Memory(LSTM) network, upon which we build our base model.

In [23] and [29], the structure of the LSTM unit is proposed to construct an RNN which overcomes the issue of gradient exploding and gradient vanishing. Different from a vanilla RNN unit, the LSTM unit has four computing gates working for different purposes, as illustrated in Fig. 5.

In a vanilla RNN unit, the hidden state is computed by

$$h_t = tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh}), \quad (1)$$

where $h_t \in \mathbb{R}^h$ is the hidden state at time t, $x_t \in \mathbb{R}^d$ is the input, and $W_{ih} \in \mathbb{R}^{h\times d}$, $W_{hh} \in \mathbb{R}^{h\times h}$, b_{ih}, $b_{hh} \in \mathbb{R}^h$ are trainable parameters. However, LSTM units are much more complicated and contain 4 times of parameters as the vanilla RNN units. Apart from the hidden state h_t, the LSTM units also maintain a cell state c_t to represent the memory. There are many variants of LSTM units and the one we adopted is given as follows [29]:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t * c_{t-1} + i_t * g_t \\ h_t &= o_t * tanh(c_t) \end{aligned} \quad (2)$$

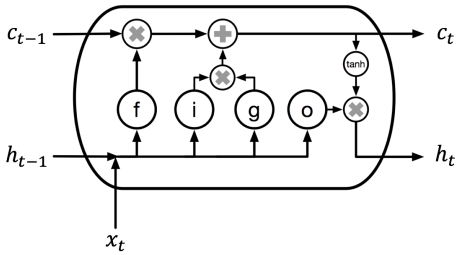


FIGURE 5. Structure of the LSTM unit.

where $h_t \in \mathbb{R}^h$ is the hidden state at time t, $x_t \in \mathbb{R}^d$ is the input vector, $c_t \in \mathbb{R}^h$ is the cell state, $i_t, f_t, g_t, o_t \in \mathbb{R}^h$ are the input, forget, cell, output gate respectively, W_{ii}, W_{if}, W_{ig}, $W_{io} \in \mathbb{R}^{h\times d}$ are trainable weights on the input vector of the respective gates, W_{hi}, W_{hf}, W_{hg}, $W_{ho} \in \mathbb{R}^{h\times h}$ are trainable weights on the hidden state of the respective gates, b_{ii}, b_{hi}, b_{if}, b_{hf}, b_{ig}, b_{hg}, $b_{io}, b_{ho} \in \mathbb{R}^h$ are trainable biases, and $*$ is the Hadamard product. Here, σ is the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3}$$

As illustrated in Fig. 6, our base model contains two stacked LSTM layers and a linear output layer. The output of the first LSTM layer serves as the input of the second LSTM layer. The network is designed to have precisely $\mathcal{T} = 1152$ time steps, which covers a range of 8 days since the data is sampled every 10 minutes. The first $\mathcal{T}_h = 1008$ time steps, i.e., 7 days, cover the history part of the data, including the historical weather forecast data and heating load. The last $\mathcal{T}_f = 144$ time steps correspond to the predicted heating load for the next day. Here, we assume that data before one week has little effect on the prediction.

At the t-th time step of the model, a 5-dimensional vector $\mathbf{x}_t = [T_t, P_t, W_t, H_t, L_{t-1}]$ is fed to the model. The vector $\mathbf{x}_t$ consists of temperature T_t, pressure P_t, wind speed W_t, humidity H_t, along with the heating load of the previous time step L_{t-1} as a history feature.

To acquire sufficient amount of sample for training the model, we slice the original data $\{(T_t, P_t, W_t, H_t, L_t)\}_{t=1}^{\mathcal{T}_d}$, where $\mathcal{T}_d$ is the amount of samples, using a sliding window of size $\mathcal{T} = 1152$ to produce N data strands $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{N}$, where $\mathbf{X}_i = \{\mathbf{x}_{i,t}\}_{t=1}^{\mathcal{T}}$ and $\mathbf{Y}_i = \{L_{i,t}\}_{t=1}^{\mathcal{T}}$. The data of two heating seasons are sliced separately so that each data strand is completely contained in one of the two heating seasons.

During training, in one forward pass, the whole network takes a sequence $\mathbf{X}_i$ as input (where the time length $\mathcal{T} = 1152$). The output of the future part $\hat{\mathbf{Y}}_i = \{\hat{L}_{i,t}\}_{t=\mathcal{T}-\mathcal{T}_f}^{\mathcal{T}}$ is used to evaluate the loss.

During inference, the input consists of two parts. Input of the history part is $\mathbf{X}_i^h = \{\mathbf{x}_{i,t}\}_{t=1}^{\mathcal{T}_h}$, which is the recorded historical data. Input of the future part is $\mathbf{X}_i^f = \{[\tilde{\mathbf{x}}_{i,t}, \hat{L}_{i,t-1}]\}_{t=\mathcal{T}-\mathcal{T}_f}^{\mathcal{T}}$, where $\tilde{\mathbf{x}}_{i,t} = [\tilde{T}_{i,t}, \tilde{P}_{i,t}, \tilde{W}_{i,t}, \tilde{H}_{i,t}]$ includes only weather features obtained from weather forecasting, and $\hat{L}_{i,t-1}$ is the output of the network at the previous time step since the heating load in the future is unknown at the current time.

Although LSTM is well-performed in dealing with non-stationary time series data, we have the following reasons to apply modifications to the base model to boost performance:

1) Improving prediction at the end of each heating season

The heating season ends in mid-March each year. During the ending month of each heating season, the heating load drops dramatically as the heating machines are gradually shut down. However, this pattern occurs only once a year and covers 1-2 weeks, which makes it hard to learn. Therefore, we reconsider the pre-processing of data so that the model learns the importance of the drop, which is illustrated in section B and C.

2) Better generalization

The model is finally to be implemented on an online system to produce heating load forecasting automatically and continuously with minimal human interactions. Therefore, it should work as long time and with as few adjustments as possible, which requires the model to generalize well on new data and be robust to input noise. For this, an ensemble strategy is adopted, which is detailed in section D.

B. SMOOTHING

The recorded heating load is rather noisy. Proper smoothing of the data can make it easier for the model to capture local features and the major trend.

The heating load is smoothed by a sliding window of size b. Every data point is replaced by an average of over b consecutive data points in the previous time instances (including itself). This is to ensure that no future information is required to calculate the average. To be more precise, at each t, the smoothing operation can be formulated as

$$y_t^{smooth} = \frac{1}{b}\sum_{i=0}^{b-1} y_{t-i} \tag{4}$$

where y_t is the unprocessed value of heating load at time t, and b represents the size of the sliding window. We choose $b = 12$ in practice.

C. LOCAL RESCALING

Attributes of the raw data are of different scales and distributions and need to be scaled appropriately before being fed to the model. Otherwise, we often observe the slow convergence of the training and poor performance of the trained model. It is natural to pre-process the raw data to remove certain physical meanings while maintaining the statistical characteristics of the data. For that, we apply the following Min-Max rescaling transformation on training data to transform all the features into a fixed range $[m, M]$. For a given time series $\{x_i\}$, this can be formulated as

$$R(x_i; s_x) = s_x(x_i - x_{min}) + m \tag{5}$$

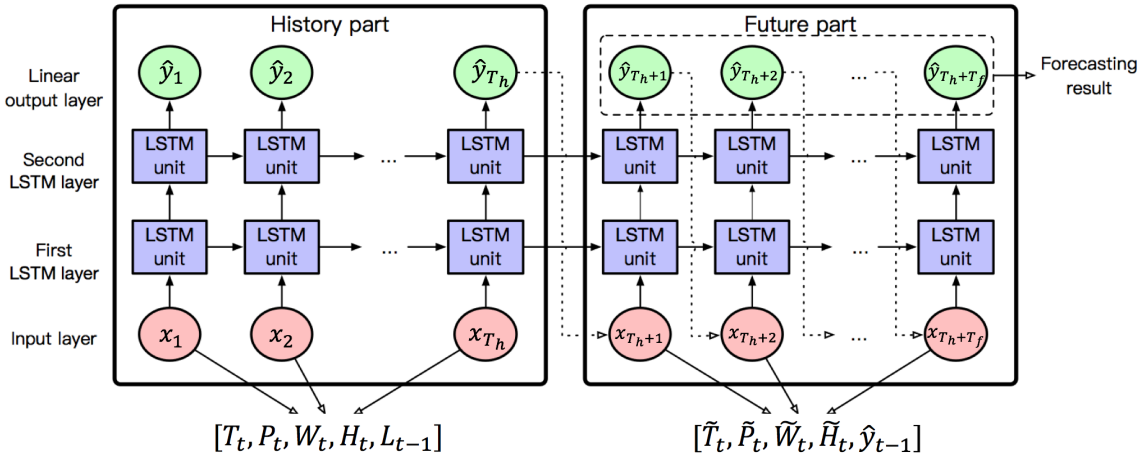


FIGURE 6. Structure of the base model.

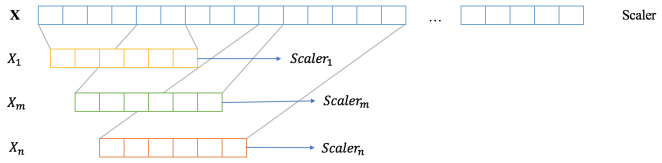


FIGURE 7. Fitting a scalar separately on each data strand generated by sliding windows instead of fitting one scalar on all the data

where

$$s_x = (M - m)/(x_{max} - x_{min}), \tag{6}$$

$x_{min} = \min_i x_i$ and $x_{max} = \max_i x_i$. In practice, we take $m = -0.7$ and $M = 0.7$.

The data are transformed by applying (5) to each data attribute of each strand independently, i.e.

$$\mathbf{x}'_i = \mathbf{R}(\mathbf{x}_i; \mathbf{s}_\mathbf{x}), i = 1, ..., N \tag{7}$$

where $\mathbf{R}$ represents a vector function which transforms $\mathbf{x}_i$ component-wisely and $\mathbf{s}_\mathbf{x}$ is a component-wise scaling factor of the entire time series.

Since the proposed model only takes in data strands of a week in length, we have another option to scale each data strand locally rather than globally. For the i-th data strand, this can be formulated as

$$\mathbf{x}'_{i,t} = \mathbf{R}_i(\mathbf{x}_{i,t}; \mathbf{s}_{\mathbf{x}_i}), t = 1, ..., T \tag{8}$$

where $\mathbf{R}_i$ represents the function used only to transform the data strand X_i. The training target $L_{i,t}$ shares the same transformation with the component $L_{i,t-1}$ of $\mathbf{x}_{i,t}$. Note that $\mathbf{s}_{\mathbf{x}_i}$ is specific to X_i and is calculated locally.

In this situation, each data strand is transformed by a different scale factor $\mathbf{s}_{\mathbf{x}_i}$, but has the same range after rescaling. Moreover, the rescaling within a given data strand reflects only the variation in the underlying period and is unaffected by the absolute value of the entire time series.

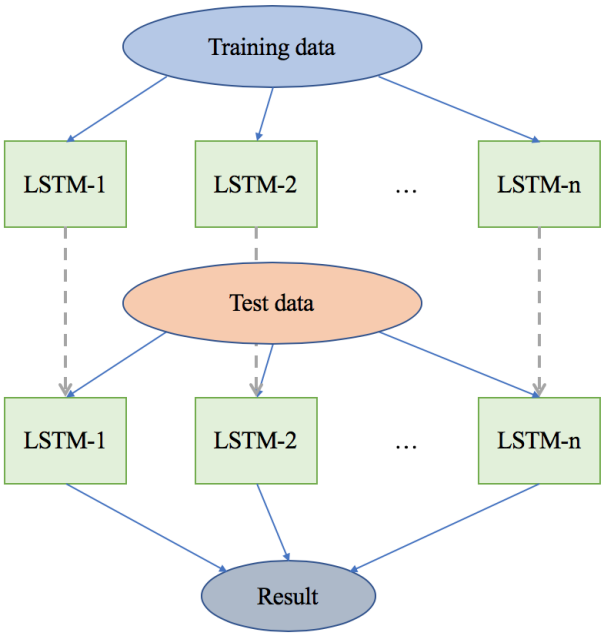


FIGURE 8. Model ensembling structure

During inference, the scales $s_{\mathbf{x}_i}$ are only fit on the history part of the data strands and are then applied to transform both the history part and future part, so that no future information is used in the calculation.

D. ENSEMBLE

Inspired by EnResnet proposed by [30], we ensemble the LSTM units in a similar way. We train n LSTMs with different initialization in parallel. Then we average the results of each network when making predictions. This simple ensemble method improves generalization and robustness.

Also, during the training of each model, a gaussian noise is added to the hidden states between every time step in both of

the LSTM layers. This is to ensure the model to be resistant to perturbation of input data. During inference, we evaluate the model several times with noise injections and then average all the outputs.

Specifically, the native LSTM cells give the hidden states by

$$h_t = U(h_{t-1}, x_t), \tag{9}$$

where such U is given by (2). The noise injection is given by

$$\tilde{h}_t = U(h_{t-1}, x_t) + \gamma\mathcal{N}(0, 1), \tag{10}$$

where $\gamma = \beta\sqrt{Var(U(h_{t-1}, x_t))}$ and β is a tunable parameter which is set to 0.1 in our experiments. The noise is sampled independently at each time of evaluation.

E. LOSS FUNCTION

It is natural to choose the following loss function to measure the absolute error of the prediction.

$$\mathcal{L} = \frac{1}{N\mathcal{T}}\sum_{i=1}^{N}\sum_{t=\mathcal{T}-\mathcal{T}_h}^{\mathcal{T}} |\hat{y}'_{it} - y'_{it}| \tag{11}$$

where $\hat{y}'_{it}$ is the model's output at the t-th time step for the i-th data strand, y'_{it} is the corresponding ground truth, N is the total number of samples, and t from $\mathcal{T} - \mathcal{T}_h$ to $\mathcal{T}$ represents the future part of our model which produces the predicted sequence. The length is 144 in our settings since we need the model to predict the heating load with 10 minutes' increment for the next 24 hours.

However, under local rescaling (section C), the absolute error on each data strand is not within equal value due to the different rescaling function applied, which eventually hurts the prediction accuracy (see Table 3). To solve this issue, a weighted ℓ_1 loss is introduced to help correct the difference in the weights of the samples. The weighted ℓ_1 loss is given by

$$\mathcal{L}_{weighted} = \frac{1}{N\mathcal{T}}\sum_{i=1}^{N}\sum_{t=\mathcal{T}-\mathcal{T}_h}^{\mathcal{T}} w_i|\hat{y}'_{it} - y'_{it}|, \tag{12}$$

where the weights

$$w_i = \alpha\frac{\max_t y_{it} - \min_t y_{it}}{\bar{y}_i} \tag{13}$$

and $\bar{y}_i = \frac{1}{\mathcal{T}}\sum_{t=1}^{\mathcal{T}} y_{it}$, α is a constant to regularize the weights. As was described in section C, the scaling is a linear mapping. For the i-th data strand, the heating load at time step t is transformed by

$$y'_{it} = s_i y_{it} + d_i \tag{14}$$

where $s_i = (M - m)/(\max_t y_{it} - \min_t y_{it})$ and $d_i = m - s_i \min_t y_{it}$. Then the absolute error before scaling is

$$|\hat{y}_{it} - y_{it}| = \frac{1}{s_i}|\hat{y}'_{it} - y'_{it}|, \tag{15}$$

which indicates that we should multiply the absolute error calculated on the scaled data by $\frac{1}{s_i}$. Moreover, we notice that the absolute errors of different samples are still not of equal importance even on the original scale. Considering error rate is a more frequently-used measurement to assess the result, we further correct the loss on each sample to

$$\ell(\hat{y}_{it}, y_{it}) = \frac{1}{s_i}\frac{1}{\bar{y}_i}|\hat{y}'_{it} - y'_{it}| \approx \frac{|\hat{y}_{it} - y_{it}|}{y_{it}}. \tag{16}$$

This leads to an approximate value of error rate, and also the weighted ℓ_1 loss described in (12) and (13). Therefore, the scale and level of the original sequence are taken into consideration to leveraging the importance of each sample, which makes (12) a more consistent loss.

IV. RESULTS AND DISCUSSION

In this section, we apply our model to the real data described in Section II to evaluate its performance. First, we explore how the smoothing mentioned above, scaling, the weighted loss function influence the performance of the model using off-line data and hence conclude the best model settings. Then, we introduce how the best model is applied to the online system and report its performance.

A. IMPLEMENTATION DETAILS

We first integrate the weather forecast data into historical data. As mentioned above, the weather forecast data are interpolated so as to be connected with the history data with respect to the timestamp. Then we compute the smoothing on the heating load as depicted in section III.B. If local rescaling is not adopted, a global scaling (7) is applied at this stage. The data is then sliced to generate data strands. If local rescaling is adopted, we perform the local transformation (8) for each data strand.

We evaluate the model using the mean absolute percentage error (MAPE), which calculates a percentage error on each location and then average over all locations. The MAPE measure is formulated as

$$MAPE = \frac{1}{N\mathcal{T}}\sum_{i=1}^{N}\sum_{t=\mathcal{T}-\mathcal{T}_h}^{\mathcal{T}} \frac{|\hat{y}_{it} - y_{it}|}{y_{it}}. \tag{17}$$

We scale the output of our model back to its original scale before evaluating the error using the MAPE.

During training of the model, we use stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.05 which decays to 1/10 every 30 epochs. The LSTM models are trained within 100 epochs and the training takes 117 seconds per epoch in average on a Nvidia Geforce 1080Ti GPU. The models are implemented in PyTorch.

B. OFF-LINE TESTS

The historical data covers the period of 2 heating seasons. The first 70% data are used for training the model, roughly including the first heating season and half of the second heating season. The rest 30% data are used for testing. The data from City A and City B are split at 2018/1/21 13:10 and 2018/1/21 22:40, respectively. We train and test our model separately on the two datasets.

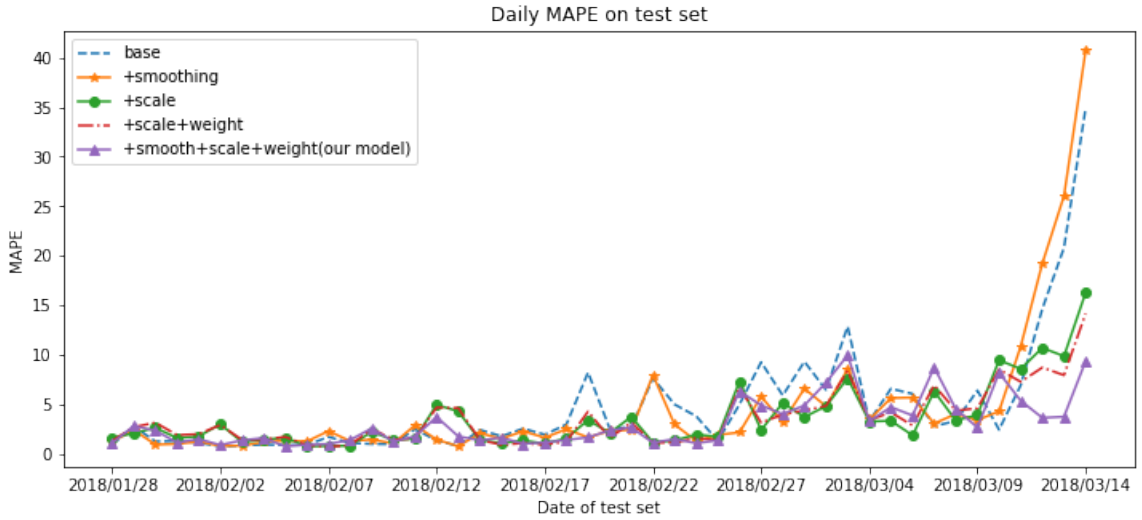


FIGURE 9. Daily MAPE on Test Set of City A

TABLE 2. comparison of traditional statistical models and the base LSTM model in terms of MAPE(%).

Index	Model name	City A		City B	
		train	test	train	test
0	linear	5.51	9.54	5.90	10.91
1	random forest	0.54	10.28	0.73	9.94
2	gradient boosting	4.12	8.89	4.77	10.06
3	base LSTM	3.69	4.80	5.45	7.92

TABLE 3. comparison of different model settings in terms of MAPE(%).

Index	Model setting	City A		City B	
		train	test	train	test
0	base	3.69	4.80	5.45	7.92
1	+smoothing	3.59	4.60	5.44	8.21
2	+scale	3.21	3.56	5.63	6.20
3	+scale+weight	3.25	3.48	5.79	**6.01**
4	**our model**	2.98	**3.08**	6.04	6.38
5	our model (ensemble)	2.96	3.12	5.77	6.31

First, we compare the performance of the LSTM base model with several traditional statistical models to demonstrate the advantage of deep models. Considering temporal factors, we extract data and time features within one-hot transformation to be included in the linear, random forest, and gradient boosting models. The results are shown in Table 2. The traditional models generally produce much higher MAPE on test data than on train data, indicating that they suffer from over-fitting. However, our base LSTM model shows a competitive result on test set among all the models, which reduces the MAPE from around 10 to 4.80 and 7.92 in the test set of City A and City B, respectively.

To improve the base LSTM model, we have proposed two data pre-processing methods mentioned in section III.B and III.C. Now we demonstrate the effectiveness of the two operations. The results are shown in Table 3. Based on the base model, adding smoothing causes a slight drop in both training and testing errors for City A, whereas it does not benefit the model on City B. As for the local rescaling, the MAPE drops significantly for both cities, indicating that it is rather crucial to the model.

In particular, the forecast result and daily MAPE under different settings in the test set of City A are shown in Fig. 9. In the last several days of the test set, which falls in March 2018 correspondingly, the heating load encounters a sharp decline, and the MAPE is high at this stage. Our proposed methods bring some contribution in reducing the MAPE of these days and thus have a better overall performance.

The local rescaling brings a notable decrease in MAPE. However, as mentioned above, the unweighted ℓ_1 loss does not match this option. Experiments show that using the weighted ℓ_1 loss leads to a better result, where the MAPE drops around 0.1 compared with the test using only local rescaling.

We finally propose our model by combining the above methods, which adopts the smoothing, local rescaling, and the weighted ℓ_1 loss. In City A, our model produces the best results among all the experiments discussed above. In City B, the smoothing operation can be optional since it does not improve the model.

Furthermore, we explore the performance of ensembling the models. We can see that there is a slight decrease in MAPE of the ensembled models. Nonetheless, they may perform better than the single model when there are missing entries in the input data. In practical applications, data with noise or missing entries are inevitably due to the complex working environment of the power stations. To simulate such a situation, we randomly set each entry of the input data to 0 with a probability $0 < p < 1$. In City B, as p rises, the MAPE of the ensembled models is generally lower than that of the

TABLE 4. The performance on test set of the ensembled models under different missing rate p in terms of MAPE(%).

p		0	0.01	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7
City A	single	3.0770	3.0713	3.1035	3.1484	3.3352	3.4325	3.8298	3.8826	4.1991	4.7383
	ensemble	3.1223	3.1258	3.1596	3.1822	3.3503	3.4241	3.7871	3.8613	4.1743	4.7150
City B	single	6.3827	6.3713	6.5158	7.5017	7.6739	9.2842	9.4417	9.1473	9.4523	9.7209
	ensemble	6.3092	6.3425	6.4718	7.2690	7.4386	8.9747	9.0367	8.8829	9.3564	9.5860

single model by around 0.2. However, the ensembled models do not show a clear improvement in City A.

With the above results, the best model for City A is our proposed model (Index 4 in Table.3) because of its lowest MAPE in test data. For City B, we suggest the model without smoothing and better to be the ensembled version, because the heating load of City B is noisier and has a more considerable variation than City A.

C. ONLINE TESTS

We implemented our model to the online system, which is constructed as Fig. 10.

The data collection module collects and integrates the recorded historical data into a database. Every day at 8 a.m., a scheduled program starts and fetches the nearest 1008 records along with a 24-hours weather forecast data, and calls the prediction module. The prediction module first preprocesses the loaded data and evaluates through the model pre-trained on the off-line data as described by the previous subsection. Then, the results are sent to the "intelligent data center" and made available to the decision-makers.

An automated model updating mechanism was designed in order to re-train the model using the most recent data. To make it simple for users to operate, we build a neat and user-friendly interface to re-train the model with preconfigured hyper-parameters. Therefore, users can update the model through very simple actions while monitoring its performance. In practice, The training interface can also be called by the training program to train new models on the latest data. The newly trained models are saved to the model warehouse and can be loaded by the prediction interface.

Our model trained on historical data from December 2016 to March 2018 was submitted to the online system to forecast the heating load in City A in the heating season of the year 2018. Due to the limited computing resource of the online system, the ensembled version was not deployed. From November 2018 to March 2019, the model was tuned using the automated training program three times on December 2018, January 2019, and February 2019 separately, and the updated model was employed on a monthly basis. At the end of the heating season, the model reported a 4.09% MAPE for the entire season, which is comparable to the results of the off-line tests on City A. See Fig. 11 for a visual comparison between the predicted and measured heating load during the heating season of 2018 in City A.

V. CONCLUSION AND FUTURE WORK

We proposed a novel heating load prediction model based on LSTM. With carefully designed network structure, data preprocessing operations, and the loss function, our proposed model has achieved satisfactory accuracy on historical data provided by Electric Power Academy of Shandong Province in China. The model was further implemented to an online system in a city of Shandong province to make continuous and fully automatic forecasting and to produce highly reliable results during the heating season in 2018.

For further work, we would like to adapt the model to other cities in the northern provinces of China using transfer learning techniques, as data from more power plants have been collected. In addition, deep network architecture can be further improved as complex data are introduced.

REFERENCES

[1] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," IEEE Transactions on Smart Grid, 2018.
[2] E. Dotzauer, "Simple model for prediction of loads in district-heating systems," Applied Energy, vol. 73, no. 3-4, pp. 277–284, 2002.
[3] K. M. Powell, A. Sriprasad, W. J. Cole, and T. F. Edgar, "Heating, cooling, and electrical load forecasting for a large-scale district energy system," Energy, vol. 74, pp. 877–885, 2014.
[4] K.-B. Song, Y.-S. Baek, D. H. Hong, and G. Jang, "Short-term load forecasting for the holidays using fuzzy linear regression method," IEEE transactions on power systems, vol. 20, no. 1, pp. 96–101, 2005.
[5] B.-J. Chen, M.-W. Chang et al., "Load forecasting using support vector machines: A study on eunite competition 2001," IEEE transactions on power systems, vol. 19, no. 4, pp. 1821–1830, 2004.
[6] M. Mohandes, "Support vector machines for short-term electrical load forecasting," International Journal of Energy Research, vol. 26, no. 4, pp. 335–345, 2002.
[7] J. Zeng and W. Qiao, "Support vector machine-based short-term wind power forecasting," in 2011 IEEE/PES Power Systems Conference and Exposition. IEEE, 2011, pp. 1–8.
[8] A. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, and B. Venkatesh, "Improved short-term load forecasting using bagged neural networks," Electric Power Systems Research, vol. 125, pp. 109–115, 2015.
[9] P. Singh and P. Dwivedi, "Integration of new evolution-

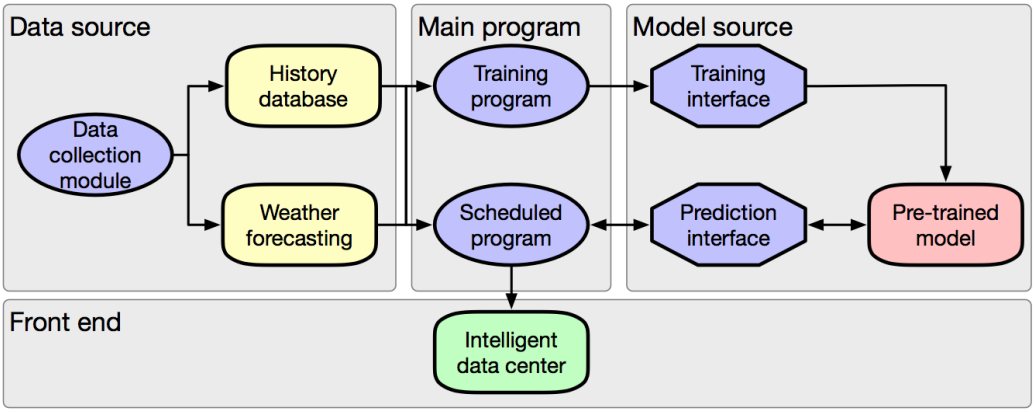


FIGURE 10. Online system structure

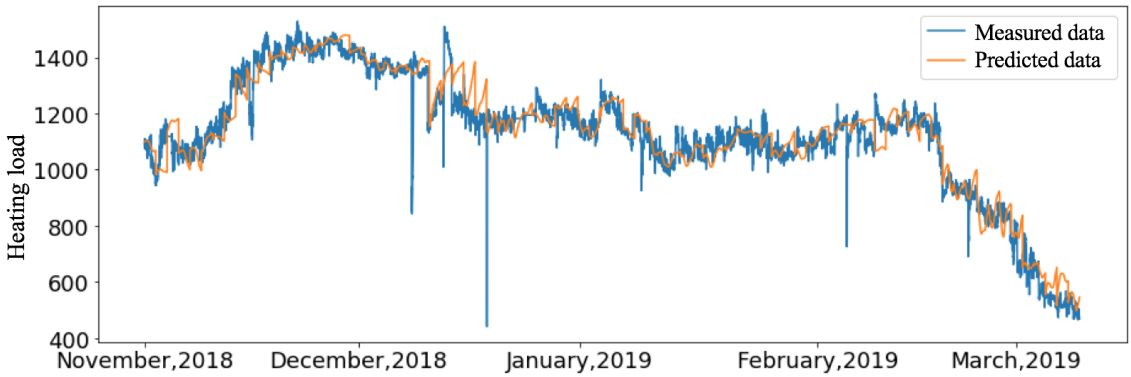


FIGURE 11. Online test result in City A. The blue line represents the measured heating load during the heating season. The yellow line represents the one-day-ahead forecast of the heating load given by our online model. In December 2018, the heating load fluctuated by a large margin due to the manual operations within the power plant or some mistake in the data collection procedure. Since the online model takes historical data of a week's length as input, the abnormal data affected the performance of the forecast.

ary approach with artificial neural network for solving short term load forecast problem," Applied energy, vol. 217, pp. 537–549, 2018.

[10] E. Ceperic, V. Ceperic, and A. Baric, "A strategy for short-term load forecasting by support vector regression machines," IEEE Transactions on Power Systems, vol. 28, no. 4, pp. 4356–4364, 2013.

[11] A. Kavousi-Fard, H. Samet, and F. Marzbani, "A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting," Expert systems with applications, vol. 41, no. 13, pp. 6047–6056, 2014.

[12] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," IEEE Transactions on power systems, vol. 16, no. 1, pp. 44–55, 2001.

[13] T. Ahmad and H. Chen, "Short and medium-term forecasting of cooling and heating load demand in building environment with data-mining based approaches," Energy and Buildings, vol. 166, pp. 460–476, 2018.

[14] S. S. Roy, R. Roy, and V. E. Balas, "Estimating heating load in buildings using multivariate adaptive regression splines, extreme learning machine, a hybrid model of mars and elm," Renewable and Sustainable Energy Reviews, vol. 82, pp. 4256–4268, 2018.

[15] G. Suryanarayana, J. Lago, D. Geysen, P. Aleksiejuk, and C. Johansson, "Thermal load forecasting in district heating networks using deep learning and advanced feature selection methods," Energy, vol. 157, pp. 141–149, 2018.

[16] S.-J. Huang and K.-R. Shih, "Short-term load forecasting via arma model identification including non-gaussian process considerations," IEEE Transactions on power systems, vol. 18, no. 2, pp. 673–679, 2003.

[17] S. Gao, Y. He, and H. Chen, "Wind speed forecast for wind farms based on arma-arch model," in 2009 International Conference on Sustainable Power Generation and Supply. IEEE, 2009, pp. 1–4.

[18] K. Liu, S. Subbarayan, R. R. Shoults, M. T. Manry, C. Kwan, F. I. Lewis, and J. Naccarino, "Comparison of very short-term load forecasting techniques," IEEE Transactions on Power Systems, vol. 11, no. 2, pp. 877–

882, May 1996.
[19] M. Valipour, M. E. Banihabib, and S. M. R. Behbahani, "Comparison of the arma, arima, and the autoregressive artificial neural network models in forecasting the monthly inflow of dez dam reservoir," Journal of hydrology, vol. 476, pp. 433–441, 2013.
[20] N. Amjady, "Short-term hourly load forecasting using time-series modeling with peak load estimation capability," IEEE Transactions on Power Systems, vol. 16, no. 3, pp. 498–505, 2001.
[21] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "Arima models to predict next-day electricity prices," IEEE transactions on power systems, vol. 18, no. 3, pp. 1014–1020, 2003.
[22] D. Alberg and M. Last, "Short-term load forecasting in smart meters with sliding window-based arima algorithms," Vietnam Journal of Computer Science, vol. 5, no. 3-4, pp. 241–249, 2018.
[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735
[24] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," arXiv preprint arXiv:1506.01057, 2015.
[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in neural information processing systems, 2014, pp. 3104–3112.
[26] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv preprint arXiv:1609.08144, 2016.
[27] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in Fifteenth annual conference of the international speech communication association, 2014.
[28] D. Koschwitz, J. Frisch, and C. Van Treeck, "Data-driven heating and cooling load predictions for non-residential buildings based on support vector machine regression and narx recurrent neural network: A comparative study on district scale," Energy, vol. 165, pp. 134–142, 2018.
[29] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," Neural Computation, vol. 12, no. 10, pp. 2451–2471, 2000.
[30] B. Wang, B. Yuan, Z. Shi, and S. J. Osher, "Enresnet: Resnet ensemble via the feynman-kac formalism," arXiv preprint arXiv:1811.10745, 2018.