Energy & Buildings 252 (2021) 111379

Contents lists available at ScienceDirect

Energy & Buildings

journal homepage: www.elsevier.com/locate/enb

Interpretable deep learning model for building energy consumption prediction based on attention mechanism

Yuan Gao [a], Yingjun Ruan [b,*]

[a] School of Engineering, The University of Tokyo, Japan
[b] School of Mechanical and Energy Engineering Tongji University, China

ARTICLE INFO

Article history:
Received 29 April 2021
Revised 15 July 2021
Accepted 19 August 2021
Available online 21 August 2021

Keywords:
Building energy forecasting
Encoder and decoder
Attention
Interpretable deep learning model

ABSTRACT

An effective and accurate building energy consumption prediction model is an important means to effectively use building management systems and improve energy efficiency. To cope with the development and changes in digital data, data-driven models, especially deep learning models, have been applied for the prediction of energy consumption and have achieved good accuracy. However, as a deep learning model that can process high-dimensional data, the model often lacks interpretability, which limits the further application and promotion of the model. This paper proposes three interpretable encoder and decoder models based on long short-term memory (LSTM) and self-attention. Attention based on hidden layer states and feature-based attention improves the interpretability of the deep learning models. A case study of one office building is discussed to demonstrate the proposed method and models. Firstly, the addition in future real weather information yields only a 0.54% improvement in the MAPE. The visualization of the model attention weights improves the interpretability of the model at the hidden state level and feature level. For the hidden state of different time steps, the LSTM network will focus on the hidden state of the last time step because it contains more information. The Transformer model gives almost equal attention weight to each day in the coding sequence. For the interpretable results at the feature level, daily max temperature, mean temperature, min temperature, and dew point temperature are the four most important features. The four characteristics of pressure, wind speed-related features, and holidays have the lowest average weights.

1. Introduction and literature review

With the growth of the global population, energy and environmental issues have undoubtedly become important issues for the development of human society in the 21st century [1]. In the development of urbanization, more than 80% of human activity time is spent in buildings [2]. This change in behavior has led to an increasing proportion of building energy consumption in the total energy consumption. According to data from different sources, building energy consumption accounts for nearly 40% of the total global energy consumption and has become the world's largest energy consumption category [3,4]. Such severe building energy consumption has aroused extensive research and attention for energy management systems by researchers [5,6]. Through the energy management system, building users can effectively improve energy efficiency, carry out timely fault diagnosis and demand response strategies, and accurate load forecasting is an indispensable part and prerequisite. However, since the components of energy consumption data are often quite complex and involve many parameters such as human behavior, the prediction of energy consumption is usually quite complicated and difficult.

The current popular building energy consumption prediction methods can be roughly divided into physical model methods and data-driven model methods [7]. Among the two methods, the data-driven method is gaining increasing popularity due to the practicality of its algorithm, faster update frequency and better prediction accuracy. In addition, with the installation of various intelligent measuring meters in buildings, data-driven methods can better apply brand-new data obtained from buildings to construct models [8].

Traditional machine learning algorithms have received extensive attention in the field of energy consumption prediction due to their good mathematical proofs and excellent interpretability. Fumo et al. [9] used meteorological parameters such as outdoor dry bulb temperature to predict the energy consumption of a single building using a multivariate linear return method. Ruiz et al. [10] focused on a university building by using a nonlinear autoregres-

* Corresponding author.
E-mail address: ruanyj@tongji.edu.cn (Y. Ruan).

https://doi.org/10.1016/j.enbuild.2021.111379
0378-7788/

sive model, and the inputs also included previous energy consumption and temperature. Paudel et al. [11] set the research target as residential buildings in France and used support vector machines to predict energy consumption. The input data included energy consumption, weather data and personnel usage for users. Sun et al.[12] developed a simplified online cooling load forecasting model based on reference days and forecast errors, and verified the effect of the model using measured data from super high-rise buildings in Hong Kong. Other popular traditional machine learning algorithms such as decision trees [13,14] and random forest [2], are also used in the field of energy consumption prediction in similar situations. The advantages of the traditional machine learning algorithms lie in their good mathematical explanations and relatively mature external application libraries. However, traditional machine learning algorithms need to construct data through complex feature engineering[15,16], models often cannot handle high-dimensional feature data, and this model capability is becoming increasingly important.

Deep learning is another branch of data-driven models. These algorithms have very good application cases in speech recognition [17], image classification [18], and natural language processing [19]. Because of its powerful feature extraction capabilities and high-dimensional data processing capabilities brought about by its model depth, an increasing number of researchers are favored in the field of building energy forecasting. Whether it is a hybrid model that combines a neural network and a particle swarm intelligence (SI) algorithm [20] or simply uses a simple artificial neural network (ANN) for modeling [21], the application of deep learning methods is also diverse. Yuan et al. [22] used wavelet neural networks to predict building energy consumption in hotels and shopping malls. Chitalia et al. [23] used a deep learning model to construct a mature reuse framework for short-term energy consumption prediction in commercial buildings. Fan et al. [24] used more advanced transfer learning methods for short-term building energy consumption prediction, and the results showed that transfer learning methods can effectively alleviate the problem of poor building energy consumption prediction accuracy under poor information. The pure dense neural network avoids the more complex feature processing and the difficulty of traditional machine learning algorithms for high-dimensional data. However, the calculation between the input data of the model is usually still independent, and there is a lack of means of processing for time series data such as energy consumption. As a neural network that combines time dependence, recurrent neural networks (RNNs) are widely used in the prediction of nonlinear time series, and their effectiveness has naturally been proven in the field of building energy consumption[25,26]. With the deepening of research, the application of RNNs in the field of building energy consumption prediction has also expanded to more advanced LSTM [27] and GRU [28] networks.

From the above literature review, the research on the data-driven building energy consumption prediction model, especially the deep learning model, can be relatively mature, but there is still one problem that has not been resolved. Compared with the traditional machine-learning model, the deep-learning method is essentially a black box model, and its lack of interpretability is a big limitation for the further expansion of its application. In fact, the interpretability of deep learning model could have a positive impact on error analysis, feature selection, and installation of smart sensors[29,30].

In light of these studies and the current problems with physics models and data-driven models mentioned above, the model proposed in this paper improved all the corns used the methods in the following: Single step encoder and decoder structure with RNN; Attention mechanism; Transformer model.

The idea is to first use the encoder and decoder structure to distinguish the processing mode for historically known information and future known or unknown information. Encoders and decoders are widely used in the field of natural language processing and were first used for sequence-to-sequence encoding and processing [31,32]. However, in this paper, by adjusting the encoder and decoder structure to single-step energy consumption prediction and flexibly set the input data dimensions in the decoder, the input of future information and the input of historical information are effectively distinguished by the encoder and the decoder. Further, by changing the calculation method of "context" vector in encoder [33], the entire model framework has strong flexibility, and different interpretable components can be easily added to improve the performance of the model.

In addition, to improve the interpretability of the deep learning model, this paper uses the attention mechanism combined with the recurrent neural network to construct an interpretable energy consumption prediction deep-learning model. The attention mechanism in deep learning is similar to the human selective visual attention mechanism, which has been widely used in various tasks, such as machine translation, image captioning, and video motion recognition [34,35]. However, its application in the field of building energy consumption prediction is still relatively limited, and this application is used mainly in conjunction with recurrent neural networks at the hidden state level. That is, the interpretable operation of attention is performed on the hidden layer state of the recurrent neural network. Recurrent neural networks using the attention mechanism will consider all previous hidden states, while ordinary recurrent neural networks will use only the last hidden state output, and the use of incorrectly generated hidden state vectors cannot obtain satisfactory prediction accuracy [36]. However, in this paper, we not only build the attention mechanism on the hidden state level but also further calculate the attention mechanism for the input features in the network so that the attention results of the model can show in more detail the importance of different features at different time steps.

Finally, considering the unequal amount of state information in the hidden layer in the recurrent neural network, a transformer model [37] based on a pure attention mechanism for encoding and decoding is proposed. The transformer can output the hidden layer state of the same amount of information at each time step while avoiding the iterative calculation in RNN. By visualizing the attention weight in the transformer model, we can understand the importance of each day and each feature more clearly through the results than in the LSTM model.

Above all, considering the demerit of the existed deep learning models, the contribution of this paper can be concluded as follows:

- Single step encoder and decoder model structure is introduced for the basic of interpretable deep learning prediction model
- Attention mechanism is applied in both hidden state level and feature level to improve interpretability;
- A brand-new transformer model is proposed to deal with the problem of information accumulation in RNNs.

The paper is organized as follows: Section 2 introduces related concepts and methods, the LSTM, the attention layer, and the transformer. Section 3 presents a case study in detail of office buildings using the model proposed. Furthermore, section 4 introduces the results and discussion section. Finally, section 5 delivers our conclusions.

2. Methods

This section will give a detailed description of all the deep learning models and methods used in this paper. Fig. 1 provides the common workflow for each prediction model. In the whole

method, we hope that RNN and Transformer can process and encode time series information well, while the decoder combines the state vector of the encoder with additional information to produce good prediction results. On this basis, we hope that additional attention components can effectively improve the interpretability of the model by visualizing the calculation process. The following section will describe the data processing methods and the building methods in detail for each model.

2.1. Data preprocessing

For machine learning algorithms, data preprocessing, which plays a pivotal role in the entire machine learning process, is also called feature engineering. Although deep learning greatly weakens the role of feature engineering, the difference in the value of input features will greatly affect the weight of the feature during model training, so it is still necessary to normalize features in deep learning [33].

In this paper, three normalization methods are used for different features and labels.: z-score, log normalization, and one-hot code.

The z-score is a very widely used normalization method. The specific formula is as follows:

$$x = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean of all of the characteristics, σ indicates the standard deviation of the mean, and x^* is the result of standardizing x.

However, for time-series forecasting, the statistical characteristics (mean or variance) of the same time series in different time periods are likely to be different, causing the training set and test set data in the same time series not to obey the same distribution. For relatively stable weather parameters, it is more reasonable to use the z-score. For the time series of building energy consumption data that may be significantly improved over time and social development, this study uses log standardization for preprocessing [38].

The log normalization is represented by the following equation:

$$z = \log_a x \tag{2}$$

where x is the raw data, and z is the result of log standardizing$x$.

Using log standardization can solve the problem of the possible heterogeneous distribution of training data and test data, and the statistical characteristics of training data will not be involved when recovering the prediction results. In addition, due to the characteristics of the function itself, the log function can smooth the original time series appropriately, which is very advantageous for the prediction of the time series.

2.2. Encoder and decoder learning

2.2.1. Lstm

Deep networks use multilayer feedforward neural networks for stacking. The nonlinear activation function between each layer of the neural network gives the deep network a powerful representation [39]. However, this kind of network cannot effectively process and learn sequence data because the calculation process of the entire network is parallel, and there is no time dependence.

Considering the disadvantages of dense deep networks, this study adopts long- and short-term memory (LSTM) networks as the basic unit of constructing models [40]. As an improved recurrent neural network, LSTM has a unique network structural advantage in processing time series.

$$z^f = \text{sigmoid}(\boldsymbol{bf} + U_f xt + W_f h_{t-1}) \tag{3}$$

$$z^i = \text{sigmoid}(\boldsymbol{b}_i + U_i xt + W_i h_{t-1}) \tag{4}$$

$$z^o = \text{sigmoid}(\boldsymbol{b}_o + U_o xt + W_o h_{t-1}) \tag{5}$$

$$z = \tanh(\boldsymbol{b} + Uxt + Wh_{t-1}) \tag{6}$$

The above four formulas represent four different results obtained in different ways by LSTM processing input information. U, W and b represent the parameters that the model needs to learn. The superscripts and subscripts of 'f', 'i', and 'o', respectively, represent the calculation process of the forget gate, input gate and output gate. sigmoid$(\cdot)$ represents the sigmoid activation function. This activation function transforms the result into a probability value of 0–1, becomes a gated state, and finally obtains the forget gate control state z^f, the input gate control state z^i, and the output gate control state z^o. z converts the result into a value between -1 and 1 through a tanh activation function (tanh is used here because it is used as input data, not a gate signal).

The following formulas further explain how to use these four states to obtain the output of the current time step in the LSTM.

$$c^t = z^f c^{t-1} + z^i z \tag{7}$$

$$h^t = z^o \tanh(c^t) \tag{8}$$

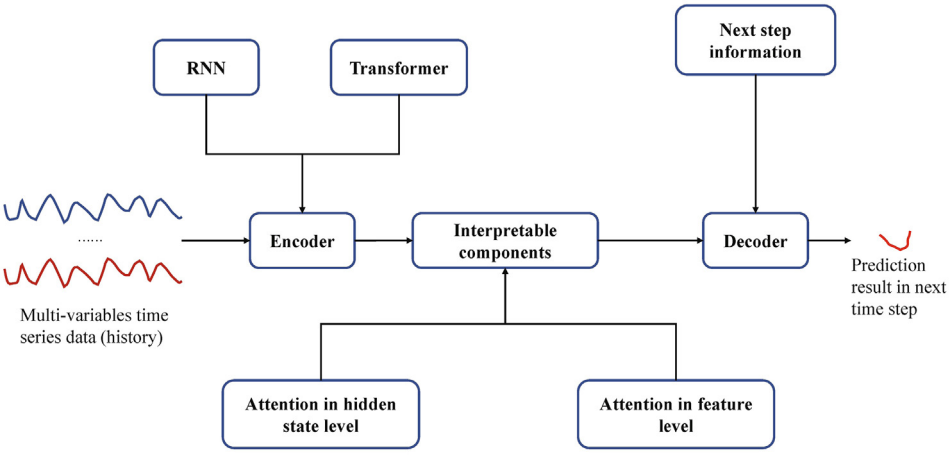

Fig. 1. Whole workflow.

There are three main stages inside LSTM. First, forgotten stage: this stage is mainly to selectively forget the input vector passed in the previous moment. To put it simply, "forget the unimportant and remember the important". Specifically, the calculated z^f (f means forget) is used as the forget gate control to control the c^{t-1} of the previous state that needs to be left, and that needs to be forgotten. Second, selecting the memory stage: this stage selectively "memorizes" the input of this stage, mainly to select and memorize the input x^t. Record what is important and write less about what is not. The current input content is represented by the z calculated above in equation (8). The selected gating signal is controlled by z^i (i stands for input). Add the results obtained in the above two steps to obtain the c^t that is transferred to the next state. This is equation (7), above. Finally, output stage: this stage will determine which will be regarded as the output of the current state, with the output stage mainly controlled by z^o, also containing the c^t obtained in the previous stage (changes through a tanh activation function). This process completes the input and output of a time-step LSTM. The main models in this paper will use LSTM as the basic unit.

2.2.2. Encoder and decoder model

The encoder and decoder model was first proposed in the machine translation model [31]. Since the translation task requires the input and output of the network to be a sequence, it is difficult for the dense deep-network structure to handle this type of task. Based on this difficulty, the encoder and decoder model was proposed and gradually extended to time series tasks other than the field of machine translation. There are many options for encoders and decoders, and recurrent neural networks (such as LSTM and GRU) are usually used. Moreover, using LSTM as the encoder and decoder, the process of the entire model will be explained in more detail.

Fig. 2 shows the encoder and decoder method using LSTM in traditional translation tasks.

First, the words are transformed into word representation vectors through the embedding layer [41]. These vectors enter the LSTM of the encoder in order. For simplification, only the transfer of hidden layer states in LSTM is shown in the figure. After encoding sequentially, the encoder outputs the context vector c (usually the hidden state h in the last time step) and passes to the decoder and inputs this context vector at each time step of the decoder. This context vector is considered to contain the context information of the sentence to be translated.

In the decoding process, the context vector of the encoder is used as the initial state to initialize the decoder. In addition, the context vector is also input into each decoding step. Furthermore, the predicted result of the previous time step is also input to the next time step, thus realizing the inference process that can continuously output the predicted result.

Above all, using the LSTM in the encoder and decoder model, the whole information will be time dependent. However, when we fall into the field of building energy consumption prediction, traditional encoder and decoder learning is not suitable, mainly because there is no additional input information in the entire decoding process, which is reasonable in the translation task since all the information is already contained in the original language sentence. However, in the field of energy consumption, as time steps go by, researchers can fully obtain accurate information about the previous time steps, so it is necessary to design more reasonable encoder and decoder processes for time series prediction.

Fig. 3 shows the improved encoder and decoder modes used in this task (building energy consumption prediction). There is no difference between the encoder part of this model and the machine translation model. In the encoder process, T_0, T_1 and T_2 represent the input time series at each time step. The input in the encoder contains all the features' time series and the target time series to be predicted. The state vector c input from the encoder to the decoder uses the state of the last time step in the encoder and is considered to contain all the information input by the encoder. Since the model involves LSTM, the input process will be time-dependent, which may be more suitable for the property of time series.

However, the decoder part outputs only the prediction result of one time-step, and the decoder part also adds a step of inputting additional information. The additional information dimensions of the decoder input can be freely decided. Thus, it is completely possible to avoid inputting features such as prediction of future weather and input only some known truth values. Moreover, the input of the decoder can consider more static known characteristics such as whether tomorrow is a holiday or a working day, to better improve the prediction effect of the model.

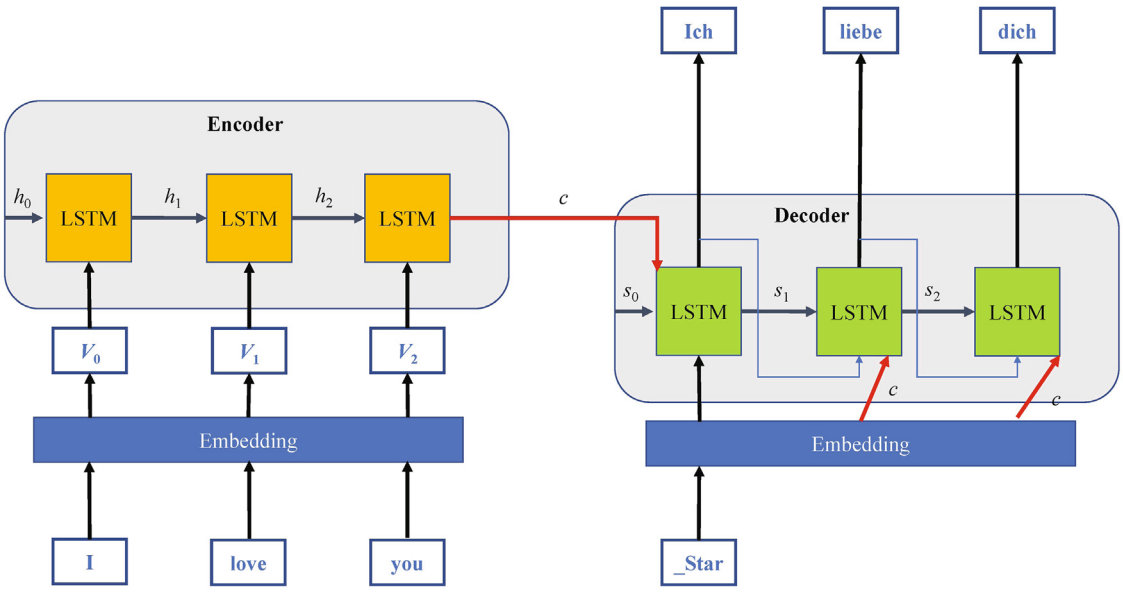

Fig. 2. Structure of encoder and decoder model using LSTM.

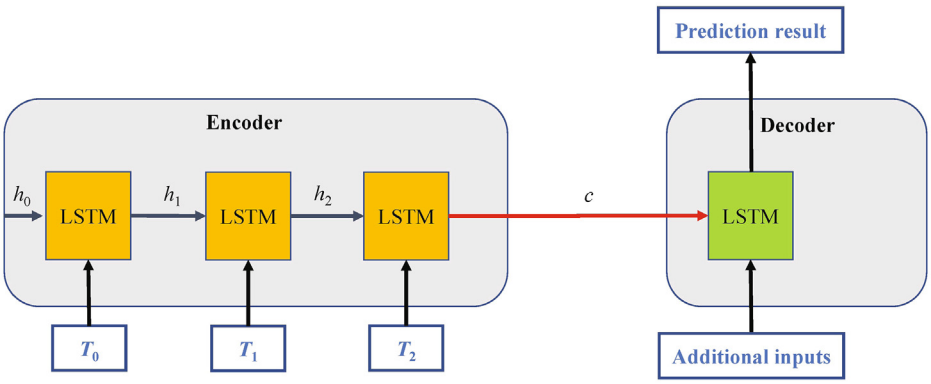

Fig. 3. Improved encoder and decoder model for this research.

At the same time, since the decoder can only output the result of one time-step in the future, when performing the multistep prediction process, it is necessary to use the prediction result of the previous time step and other collected real features of the encoder for prediction. Thus, each prediction step (data) needs to be encoded and decoded in multistep prediction, which may be investigated in future.

2.2.3. Attention

The attention mechanism was first proposed by Ref. [42] to determine whether each word in the translation result can correspond to the word in the original sentence. In 2017, Google research [37] thoroughly carried forward attention. At present, attention research has also expanded to more fields. The wide application of the attention mechanism stems from its relatively simple formula expression. As long as the 'Query (Q)', 'Key (K)', and 'Value (V)' of the current scene are clear, users can easily use attention to improve their model performance, allowing attention to be applied easily to a variety of different scenarios. The general expression of attention is as follows.

$$att(q, X) = \sum_{i=1}^{N} \alpha_i X_i \tag{9}$$

$$\alpha_i = \text{softmax}(s(key_i, q)) \quad = \quad \text{softmax}(s(X_i, q)) \tag{10}$$

The essence of the attention mechanism is actually an addressing process, as shown in the figure above (Fig. 4). Given a task-related query vector q, the attention value is calculated by calculating the attention weight of the key and attaching it to the value. The main process can be divided into three steps.

(1) information input, $X = [X_1, \ldots, X_n]$ represents N input information;

(2) Attention weight calculation: let Key = Value = X, then use the query vector to calculate by equation (10), where $s(X_i, q)$ refers to the score function. Traditional attention uses a simple dot product for calculation ($s(X_i, q) = X_i^{\mathrm{T}} q$), but the score function in this research uses a neural network for calculation ($s(X_i, q) = V^{\mathrm{T}} \tanh(WX_i + Uq)$). The result of the score function is converted into a probability form through a softmax function, which is the attention weight α_i that needs to be calculated in equation (10).

(3) The information-weighted average and attention weight α_i can be interpreted as the degree to which the information is considered when querying q in the context. A "soft" information selection mechanism is used to encode the input information X as equation (9).

2.2.4. Encoder and decoder architecture with attention

By using the improved encoder and decoder model, the process of energy consumption prediction introduces time dependence in encoding and free definition of the dimensions in decoder input. However, the improved encoder and decoder model is still a pure black box model. Based on the unique scoring mechanism and attention value in the attention calculation, this research will integrate the attention mechanism into the model to improve the interpretability of the encoder and decoder model.

In the calculation process of the encoder, the input time series will be sequentially input into the encoded LSTM in chronological order, and the LSTM will give a hidden state output h at each time step. Since the h output at each time step mainly represents the current date information, a naïve idea is to use the h of each time step to calculate the final output context vector c of the entire encoder. Compared with the traditional encoder and decoder model that uses the hidden state of the last time step directly as the encoder output, using h at different time steps for the context vector can effectively avoid the problem of forgetting caused by the excessively long encoding sequence. More importantly, the calculation of the weight vector in the process can easily be extracted and used as a key result of model interpretation. Based on the discussion mentioned above, this study uses the attention mechanism to calculate the context vector output by the encoder [42], and the process is shown in Fig. 5.

Compared with the traditional encoder and decoder method, the attention mechanism performs a weighted summation of each hidden layer state in the encoder and inputs it into the decoder. As shown in Fig. 6, $\oplus$ represents the calculation method of weighted summation; ∂ represents the weight of different hidden layer states, and the calculation formula is shown below (taking the current time step t as an example).

$$e_j = score(s_{pre}, h_j) \tag{11}$$

$$\partial_j = \frac{\exp(e_j)}{\sum_{k=1}^{T} \exp(e_k)} \tag{12}$$

where s_{pre} represents the hidden layer state of the previous time step of the current prediction step, and h_j is the hidden state of each time step in the current LSTM input sequence. These two variables calculate the score value of the hidden layer state at the current time step through the score function. In this study, a fully connected neural network is used as the score function. After obtaining the score e_j, the score is normalized in the time dimension by equation (12) to obtain the final attention weight. The attention weight is used to calculate the hidden layer state of the final output of the encoder, as shown in equation (13).

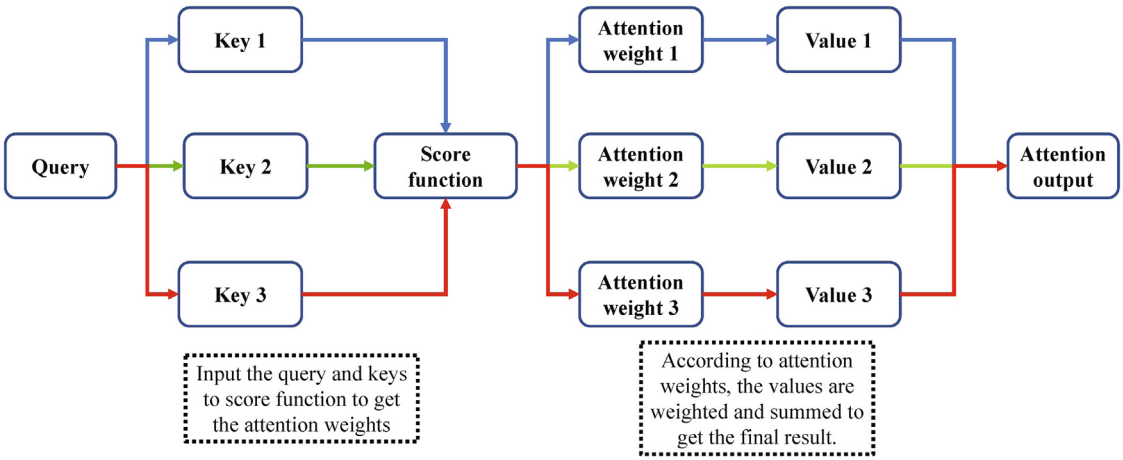

Fig. 4. Attention addressing process.

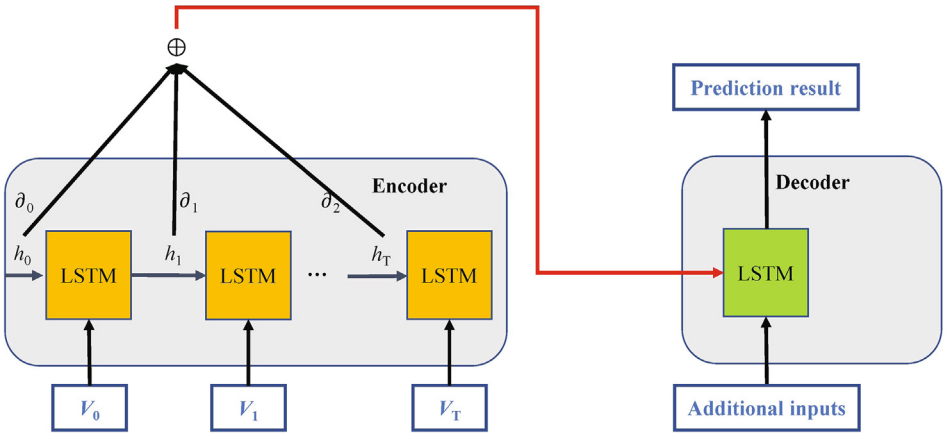

Fig. 5. Encoder and decoder with attention.

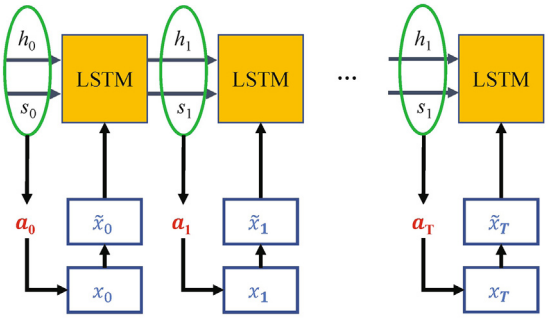

Fig. 6. Feature attention diagram (encoder).

$$h_{attention} = \sum_{j}^{T} \partial_j h_j \tag{13}$$

The final attention state will also be input to the decoder as a part of the initial vectors; therefore, the value of the weight also indicates the degree of emphasis on different hidden layer states in decoding and forecasting.

2.2.5. Encoder and decoder architecture with attention in double stage

Currently, with the rapid development of intelligent measurement tables and energy monitoring platforms, the prediction of building energy consumption is often not a univariate regression model but a multivariate model involving multiple external variables. However, due to the high cost of data collection and measurement, the discussion of the importance of input features will be extremely important. These more important features will be measured and collected first in the construction of the energy consumption platform.

Deep-learning models can handle high-dimensional data well, but dense neural networks do not have a good explanation for the importance of different feature inputs at each time step. Based on the above model, to enrich the interpretability of the model, this research also performed attention calculations at the input feature level [9,10] to indicate the importance of different features in different time steps [43,44].

First, this part describes and calculates the encoder part of the network. The attention calculation of the feature is completed mainly in the encoder stage, assuming that the encoding stage needs to input n external variables of length T (equations (14) and (15)). These equations actually represent a multivariate time series, which has two dimensions: the time dimension T and the space dimension n. equation (14) represents the expansion of the space dimension, and equation (15) represents the expansion of the time dimension.

$$\begin{pmatrix} x_1^1 & \cdots & x_T^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_T^n \end{pmatrix} = \begin{pmatrix} x^1 \\ \vdots \\ x^n \end{pmatrix} \tag{14}$$

$$\begin{pmatrix} x_1^1 & \cdots & x_T^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_T^n \end{pmatrix} = (x_1 \cdots x_T) \tag{15}$$

When using RNN to deal with time series, all spatial dimensions are equally weighted. The natural idea is that the different spatial dimensions could be given different weights, which can be calculated by the feature attention mechanism (Fig. 6)

First, the LSTM is still used as the basic block for the encoder. In addition, the attention mechanism is used to calculate the attention scores at different time steps and different feature dimensions. The specific calculation formula is as follows.

$$e_t^k = v_e^T \tanh(W_e[h_{t-1}, s_{t-1}] + V_e x^k) \tag{16}$$

where x^k represents the k[th] dimension time series (length is T); h_{t-1} and s_{t-1} represent the output of the last time step of LSTM; and W_e, U_e, and V_e are the parameters that the neural network needs to learn. In other words, a neural network is also used here as a scoring function for calculation. The final result e_t^k represents the score of the dimension k in the t time. Because it is necessary to ensure that the sum of the weights is 1, it is necessary to perform softmax processing on e_t^k in the time dimension.

$$a_t^k = \frac{\exp(e_t^k)}{\sum_{i=1}^n \exp(e_t^i)} \tag{17}$$

The a_t^k result obtained represents the weight of all input features at time t, and the sum of its spatial dimensions is 1, that is, $a_t^1 + a_t^2 + \cdots a_t^n = 1$. Finally, the attention weight obtained is applied to the original input vector x , and the new input $\tilde{x}$ is obtained at the current time step for LSTM, as shown in equation (18). Therefore, the original input vector does not directly enter the LSTM network.

$$\tilde{x}_t = \begin{pmatrix} a_t^1 x_t^1 \\ a_t^2 x_t^2 \\ a_t^3 x_t^3 \\ \vdots \\ a_t^n x_t^n \end{pmatrix} \tag{18}$$

Furthermore, to demonstrate the weights of different input features, the input of the encoder at this time will not contain the target sequence to be predicted, and the target sequence will be processed in the decoder.

Different from the previous decoder structure of the single attention model, the decoder of the two-stage model focuses on processing the target sequence. Therefore, the decoder uses a different LSTM from the encoder and needs to decode and calculate at the same time step. The specific process is shown in Fig. 7 below:

The purpose of the decoder is to take advantage of *h*t from the ‘‘external variables” in the previous encoder time series to obtain *ct* by applying unequal weights. Additionally, the connection between the encoder and the decoder is established by using the hidden state of the encoder. Through another LSTM network, the decoder process will combine with the input for decoding. The core decoding attention equations are as follows (equation (19)):

$$l_t^k = v_d^T \tanh(W_d[h'_{t-1}, s'_{t-1}] + U_d h_i) \tag{19}$$

W_d, U_d, and v_d are the parameters that need to be learned in the decoder attention network, and $[h'_{t-1}, s'_{t-1}]$ is the output of the last time step LSTM. *hi* is the hidden state of the corresponding time step in the encoder. The calculation l_t^k result indicates the importance of the k[th] hidden state of the encoder in the t[th] step of the decoding process. Similarly, we need to perform softmax processing on l_t^k; at this time, the softmax needs to be calculated in the time dimension. The processed result performs a weighted summation on the hidden layer state in the encoder to obtain the final context vector *c*, as shown in equations (20) and (21).

$$\beta_t^i = \frac{\exp(l_t^k)}{\sum_{j=1}^T \exp(l_t^j)} \tag{20}$$

$$c_t = \sum_{i=1}^T \beta_t^i h_i \tag{21}$$

After obtaining the context vector, the current step of the decoding concatenates the original target sequence value and the state vector together and inputs them into a neural network to obtain $\tilde{y}$. At this time, $\tilde{y}$ will be input into the LSTM of the decoder for iteration until the entire decoding sequence End. After the decoding of the original target sequence is completed, this research inputs the context vector hidden state and additional information of the last time step into the final output neural network to obtain the final prediction result.

2.2.6. Transformer

After the attention mechanism was proposed [31], its main application scenarios were used in conjunction with recurrent neural networks [9,10]. However, because the recurrent neural network cannot perform parallel calculations, the processing of the sequence can only be performed iteratively, the calculation efficiency is not very high, and it will be slow when processing long-distance sequences. In 2017, Google proposed using the self-attention mechanism and transformer [37] to replace the recurrent neural network for encoding and decoding modes. Based on the good visualization characteristics of the attention mechanism and the advantages of parallel computing, this study builds an energy consumption prediction model based on the transformer model. The specific model process is shown in the figure below.

The left side of Fig. 8 shows the encoding process of the model, the right side shows the decoding process of the model, and the gray part shows the encoder and decoder parts. During the calculation process, the number of encoders and decoders can be stacked to improve the model complexity.

The input data that have been improved in dimensionality will enter the encoder for processing. The core of the encoder is the multihead attention module, and the calculation method of attention has been mentioned in the previous chapter; however, the encoder of the transformer model uses the self-attention method, in which the ‘Query’, ‘Key’, and ‘Value’ are all the same (equation (22)).

$$Attention(Q, K, V) = Attention(X, X, X) \tag{22}$$

The encoder of the transformer will perform self-attention calculations between the input of each time step and the input of other time steps and replace the original input vector with the calculated attention vector. The specific workflow is shown in the figure below (Fig. 9).

When calculating the attention vector (Attention [1]) of the first-time step, q^1 will be calculated with k of each time step to obtain the attention weight ∂. Then, the weighted sum of v in each time step will obtain the attention output. Consequently, after the first self-attention, the original input $[Input^1, \cdots Input^T]$ becomes $[Attention^1, \cdots Attention^T]$. Then, the encoder uses a feedforward neural network and two residual connections to obtain the final encoder output. In addition, layer normalization is used for better optimization results. To facilitate the stacking of multilayer encoders, the dimensions of the input data and output data of the encoder are often the same. From the calculation structure of the model,

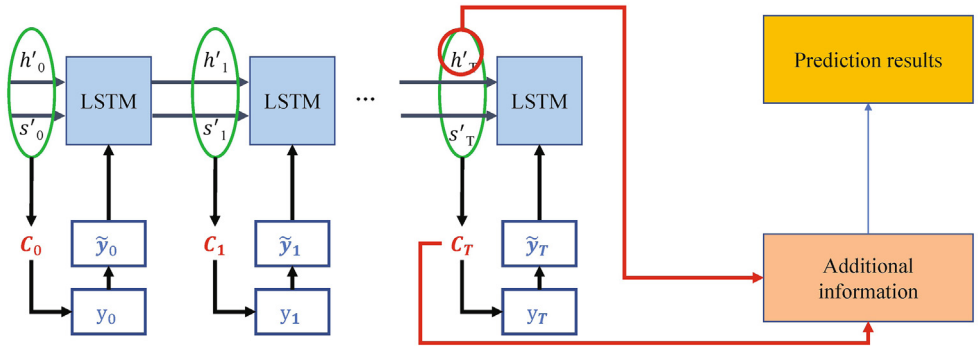

Fig. 7. Feature attention diagram (decoder).

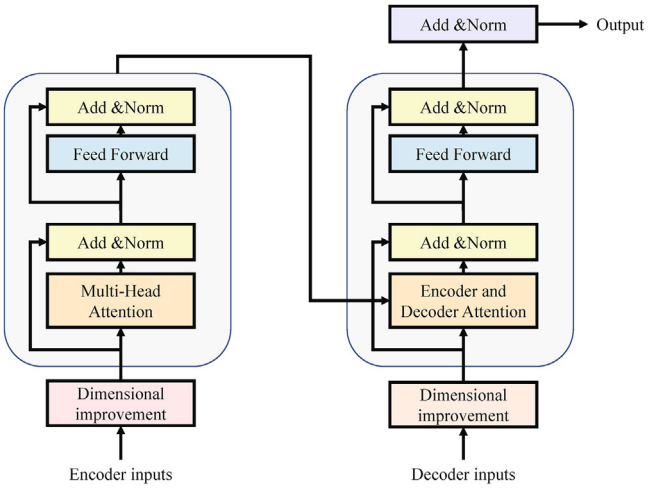

Fig. 8. Transformer workflow.

each time step will have a direct connection with other time steps without any loss due to the excessively long sequence (which is very strict in RNN). Compared with recurrent neural networks, this coding structure can better preserve the forward and reverse relationships between each time step.

The structure of the decoder in the transformer is very similar to the structure of the encoder. However, because this target is a single-step prediction model, there is only one time-step input in the decoder, which leads the self-attention part to be unnecessary in the decoder. Moreover, the encoder attention results are used in the decoder, which can input the result of the encoder into the decoder (equation (23)).

$$Attention(Q, K, V) = Attention(Decoder, Encoder, Encoder) \quad (23)$$

The above equation shows that the attention calculation in the decoder assumes that Query is the input of the decoder, and Key and Value are from the encoder. This calculation can obtain the relationship between the encoder input and the decoder, which is very similar to the attention calculation using LSTM.

At the same time, the calculation of the encoder and the decoder adopts the form of multiheaded attention. That is, one attention calculation is divided into several small parts of the same size for simultaneous calculation. This method not only speeds up the calculation but is also a framework for an ensemble model because the multiheaded form will also produce different results during the attention calculation.

3. Case study

This section describes a case study in which we applied all the models mentioned in the Methods section. The expectation was as follows: first, with the help of the encoder and decoder architecture, the prediction of future energy consumption can freely define the input feature dimension in decoder; in addition, the attention

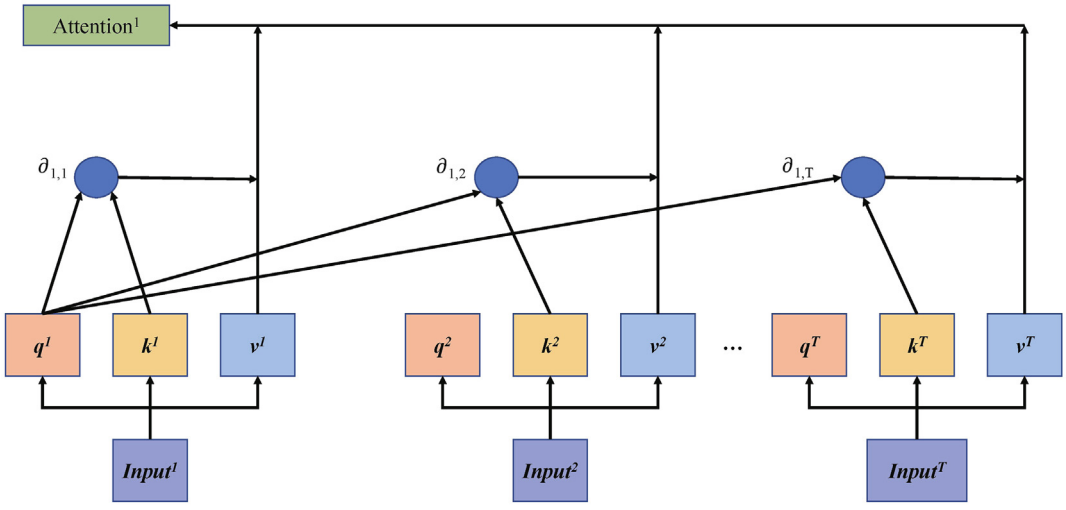

Fig. 9. Self-attention workflow.

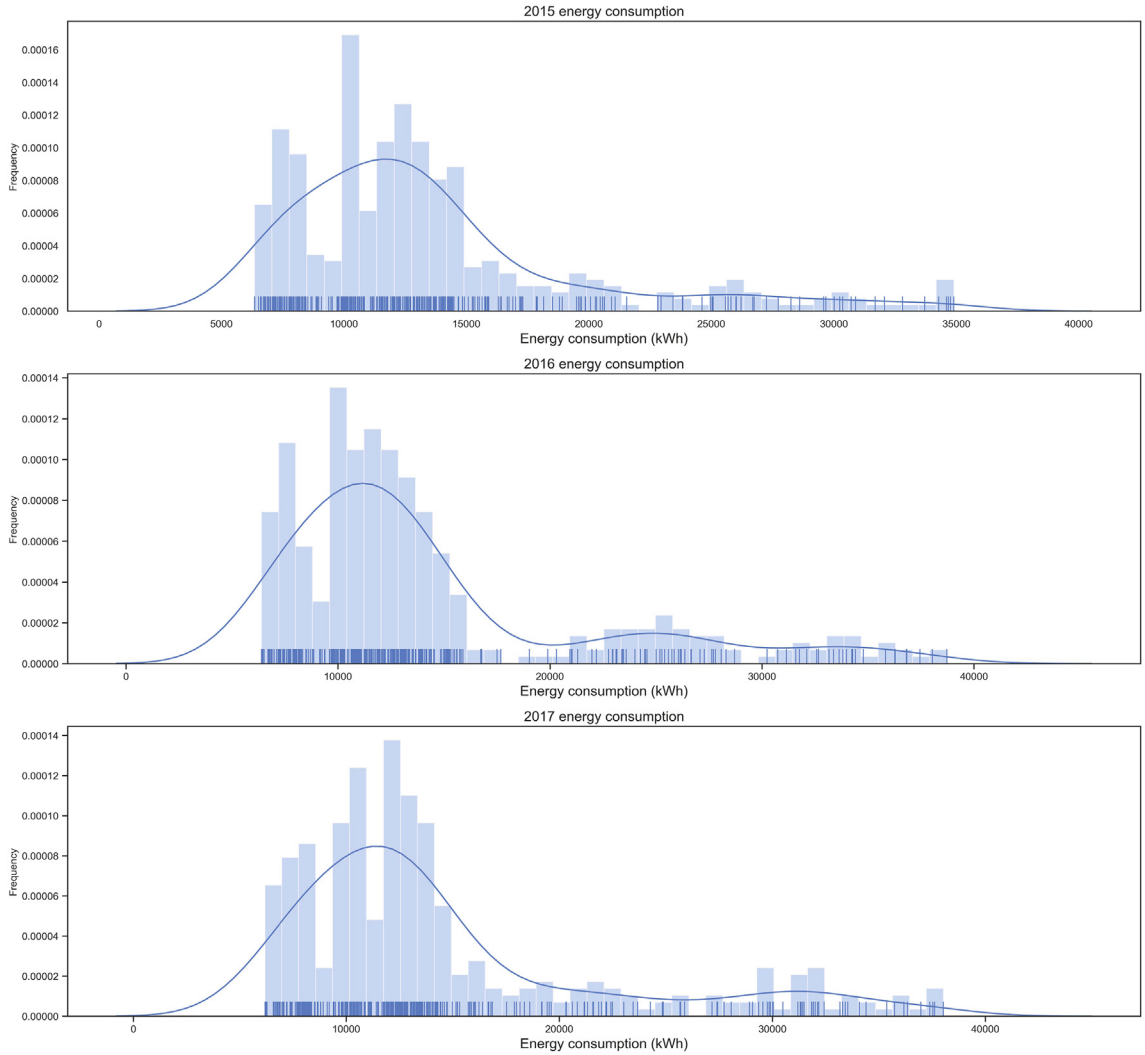

Fig. 10. Frequency histogram and kde for each year of target building in the case study.

mechanism is used in different stages of the model to enhance the interpretability of the model calculation process.

3.1. The introduction of building and dataset

We used a government office building from Qingdao city, Shandong Province, China. The building energy consumption data were gathered by the government platform. The energy consumption of the building is composed mainly of the following parts: (1) daily equipment electricity usage, (2) lighting usage, and (3) consumption by the air conditioning system in summer (since the building is located in northern China, the winter heating demand is covered by the local government). Therefore, all the data in this article are electricity consumption data. In addition, as a government office building, the use of the entire building usually abides strictly by the government vacation policy.

The datasets contained three years of daily data from January 1, 2015, to December 31, 2017, on a time scale. In the dataset, there are eleven features, including ten numeric features (mean temperature, maximum temperature, minimum temperature, mean humidity, maximum humidity, minimum humidity, dew point, mean atmospheric pressure, mean wind speed, maximum wind speed) and a categorical feature (holiday, which is a binary feature denoting whether it is a holiday) as well as the label energy consumption. Fig. 10 shows the frequency histogram of building energy consumption in each year. The curve in the figure is the kernel density estimation of the annual distribution.

The figure demonstrates that the building has a different histogram for three years. In particular, the peak energy consumption in 2016 and 2017 was significantly higher than the peak energy consumption in 2015. However, the kernel density estimates (kde) for each year have relatively similar distributions.

Based on the content of the method chapter, whether it is a LSTM or a transformer, a special format is required for input, usually a tuple of (batch, length, features). Considering the lack of current energy consumption data, the dataset needs to be processed more carefully. Accordingly, data augmentation has been proposed as a technology for increasing the quantity of data trained to improve the generalizability and robustness of a model [45]. From the definition of LSTM, the time step of the input sequence can be different for each calculation, but considering the speed of batch training, it is still necessary to reshape all the data to the same time step. Fig. 11 shows how this method works. In the figure, the raw training data are the original data for training, which are in a sequence of length m. This m is determined by the length of the data collected in sequence, which may be very large and is basically the time span of the training data such as 365 (1 year) or 730 (2 years). The sample window denotes the sliding window

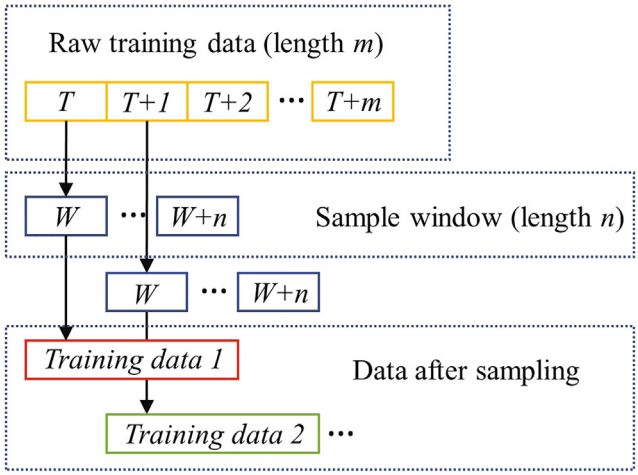

Fig. 11. Sampling method for data augmentation.

for the sampling process and has length n, usually $n < m$. This n is the length of the sequence input into the model.

This processing method adds a large number of training samples under the premise of repeated sampling. Take the data of 1 year and the sampling length of 7 days as an example. In this case (m = 365, n = 7), if every sample is not sampled repeatedly, the number of training samples will be $T_1 = m/n = 52.14$. At this time, there was an indivisible situation. After discarding the last indivisible part, only 52 samples were available for training. However, after repeated sampling, as shown in Fig. 11, the number of training samples will be $T_2 = m - n = 358$. There is no data waste due to indivisible data, and the number of training samples has also been greatly increased. As long as $T_2 > T_1$, this technology will certainly increase the number of training samples. As n will normally be much less than m, the number of training data points will definitely be increased. In addition, this repeated sampling also invisibly adds different perspectives to the data, which is also an important improvement for the diversity of data.

3.2. Experiment setup

The evaluation was performed on the results from different models to assess the effectiveness of encoder and decoder models.

First, due to the limitation of the dense neural-network structure, the entire calculation process of the model has no time dependence. So, all the models in this paper will use LSTM and transformer to capture time dependence. Furthermore, the prediction processes used future features. Some characteristics are certain and knowable (whether tomorrow is a holiday or not), but for some characteristics, we could not obtain the true value (temperature, wind speed, humidity, etc.). However, when using the real weather data, the prediction results are ideal because in the real world where we could get absolutely accurate weather forecasting, there is still some error in the prediction of meteorological parameters, and this part of the error is not usually considered [46]. Thus, the model proposed in this article uses mainly the encoder and decoder structure described above. The known sequence in the encoder uses the designed additional information module in the decoder to input the information determined in the future (whether it is a holiday), which will be compared with the model including real future features. On this basis, the attention mechanism is used in the encoder and decoder to enhance the visualization and interpretability of the model prediction process.

To assess the prediction results, we defined five models for comparison:

- **M.1**: Model 1 uses the encoder and decoder structure and two LSTM networks as the encoder and decoder in a divided manner. However, the true value of the future weather is still entered in the additional information module of the decoder. The role of Model 1 is to provide the best baseline, that is, the prediction results in the case of using the time-dependent model and the true value of the future weather. We hope to be as close to the prediction accuracy of Model 1 as possible without inputting future weather data.
- **M.2**: Model 2 uses the same encoder and decoder structure as M.1, but the difference is that only information that is known in the future is input into the decoder. Although Model 2 lacks much information input compared to M.1, we still hoped that the structure of the encoder and decoder can make the model achieve relatively good prediction accuracy. In this case, the error caused by inputting the predicted value is avoided.
- **M.3**: Model 3 uses a structure similar to Model 2, that is, both use LSTM as the encoder and decoder and only input certain known information as the decoder input. The only difference is that Model 3 uses the attention mechanism to calculate a new attention vector in the encoder as the context vector and inputs it to the decoder. Model 3 hopes to make the prediction process more explanatory by visualizing the calculation results of attention, that is, we can know which hidden layer state plays a more important role in the prediction.
- **M.4**: Model 4 further adds attention calculation for features based on Model 3, forming a two-stage attention model. To avoid the influence of the target sequence, the model uses an encoder and a decoder to process the external sequence and the target sequence, respectively. Model 4 can obtain the weights of different features in different time steps of the previous encoding during prediction.
- **M.5**: Model 5 uses the self-attention mechanism completely instead of the LSTM for encoding and decoding. The self-attention weight in the encoder can obtain the relationship between the two in the encoded sequence. The attention weight in the decoder is similar to Model 4. Due to the use of the multi-head attention mechanism, the result of the weight has a variety of different perspectives.

In addition, we hope to obtain richer conclusions by comparing the results among the models. Compared with M.1, M.2 only removes the input of future weather parameters when the model structure is similar. By comparing the results of M.2 with M.1, we can determine whether the encoder and decoder structure can obtain better prediction results without inputting future weather parameters. The model starting with M.3 focuses on building an interpretable deep learning model. The analysis will focus more on the visualization results of the attention rather than the prediction accuracy.

We used the Python 3.7 programming language and the PyTorch 1.2 deep learning framework to implement our model. All the models used the dropout mechanism to prevent overfitting [47]. Furthermore, each model optimized the hyperparameters. Based on hyperparameter optimization, the number of LSTM units and the number of training rounds between different models are as much the same as possible to reduce the uncertainty between the comparison of different models as much as possible. The specific network structure and hyperparameter settings of all models are shown in the appendix (Table 8 and Table 9).

The prediction process provided a one-year prediction in 2017, for which the training data were from 2015 to 2016 in the dataset. In this case, we have 2 years of daily data for training. The daily data contain 11 features and a label. Considering the research goal of office buildings, we set the encode length of all models to 7. The data dimensions of different models during training are shown in the following table (Table 1).

Table 1
Training data dimension.

Model ID	Encoder Input	Decoder Input	Note
M.1	(*, 7, 12)	(*, 1, 11)	All features are entered in the decoding
M.2	(*, 7, 12)	(*, 1, 1)	Only the holiday information was entered when decoding
M.3	(*, 7, 12)	(*, 1, 1)	Same as M.2
M.4	(*, 7, 11)	(*, 7, 1)// (*, 1, 1)	The decoder needs to process the energy consumption sequence and enter it as holiday information
M.5	(*, 7, 12)	(*, 1, 1)	Same as M.4

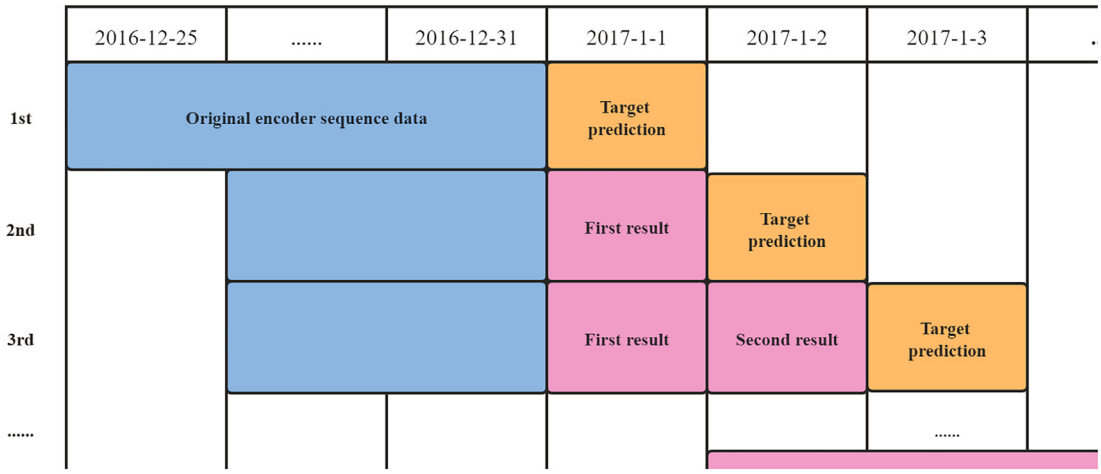

Fig. 12. Rolling prediction step.

The minibatch training method [33] is used in the training process, and * represents the number of data points in each batch. Each encoder and decoder input will output the energy consumption data of the next day (that is, the '8th' day after encoding 7 days) of the encoding sequence and perform gradient descent to update the parameters in the network.

After the training, the model needs to predict the energy consumption of the building in 2017 However, encoder and decoder models are a single-step prediction model, and each prediction step also requires all the features and energy consumption data of the previous 7 days, so this paper uses rolling prediction to output the forecast of energy consumption throughout the year, as shown in Fig. 12.

Since each prediction requires data from the previous 7 days to be encoded, the model needs to start coding from December 25, 2016, and predict the energy consumption on January 1, 2017. Since the 2016 data belong to the category of training data, the features and labels are true values, as shown in the blue block in the figure. After the first prediction, the predicted value of energy consumption in 2017 was obtained. At this time, the required 7-day window for encoding moved accordingly. The encoded data need to be from December 26, 2016, to January 1, 2017, and need to be used to predict the energy consumption on January 2, 2017. After the first prediction, the predicted value of energy consumption on January 1, 2017, was obtained. At this time, the required 7-day window for encoding moved accordingly. The coded data need to be from December 26, 2016, to January 1, 2017, and used to predict the data on January 2, 2017. At this time, the data on January 1, 2017, need to be encoded, and the feature part uses the real weather data and holiday information of the day. Since the day has ended, the true value of the weather parameters of the day can be obtained easily. The energy consumption data are encoded using the results obtained from the previous time step prediction without using real energy consumption data. In this case, the single-step prediction model of the encoder and decoder can also output the results of one year at a time, and the comparison with the results of Model 1 is more convincing. The steps of encoding using the previous prediction results are represented by pink blocks in the figure. Over time, 365 predictions were finally completed, and the final result was obtained.

Model 1 was expected to obtain the best accuracy because Model 1 not only introduces time dependence but also inputs future real weather data into each forecast. Models 2, 3, 4 and 5 introduce time dependence through LSTM or self-attention mechanisms and only input known holiday information in each prediction step. These models hope that through a special encoding and decoding structure, they can obtain better prediction results without inputting weather data that need to be predicted. Although Models 3, 4, and 5 add an attention mechanism, they do not necessarily increase the accuracy of model prediction but only increase the interpretability of the model prediction process.

4. Results and discussion

4.1. Evaluation method

For this research, the mean absolute percentage error (MAPE) and the coefficient of variation of the root-mean-square error (CV-RMSE) were selected to assess the prediction accuracy. CV-RMSE has the same dimension as MAPE, but the result of CV-RMSE is usually slightly larger than the result of MAPE because CV-RMSE first accumulates the square and then extracts the square root, which enlarges the gap between larger errors. Therefore, CV-RMSE has a positive meaning for evaluating the maximum error of prediction. In addition, the mean absolute error (MAE) for each forecast time step is also calculated for analysis. MAE, MAPE, and CV-RMSE are calculated as follows (equation (24) to equation (26)):

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|yi - \hat{y}i|}{yi} \tag{24}$$

$$CV\ -\ RMSE = \sqrt{\frac{\sum_{i=1}^{N}(yi - \hat{y}i)^2}{N}} \Big/ \frac{\sum_{i=1}^{N} yi}{N} \tag{25}$$

$$MAE = \frac{|yi - \hat{y}i|}{N} \tag{26}$$

where yi is the actual value, $\hat{y}i$ is the predicted value, and N is the number of the observations.

Table 2 shows the results for the six models. For this table, the average MAPE and CV-RMSE values were calculated. In addition, for reference, the following linear figure (Fig. 13) shows the prediction results of different models, where 'gnd' refers to the original value of energy consumption. Fig. 14 charts the values given in Table 2 to provide a more direct perspective.

4.2. The overall prediction results at the statistical point

From the evaluation metric results for the models shown in Fig. 14 and Table 2, M.1 achieved the best MAPE and CV-RMSE results among all models, mainly because M.1 uses the encoder and decoder structure to introduce time dependence and inputs real weather data in the future, M.1 is also proposed as the best baseline for other models, and the results also prove this. Compared with M.1, M.2 only uses future holiday information as the input of the decoder. Compared with M.1, M.2 increased by 0.54 and 0.74 percentage points in MAPE and CV-RMSE, respectively. It can be seen that even if the real weather data in the future is removed, the prediction results have not deteriorated much. Therefore, even if deleting the input of uncertain data in the future, the model could still get good results.

On the whole, M.2 to M.5 all use a model structure similar to M.1 and introduce time dependence. However, the difference is that M.2 to M.5 input only certain known information when predicting the future (in this case, referring to future holiday information). In the model without inputting future weather parameters, the prediction result is also close to the result of M.1: compared with M.1, the largest gap in MAPE is only 1.28 percentage points, and the minimum gap is 0.07 percentage points, while the maximum gap of CV-RMSE is only 1.52 percentage points, and the minimum gap is 0.35 percentage points. The encoder and decoder structure can still obtain very good prediction results without entering future weather conditions. In addition, due to the need to predict the whole year at one time, as the prediction time step

Table 2
Evaluation metric results for model predictions.

Model	M.1	M.2	M.3	M.4	M.5
MAPE	0.0773	0.0827	0.0901	0.0817	0.0780
CV-RMSE	0.1243	0.1317	0.1323	0.1278	0.1395

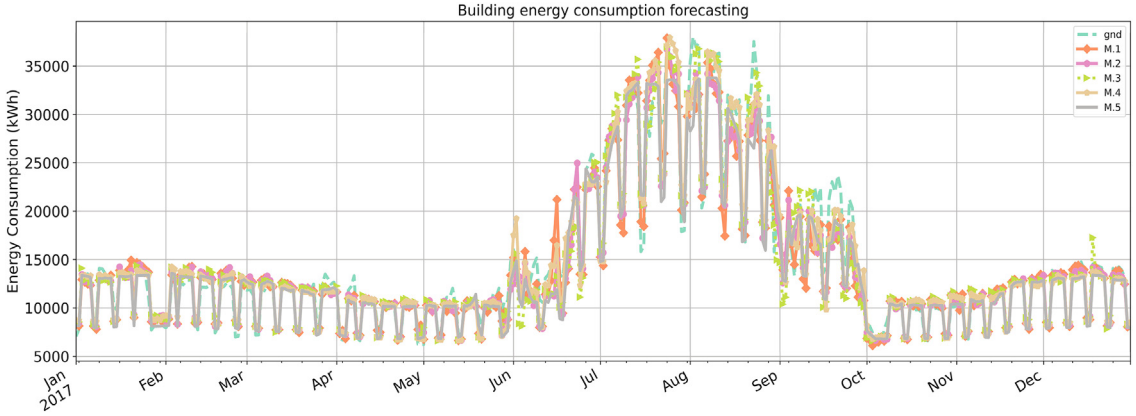

Fig. 13. The results of different models.

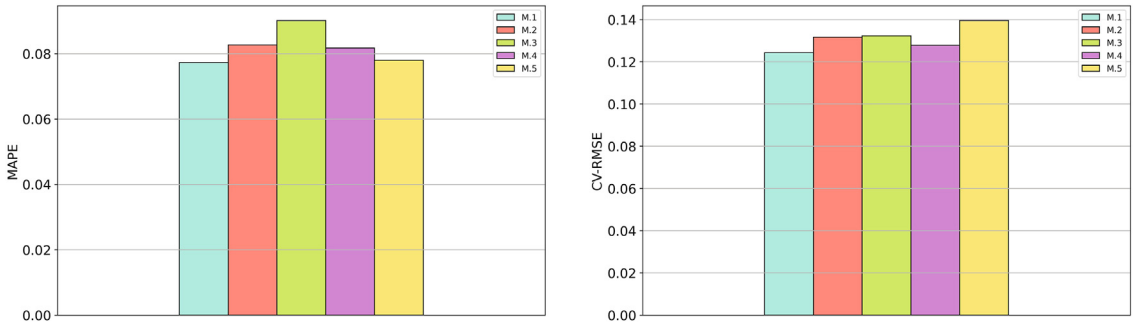

Fig. 14. Comparison of energy consumption prediction errors in each model.

progresses, the energy consumption in each coding sequence is the predicted value of the previous time step, teacher forcing [48] is not used in M.1 to M.5, undoubtedly introducing a certain amount of error and noise. If teacher forcing is used to simulate actual application scenarios, the prediction results should be further improved.

In addition, compared with M.2, M.3 to M.5 have added different degrees of attention mechanism, and M.5 even changed the encoding and decoding mechanism of the model. The results show that the attention mechanism has not significantly improved the prediction effect of the model. Except for M.5, which has some hints on the MAPE index compared to M.2, the other models are close to the results of M.2 but still have a very slight gap, which may be caused by the characteristics of the data set used. Since this paper uses a government office building, its operating cycle often follows the 7-day operating cycle mode strictly, so the length of the code is selected as 7 days. Due to the short sequence length, LSTM has a better memory effect for 7-day sequences, and the attention model is equivalent to introducing additional model uncertainty, which may also lead to changes in the prediction results, but its overall accuracy is still on a horizontal line.

4.3. Comparison of single-step prediction errors

The result of 4.1 shows that the average prediction results of all models are relatively close. Therefore, in this section, the predicted MAE at each time step in the model is specifically calculated, as shown in Fig. 15.

Through the visualization of the prediction results at each time step, the overall outliers of M.1, M.2, and M.3 are seen to be small, and there are no particularly abnormal prediction results. The largest outlier appears in the prediction results of M.4 and M.5, but there is only one point in each model. Excluding a single abnormal point, the MAE deviation of M.4 is also close to the MAE deviation of M.1 and M.3. Overall, M.1 has the best MAE outlier performance, the variance of its error is the lowest among all models. This result is also reflected in the CV-RMSE. Due to more outliers, M.5 obtained the second worst CV-RMSE result. Although M.4 has a larger outlier, its overall MAE is still small, M.4 has the second smallest CV-RMSE accuracy. In general, although the remaining encoder and decoder models have some larger outliers, the overall error is still on the same level as the result for M.1.

4.4. Interpretable model results

M.3 to M.5 are all interpretable models in which there are different degrees of attention mechanisms to enhance the interpretability of the prediction results.

M.3 uses the attention mechanism for the hidden layer state in the encoder, that is, the prediction result of each time step has an attention score vector of length 7. This score vector represents the importance weight of the hidden layer state between different days in the encoding process 7 days before the current prediction step. Therefore, the final attention weight result can be spliced into a (365, 7) two-dimensional matrix. The algebraic sum of each row in the matrix is 1, which represents the encoding weight of the hidden layer state in the first 7 days when predicting a certain day in 2017. Table 3 shows the statistical information of this weight matrix. The column names in the table are arranged in descending order, indicating how many days before the current forecast time step the data segment is. In addition, to simplify the representation, some extremely small values are simplified to 0 in the table.

The table shows that the weight of the encoding is concentrated on the first day, the fourth day and the seventh day. Fig. 16 shows the annual visualization results of attention weights, it shows that most of the predicted results need to focus on the hidden layer state of the first day to the end. This result is completely reasonable because from the perspective of the LSTM calculation, the hidden layer state of the encoder on the last day actually contains all the information calculated in the previous 7 days, showing that the LSTM network can remember the sequence length of 7 days well.

There are still the most cases where the weight is evenly distributed to the fourth and first days from the bottom, and some weights are completely biased towards the fourth day. This situation is also predictable. When the network needs to focus more on the front end of the coding sequence, it is a good way to extract a hidden layer state from the middle part of the sequence. Evenly distributing weights on these two days can undoubtedly better memorize the overall sequence. However, there are still some weights assigned to the hidden layer state of the 7th day to the end. This situation is rare and concentrated in December, January and February of the heating season. The following will take the situation in January and month as examples for a more detailed analysis. Fig. 17 shows the attention results in January and February.

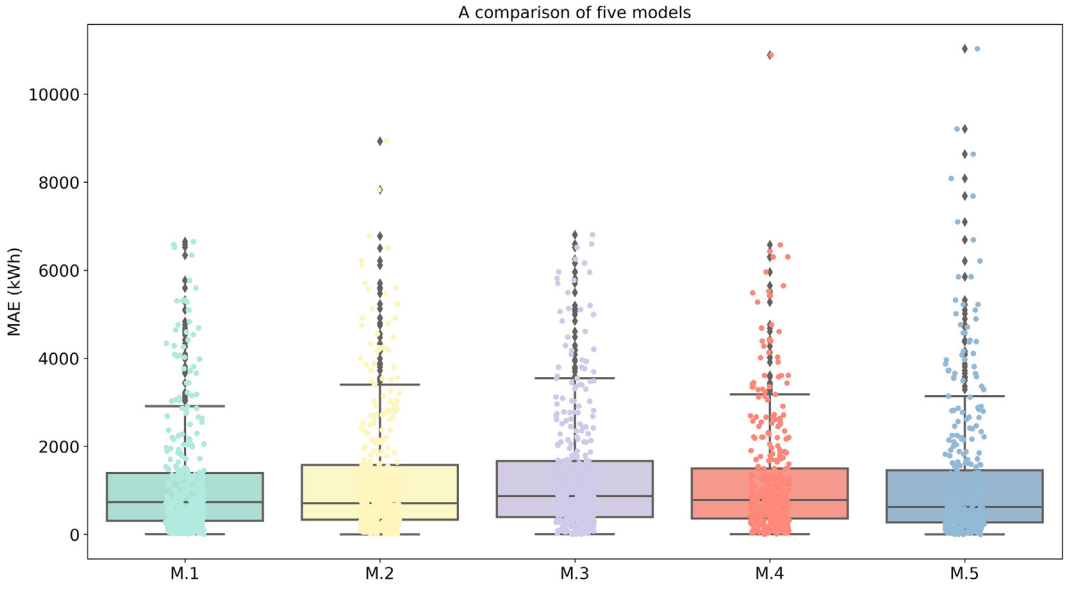

Fig. 15. Daily MAEs of different models in 2017.

Table 3
M.3 attention information.

Attention weight	Seventh	Sixth	Fifth	Fourth	Third	Second	First
Mean	0.0521	0	0	0.2464	0	0	0.7016
25%	0	0	0	0.0026	0	0	0.4275
50%	0	0	0	0.0649	0	0	0.9008
75%	0	0	0	0.4703	0	0	0.9939
Max	1	0	0	0.9952	0	0	1

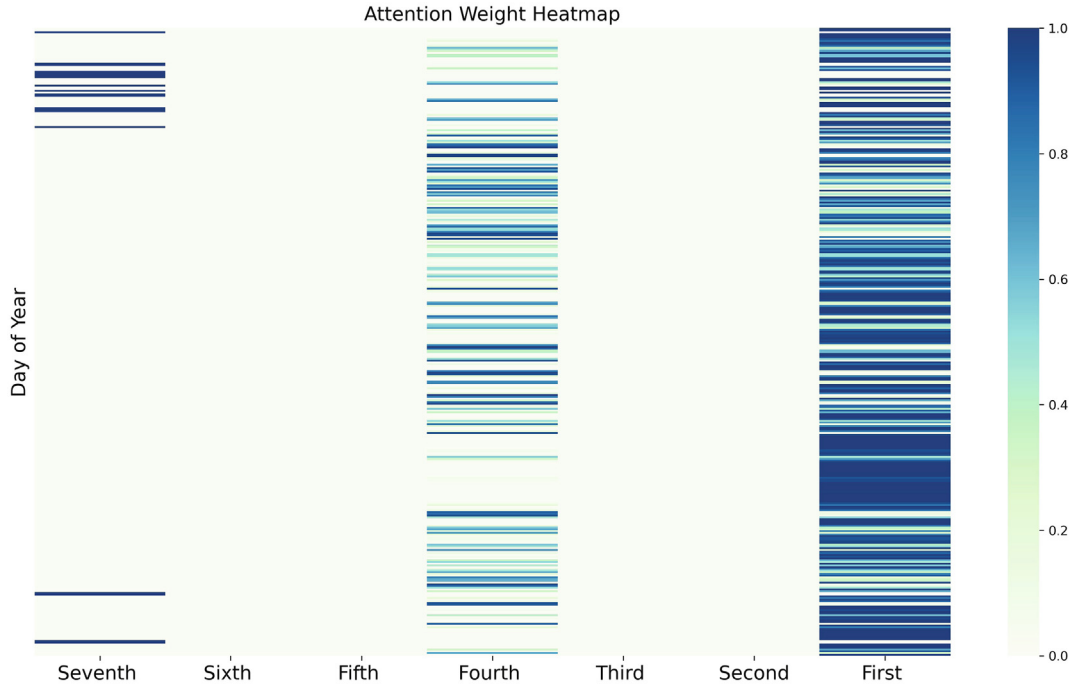

Fig. 16. Attention map for M.3.

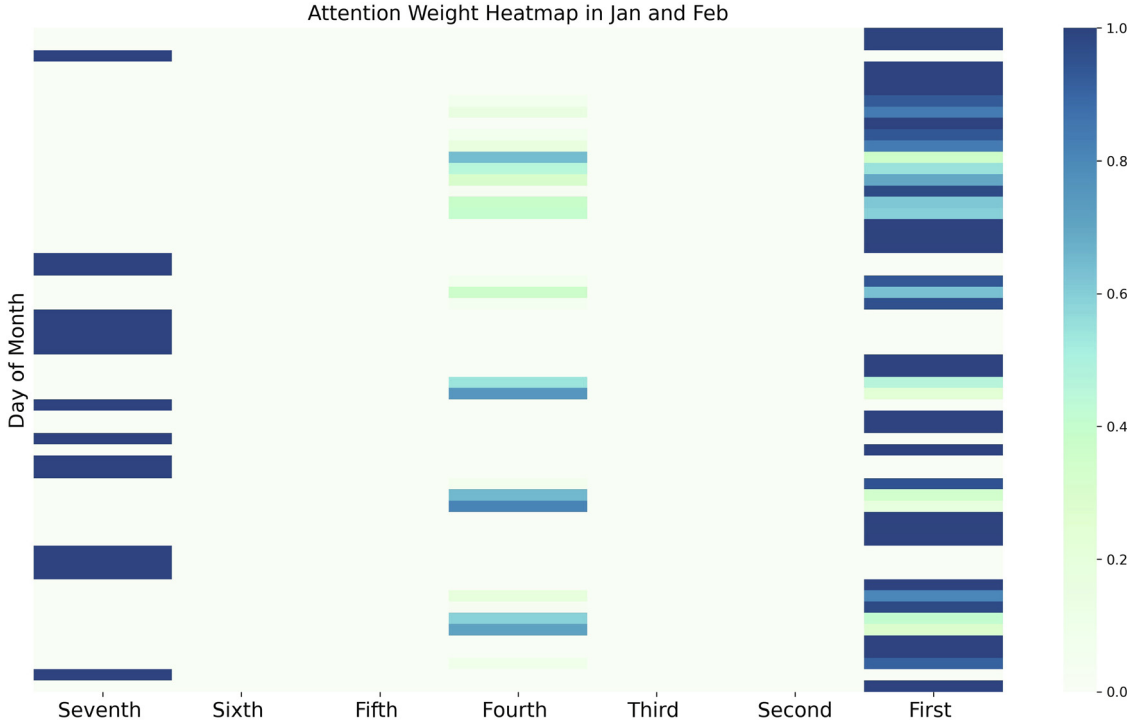

Fig. 17. Attention map for M.3 in Jan and Feb.

Table 4
Analysis of special circumstances from January to February.

Date	Note
3rd January	A special date due to the Gregorian New Year holiday.
21st to 22nd January; 26th to 29th January; 3rd February; 6th February	Special changes due to the Chinese New Year holiday
8th February	Forecast results higher than average error
9th February	Not yet

There are 15 days in January and February, where attention is focused on the 7th day from the bottom. Generally, the traditional Chinese holiday Spring Festival occurs between January or February according to the lunar calendar, and there are fixed Western New Year holidays in January. The government will make appropriate adjustments to the overall vacation according to the dates of different holidays, which will bring many unusual vacations and will also lead to instability of forecast results. We found that most of the cases originated from unusual holidays, and in a few other cases, the forecast error for that day was also greater than the annual average. However, there is still one day for which no reasonable explanation has yet been found. Table 4 summarizes the situation during these 15 days.

M.4 uses a two-stage attention mechanism. For different features at different moments in the encoding process. Corresponding weights can also be obtained to judge the output of the current features at the current moment. However, this two-stage attention mechanism is different from M.3. First, to better extract the weights of features and avoid the insignificant problems caused by encoding energy consumption and other features at the same time, M.4 processes features and labels with the encoder and decoder and uses different optimizers to optimize them separately. In addition, in the encoding and decoding process, all input data of LSTM must be processed by attention before input. Finally, since only the features are input in the encoder, the hidden layer state of each time step contains only the information of the feature, and the attention result of the hidden layer state of the encoder in the decoder will be the same as the result of M.3.

The attention information of the decoder in the M.4 model is shown in the following table (Table 5 and Table 6). The results show that the weights of the hidden layer states on each day of the encoder sequence are very close during decoding, mainly because the M.4 encoder does not include energy consumption data. That is, the model considers almost all the hidden layer states of all sequences in the encoding process.

The result of feature attention is a matrix of (365, 7, 11), that is, the model can obtain the weight of all features in each day of the coding sequence for each prediction. However, since the three-dimensional matrix cannot be visualized, we hope to consider the overall weight of each feature in the encoding. Therefore, we average the weights on the second dimension of the matrix so that the average weights of different features in the coding sequence are obtained for each prediction. The matrix dimension becomes (365, 11). The statistical information of each feature weight is shown in Table 6. Fig. 18 visualizes the weights throughout the year.

From the statistics of the average values in Table 6, the daily max temperature, mean temperature, min temperature, and dew point temperature are the four most important features, and their average weights all exceed 11%. The four characteristics of pressure, wind speed-related features, and holidays have the lowest average weights, all of which are less than 8%. The weights of the three features related to humidity are in the middle reaches, and their average weights are between 8% and 9%.

In the more detailed visualization results (Fig. 18), when the forecast date is summer, the maximum daily temperature will be more focused, as well as the temperature (air temperature and dew point temperature) and characteristics related to humidity. The weights have increased compared to winter. However, the weight of pressure and wind speed will be reduced compared to winter, possibly due to the use of air conditioning in summer and the reduction in the use of natural ventilation in the building. Therefore, the weight of outdoor wind speed, pressure and other characteristics will be correspondingly reduced in summer. Since the energy consumption of the building does not include the heating part in winter, the effect of winter temperature changes on the final energy consumption is not as obvious.

In M.3 and M.4, the encoder and decoder model based on LSTM can effectively encode time series and introduce time dependence. However, the calculation process of the model can only be carried out iteratively, which takes a long time. More importantly, the

Table 5
M.4 decoder attention information.

Attention weight	Seventh	Sixth	Fifth	Fourth	Third	Second	First
Mean	0.1369	0.1397	0.1411	0.1434	0.1452	0.1463	0.1470
25%	0.1252	0.1339	0.1368	0.1388	0.1398	0.1401	0.1396
50%	0.1405	0.1402	0.1411	0.1429	0.1438	0.1442	0.1449
75%	0.1480	0.1449	0.1455	0.1478	0.1504	0.1514	0.1525
Max	0.1674	0.1757	0.1685	0.1725	0.1770	0.1785	0.1888

Table 6
M.4 feature attention information.

Attention weight	Mean	25%	50%	75%	Max
Holiday	0.0686	0.0301	0.0853	0.0898	0.2763
Mean temperature	0.1128	0.0856	0.0895	0.1343	0.2427
Max temperature	0.1132	0.0847	0.0886	0.1213	0.3838
Min temperature	0.1120	0.0856	0.0890	0.1428	0.2679
Mean humidity	0.0862	0.0774	0.0870	0.0899	0.1927
Max humidity	0.0801	0.0706	0.0864	0.0901	0.2181
Min humidity	0.0921	0.0780	0.0869	0.0900	0.2918
Dew point	0.1140	0.0843	0.0879	0.1548	0.2393
pressure	0.0785	0.0161	0.0889	0.1130	0.3324
Wind speed	0.0716	0.0350	0.0857	0.0900	0.3761
Max wind speed	0.0704	0.0311	0.0858	0.0908	0.2358

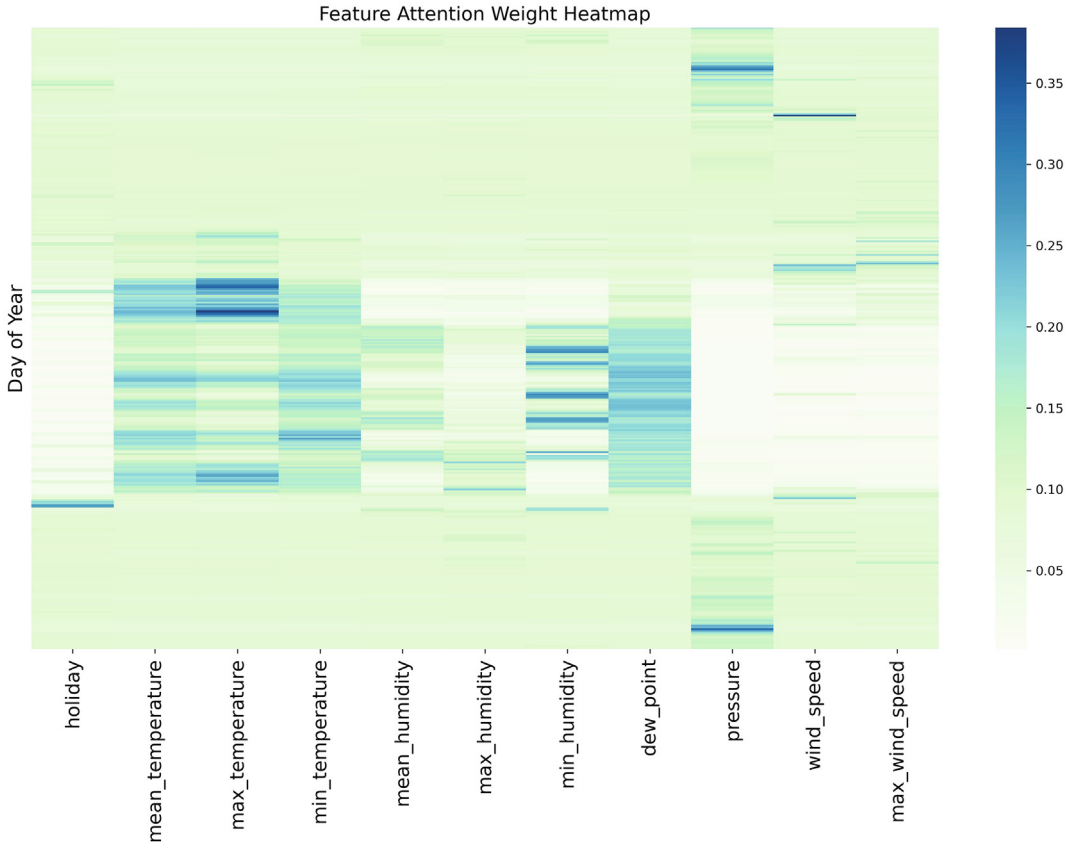

Fig. 18. Feature Attention map for M.4.

Table 7
M.5 attention information.

Attention weight	Seventh	Sixth	Fifth	Fourth	Third	Second	First
Mean	0.1427	0.1427	0.1425	0.1429	0.1430	0.1430	0.1428
25%	0.1412	0.1409	0.1406	0.1409	0.1408	0.1406	0.1407
50%	0.1431	0.1430	0.1425	0.1425	0.1424	0.1424	0.1428
75%	0.1443	0.1442	0.1439	0.1439	0.1439	0.1439	0.1441
Max	0.1780	0.1766	0.2006	0.2357	0.1995	0.2071	0.2080

Table 8
Model architecture.

Model ID	Network structures	
	Encoder	Decoder
M.1	One layer LSTM (LSTM units: 256)	One layer LSTM (LSTM units: 256)
M.2	One layer LSTM (LSTM units: 256)	One layer LSTM (LSTM units: 256)
M.3	One layer LSTM (LSTM units: 256); Two layers Attention score network (Number of neurons: 256 for each layer)	One layer LSTM (LSTM units: 256)
M.4	One layer LSTM (LSTM units: 64); One Attention score network (Number of neurons: 135)	One layer LSTM (LSTM units: 64); Two layers Attention score network (Number of neurons: 193 for first, 64 for second); One final output layer (Number of neurons: 129)
M.5	8 heads self-attention structures (query size: 8; value size: 8; Number of encoder to stack: 4; Feedforward units: 256)	8 heads self-attention structures (query size: 8; value size: 8; Number of decoder to stack: 4; Feedforward units: 256)

results of M.3 and M.4 show that the amount of information contained in the hidden layer state of each time step of LSTM is different. Generally, the later hidden layer states will contain more information, and we prefer that the hidden state of each time step represent only the information of the current time step. To eliminate this difference, M.5 uses a pure attention mechanism for encoding and decoding. The encoding and decoding of each day can be calculated in parallel.

M.5 uses the attention mechanism in both the encoder and the decoder, so corresponding weights are obtained in both the encoding and decoding processes. The self-attention mechanism is used in the encoder, that is, the state vector of each day will be calculated with the other 6-day vectors to obtain the attention weight.

This weight will readd the vectors in each time step to calculate a new state vector for the current time step. Moreover, due to the multi-head attention mechanism adopted in the coding, the coding

Table 9
Model hyperparameters.

Model ID	Hyperparameters				
	Activation function	Learning rate	optimizer	Drop out	Learning rate schedule
M.1	LeakyReLU(negative_slope = 0.01)	0.01	Adam	0.1	Every 10 steps decay 0.1
M.2	LeakyReLU(negative_slope = 0.01)	0.01	Adam	0.1	Every 10 steps decay 0.1
M.3	LeakyReLU(negative_slope = 0.01)	0.02	Adam	0.1	Every 10 steps decay 0.1
M.4	LeakyReLU(negative_slope = 0.01)	0.02	Adam	0.2	Every 10 steps decay 0.1
M.5	LeakyReLU(negative_slope = 0.01)	0.01	Adam	Feedforward layer: 0.2; Encoder and Decoder: 0.3	None

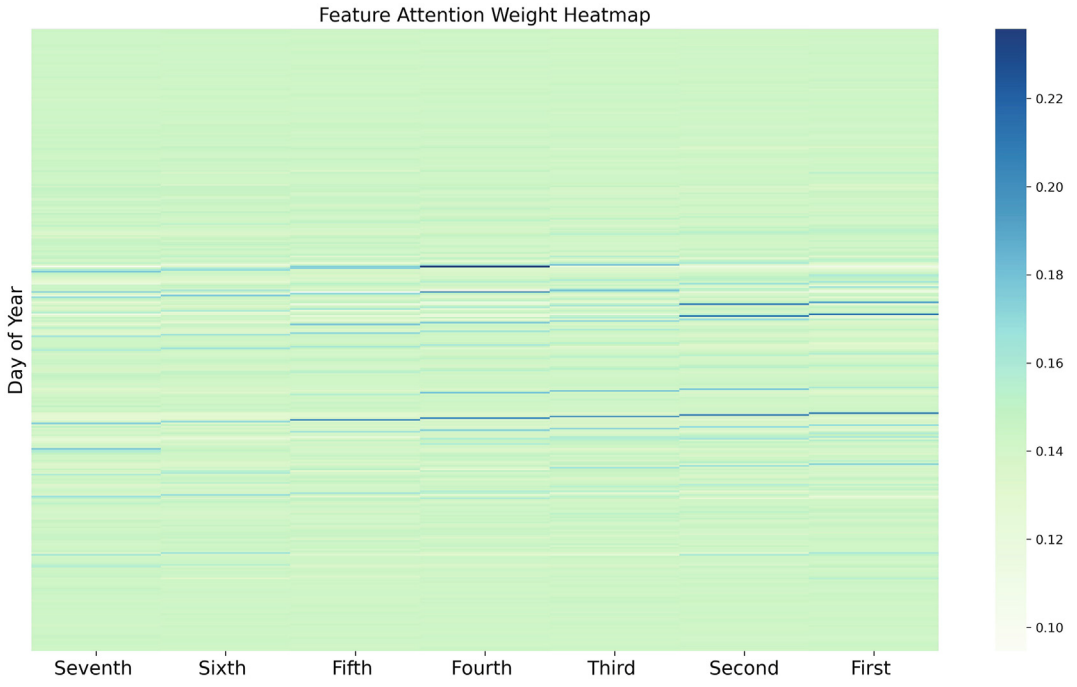

Fig. 19. Feature Attention map for M.5.

process actually realizes parallel computing of an integrated model. In this example, for the prediction step for each day, the encoder will finally output an attention score of (8, 7, 7), where 8 represents that the model uses 8 heads. Regardless of the dimension, the final output of the encoder is the state vector of each encoding time step, and the attention of the encoder will not be discussed in this example.

The decoder uses an attention mechanism similar to M.3, that is, the input of the decoder and the state vector of each time step in the encoder are calculated, and the result of the attention calculation is used to weight the state vector of the encoding sequence. We obtain the final attention vector for prediction. The attention weight of each prediction result in the decoder is (8, 1, 7). For the sake of simplification, the following multi-head attention results are averaged and analyzed. The statistics of attention weights are shown in Table 7.

Table 7 shows that the average values of the weights of the hidden layer states on each day are very close, and the change from the 25% quantile to the 75% quantile is not very large, but there are some values in the maximum part gap, showing that when using attention for encoding, the model is almost equally likely to consider the state vector of each day when decoding and predicting. The visualization results also prove this conclusion. Fig. 19 shows the visualization results of the attention weights throughout the year. Except for a few dates, most of the weights are evenly distributed in the 7-day sequence.

The results of M.5 show that in this case, when the encoded hidden layer state does not have a large information gap, the prediction will equally consider the information of each day in the encoding sequence. In addition, since M.5 contains a stack of several layers of attention mechanisms, the above attention is taken from the result of the last output before prediction.

5. Conclusions

This paper proposed three encoder and decoder models that use the attention mechanism: (1) encoder and decoder model with attention (using LSTM for encoding and decoding), (2) encoder and decoder model with two-stage attention (using LSTM for encoding and decoding), and (3) transformer. The significance of the proposed model lies in two points: (1) the structure of the encoder and decoder could get good prediction results without future weather data, and (2) the attention result improves the interpretability of the deep-learning prediction model.

To demonstrate the effectiveness of the models, we have presented a case study on a government office building in China. Among the models, the difference between the best baseline model that still inputs real weather data and the M.2 model that uses only

holiday information is small, and the gap between MAPE and CV-RMSE is less than 1 percentage point. That is, the time-dependent encoder and decoder can still achieve better prediction accuracy without inputting the weather data that need to be predicted in real scenario.

Finally, this paper uses three kinds of attention mechanisms to further improve the interpretability of the model. When the model uses LSTM for encoding and decoding, the prediction result of decoding will be more biased towards the hidden layer state of the last day of the encoding sequence. Due to the computational characteristics of LSTM, it often contains more information. The attention weight will fluctuate when encountering special holidays. When the model uses only the attention mechanism for encoding and decoding, since there is no difference in the amount of information in the hidden layer state at different time steps, in this case, the same weight is assigned to each day in the encoder sequence during decoding for the final forecast.

The attention weight of the feature shows that the model has different emphases on different features in the air-conditioned season and the non-air-conditioned season. In the air-conditioned season, more attention will be given to characteristics such as temperature, and in the non-air-conditioned season, more attention will be given to characteristics such as pressure or wind speed. The visualization results of features can be used to prioritize and focus important features when there are insufficient features in the future to improve the prediction effect.

Our future research will concentrate on more situations of energy consumption prediction based on this article. For example, energy consumption prediction is performed on hourly data or finer time granularity, and a more optimized model is used to classify and process the features in detail.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been supported by National Natural Science Foundation of China (No.51978482).

References

[1] N. Somu, K. Ramamritham,, A hybrid model for building energy consumption forecasting using long short term memory networks, Appl. Energy 261 (2020), https://doi.org/10.1016/j.apenergy.2019.114131 114131.
[2] Z. Wang, Y. Wang, R. Zeng, R.S. Srinivasan, S. Ahrentzen, Random Forest based hourly building energy prediction, Energy Build. 171 (2018) 11–25, https://doi.org/10.1016/j.enbuild.2018.04.008.
[3] Y. Zhang, C.Q. He, B.J. Tang, Y.M. Wei, China's energy consumption in the building sector: A life cycle approach, Energy Build. 94 (may) (2015) 240–251.
[4] C. Spandagos, T.L. Ng, Equivalent full-load hours for assessing climate change impact on building cooling and heating energy consumption in large Asian cities, Appl. Energy 189 (2017) 352–368, https://doi.org/10.1016/j.apenergy.2016.12.039.
[5] F. Moazeni, J. Khazaei, Dynamic economic dispatch of islanded water-energy microgrids with smart building thermal energy management system, Appl. Energy 276 (2020), https://doi.org/10.1016/j.apenergy.2020.115422 115422.
[6] M. Yousefi, A. Hajizadeh, M.N. Soltani, B. Hredzak, N. Kianpoor, Profit assessment of home energy management system for buildings with A-G energy labels, Appl. Energy 277 (2020), https://doi.org/10.1016/j.apenergy.2020.115618 115618.
[7] K. Amasyali, N.M. El-Gohary, A review of data-driven building energy consumption prediction studies, Renew. Sustain. Energy Rev. 81 (2018) 1192–1205, https://doi.org/10.1016/j.rser.2017.04.095.
[8] M.A. Jallal, A. González-Vidal, A.F. Skarmeta, S. Chabaa, A. Zeroual, A hybrid neuro-fuzzy inference system-based algorithm for time series forecasting applied to energy consumption prediction, Appl. Energy 268 (2020), https://doi.org/10.1016/j.apenergy.2020.114977 114977.
[9] N. Fumo, M.A.R. Biswas, Regression analysis for prediction of residential energy consumption, Renew. Sustain. Energy Rev. 47 (2015) 332–343, https://doi.org/10.1016/j.rser.2015.03.035.
[10] L.G.B. Ruiz, M.P. Cuéllar, M.D. Calvo-Flores, M.D.C.P. Jiménez, An Application of Non-Linear Autoregressive Neural Networks to Predict Energy Consumption in Public Buildings, Energies 9 (9) (2016) 684.
[11] S. Paudel, M. Elmitri, S. Couturier, P.H. Nguyen, R. Kamphuis, B. Lacarrière, O. Le Corre, A relevant data selection method for energy consumption prediction of low energy building based on support vector machine, Energy Build. 138 (2017) 240–256, https://doi.org/10.1016/j.enbuild.2016.11.009.
[12] Y. Sun, S. Wang, F. Xiao, Development and validation of a simplified online cooling load prediction strategy for a super high-rise building in Hong Kong, Energy Convers. Manage. 68 (Apr. 2013) 20–27, https://doi.org/10.1016/j.enconman.2013.01.002.
[13] Z. Wang, Y. Wang, R.S. Srinivasan, A novel ensemble learning approach to support building energy use prediction, Energy Build. 159 (2018) 109–122.
[14] [et al.,, Supervised based machine learning models for short, medium and long-term energy prediction in distinct building environment, Energy (2018).
[15] C. Fan, Y. Sun, Y. Zhao, M. Song, J. Wang, Deep learning-based feature engineering methods for improved building energy prediction, Appl. Energy 240 (2019) 35–45.
[16] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription," 2011.
[17] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-rahman. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process Mag. 29 (6) (2012) 82–97.
[18] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (May 2017) 84–90, https://doi.org/10.1145/3065386.
[19] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural Language Processing (almost) from Scratch, J. Machine Learn. Res. 12 (1) (2011) 2493–2537.
[20] K. Muralitharan, R. Sakthivel, R. Vishnuvarthan, Neural network based optimization approach for energy demand prediction in smart grid, Neurocomputing 273 (2018) 199–208.
[21] Y. Ding, Q. Zhang, T. Yuan, Research on short-term and ultra-short-term cooling load prediction models for office buildings, Energy Build. 154 (2017) 254–267.
[22] Z. Yuan, W. Wang, H. Wang, S. Mizzi, Combination of cuckoo search and wavelet neural network for midterm building energy forecast, Energy 202 (2020), https://doi.org/10.1016/j.energy.2020.117728 117728.
[23] G. Chitalia, M. Pipattanasomporn, V. Garg, S. Rahman, Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks, Appl. Energy 278 (2020), https://doi.org/10.1016/j.apenergy.2020.115410 115410.
[24] C. Fan et al., Statistical investigations of transfer learning-based methodology for short-term building energy predictions, Appl. Energy 262 (2020), https://doi.org/10.1016/j.apenergy.2020.114499 114499.
[25] H. Chitsaz, H. Shaker, H. Zareipour, D. Wood, N. Amjady, Short-term electricity load forecasting of buildings in microgrids, Energy Build. 99 (2015) 50–60.
[26] A. Rahman, V. Srikumar, A.D. Smith, Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks, Appl. Energy 212 (2018) 372–385.
[27] B. Gao, X. Huang, J. Shi, Y. Tai, J. Zhang, Hourly forecasting of solar irradiance based on CEEMDAN and multi-strategy CNN-LSTM neural networks, Renewable Energy 162 (2020) 1665–1683, https://doi.org/10.1016/j.renene.2020.09.141.
[28] L. Wen, K. Zhou, S. Yang, Load demand forecasting of residential buildings using a deep learning model, Electr. Power Syst. Res. 179 (2020), https://doi.org/10.1016/j.epsr.2019.106073 106073.
[29] X. Yan, "A Spatial-Temporal Interpretable Deep Learning Model for improving interpretability and predictive accuracy of satellite-based PM2.5," Environmental Pollution, p. 15, 2021.
[30] D. Zhang, C. Yin, K.M. Hunold, X. Jiang, J.M. Caterino, P. Zhang, An interpretable deep-learning model for early prediction of sepsis in the emergency department, Patterns 2 (2) (2021), https://doi.org/10.1016/j.patter.2020.100196 100196.
[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," arXiv:1409.3215 [cs], Dec. 2014, Accessed: Nov. 07, 2020. [Online]. Available: http://arxiv.org/abs/1409.3215.
[32] E. Skomski, J.-Y. Lee, W. Kim, V. Chandan, S. Katipamula, B. Hutchinson, Sequence-to-sequence neural networks for short-term electrical load forecasting in commercial office buildings, Energy Build. 226 (2020), https://doi.org/10.1016/j.enbuild.2020.110350 110350.
[33] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.
[34] Y. Ding, Y. Zhu, J. Feng, P. Zhang, Z. Cheng, Interpretable spatio-temporal attention LSTM model for flood forecasting, Neurocomputing 403 (2020) 348–359, https://doi.org/10.1016/j.neucom.2020.04.110.
[35] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data," arXiv:1611.06067 [cs], Nov. 2016, Accessed: Dec. 06, 2020. [Online]. Available: http://arxiv.org/abs/1611.06067.
[36] J. Kim, J. Moon, E. Hwang, P. Kang, Recurrent inception convolution neural network for multi short-term load forecasting, Energy Build. 194 (2019) 328–341, https://doi.org/10.1016/j.enbuild.2019.04.034.

[37] A. Vaswani et al., "Attention is All you Need," p. 11.
[38] T. Kodama, A. Okabe, and K. Kogiso, "Simultaneous Estimation of Contraction Ratio and Parameter of McKibben Pneumatic Artificial Muscle Model Using Log-Normalized Unscented Kalman Filter," 2016.
[39] C. Behm, L. Nolting, A. Praktiknjo, How to model European electricity load profiles using artificial neural networks, Appl. Energy 277 (2020), https://doi.org/10.1016/j.apenergy.2020.115564 115564.
[40] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[41] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," p. 9.
[42] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," arXiv:1409.0473 [cs, stat], May 2016, Accessed: Nov. 10, 2020. [Online]. Available: http://arxiv.org/abs/1409.0473.
[43] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical Attention Networks for Document Classification," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, 2016, pp. 1480–1489. 10.18653/v1/N16-1174.
[44] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction," arXiv:1704.02971 [cs, stat], Aug. 2017, Accessed: Nov. 17, 2020. [Online]. Available: http://arxiv.org/abs/1704.02971.
[45] Y. Gao, Y. Ruan, C. Fang, S. Yin, Deep learning and transfer learning models of energy consumption forecasting for a building with poor information data, Energy Build. 223 (2020), https://doi.org/10.1016/j.enbuild.2020.110156 110156.
[46] M. Ribeiro, K. Grolinger, H.F. ElYamany, W.A. Higashino, M.A.M. Capretz, Transfer learning with seasonal and trend adjustment for cross-building energy forecasting, Energy Build. 165 (2018) 352–363, https://doi.org/10.1016/j.enbuild.2018.01.034.
[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Machine Learn. Res. 15 (1) (2014) 1929–1958.
[48] N. Benny Toomarian, J. Barhen, Learning a trajectory using adjoint functions and teacher forcing, Neural Networks 5 (3) (1992) 473–484.