Title

A Unifying Framework of Attention-Based Neural Load Forecasting

Authors

Xiong, Jing
Zhang, Yu

Publication Date

2023

DOI

10.1109/access.2023.3275095

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 April 2023, accepted 7 May 2023, date of publication 10 May 2023, date of current version 1 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3275095

RESEARCH ARTICLE

A Unifying Framework of Attention-Based Neural Load Forecasting

JING XIONG AND YU ZHANG, (Member, IEEE)
Department of Electrical and Computer Engineering, University of California, Santa Cruz, CA 95064, USA

Corresponding author: Yu Zhang (yzhan419@ucsc.edu)

This work was supported in part by the Hellman Fellowship, and in part by the Seed Fund Award from Center for Information Technology Research in the Interest of Society (CITRIS) and the Banatao Institute, University of California.

ABSTRACT Accurate load forecasting is critical for reliable and efficient planning and operation of electric power grids. In this paper, we propose a unifying deep learning framework for load forecasting, which includes time-varying feature weighting, hierarchical temporal attention, and feature-reinforced error correction. Our framework adopts a modular design with good generalization capability. First, the feature-weighting mechanism assigns input features with temporal weights. Second, a recurrent encoder-decoder structure with hierarchical attention is developed as a load predictor. The hierarchical attention enables a similar day selection, which re-evaluates the importance of historical information at each time step. Third, we develop an error correction module that explores the errors and learned feature hidden information to further improve the model's forecasting performance. Experimental results demonstrate that our proposed framework outperforms existing methods on two public datasets and performance metrics, with the feature weighting mechanism and error correction module being critical to achieving superior performance. Our framework provides an effective solution to the electric load forecasting problem, which can be further adapted to many other forecasting tasks.

INDEX TERMS Short-term load forecasting, feature weighting, attention mechanism, error correction.

I. INTRODUCTION

Load forecasting refers to the prediction of the future load behavior, which can be derived from the historical load pattern and its relevant features. Based on different forecast horizons, load forecasting can be divided into three categories short-term (one hour to a week), medium-term (one week to a year), and long-term (one to twenty years) forecasting. Each of them benefits various applications and business needs. Long-term forecasting is mainly used in power system planning such as generation and transmission expansion planning. Medium-term forecasting plays a crucial role in maintenance scheduling. Short-term load forecasting (STLF) is indispensable for day-ahead unit commitment, market clearing, spinning reserve plans, energy bidding, as well as economic load dispatch [1].

As the forecasting time span shrinks, the requirement for forecasting accuracy increases. In addition, wide applications of renewable energy generations, energy storage systems, and electric vehicles in recent years have had a huge impact on users' load behavior. This poses significant challenges in forecasting load demand [2]. Various approaches have been proposed to improve the STLF accuracy. They can be roughly categorized into three classes: (i) time series analysis, (ii) classical machine learning algorithms, and (iii) deep learning models.

Time series analysis has been widely used in many applications. Various versions of autoregressive integrated moving average (ARIMA) models were used for STLF [3]. These methods are easily implemented and interpreted. However, they often require meticulous preprocessing to make a time series stationary [4]. Moreover, time series approaches are sensitive to irrelevant features and may fail to capture a long-term dependency.

With the vigorous development of classical machine learning theory, researchers began to explore its application in STLF. Ceperic et al. proposed a support vector regression machines (SVR) approach for STLF, which minimizes the

The associate editor coordinating the review of this manuscript and approving it for publication was Abdullah Iliyasu.

user interaction requirement by an adaptive model building strategy [5]. A random forest (RF) model was used to deal with non-stationarity, heteroscedasticity, trend, and multiple seasonal cycles in load data [6]. In order to avoid information loss, Cheng et al. used different feature sets to construct an ensemble random forest-based model [7]. Taieb et al. implemented component-wise gradient boosting models (GBM) for each hour for multi-step STLF [8]. These classic machine learning models, which are more robust to tolerate irrelevant features, are capable of capturing nonlinear behaviors of electricity load. However, most of them use predetermined nonlinear models, which may prevent them from effectively learning the true underlying mappings [9].

In the last decade, deep neural networks have demonstrated remarkable capabilities in uncovering complex input-output relationships in various fields, such as natural language processing and computer vision. The Deep Belief Network (DBN) is a prevalent model for time-series forecasting tasks. To improve the forecasting accuracy, rough set theory was introduced in [10] to enhance the feature extraction capability of the restricted Boltzmann machine (RBM) within the DBN. Furthermore, interval probability distribution learning (IPDL) in [11] uses deep generative neural networks to learn the input data distribution and provide uncertainty intervals for wind speed forecasting. Recurrent neural networks (RNNs) are commonly used with proven efficacy for sequence-to-sequence (seq2seq) learning and time-series forecasting tasks. However, the vanilla RNNs suffer from the gradient vanishing issue that limits their performance. To address this issue, Hochreiter and Schmidhuber proposed long short-term memory networks (LSTM), which use a gating technique to control information flows. LSTM uses three gates (input/forget/output) to retain relevant information for long-term memory while discarding the other information [12]. In 2014, Cho et al. proposed gated recurrent units (GRUs) which is another gating mechanism-based RNN. GRUs reduce the number of gates with fewer parameters to train; see details in [13].

Seq2seq learning solves the mapping between the sequential inputs and outputs of the task, which shares various similarities with time-series learning problems [14]. The encoder-decoder structure usually serves as the backbone for most seq2seq models [15], [16]. Specifically, in time-series tasks, the encoder encodes the historical input feature sequence into a single fixed-length vector based on which the decoder yields the output. However, coping with long input sequences can be challenging. To mitigate this drawback, the attention mechanism was introduced to search for a set of positions in historical time steps where the most relevant information can be concentrated [17]. For this new paradigm, a context vector is designed to bridge the encoder and the decoder, which is filtered for each output time step.

It is often challenging to deal with various conditions in real data by using single-module approaches. To further improve the prediction accuracy, hybrid models combining the advantages of all added modules have been developed. When it comes to the STLF, feature engineering [18] and error correction modules play an important role. Traditional feature selection approaches, such as filters, wrappers and embedding methods, aim at selecting the smallest subset of features that contribute the most to the output. A two-stage hybrid model for STLF is proposed in [19]. Based on the mutual information criterion, the selected features are fed into a forecast engine that is implemented via Ridgelet and Elman neural networks.

Rather than selecting a subset of features, feature weighting attempts to weight each feature based on their importance or relevance with the output [20]. Utilizing the feature weights given by the random forest, Xuan proposed a multi-model fusion based deep neural network to forecast the load demand [21]. Qin et al. proposed an input attention layer as feature weighting that can be trained simultaneously with the model [9]. However, their scheme is based only on the past information which cannot capture all the information of the entire input sequence. Moreover, the feature weighting part is embedded in the encoder, which makes it hard to be adopted for other basic structures; e.g. the convolutional neural network (CNN).

The prediction error generally comes from two parts: the learning capability of the original model and the newly emerging unknown data. To further improve the prediction accuracy, error correction module can learn useful hidden information from the error values. In this context, Deng et al. proposed a hybrid model which includes a decomposition module, a forecasting module, and an error correction module for wind speed forecasting [22]. Leveraging the dynamic mode decomposition (DMD) method in fluid dynamics, Kong et al. captured the spatio-temporal dynamics of error series in STLF [2]. This algorithm first constructs the error Hankel matrix and then does the pattern decomposition of the error. Existing approaches for the error correction are based on either a completely new model such as ARIMA [22], [23] or extreme learning machine [24]. However, the design of a new model will increase the learning cost. Model selection and hyperparameter tuning are necessary, which may greatly affect the sampling complexity and training time. In addition, the useful knowledge learned by the predictive model will be lost in the new model.

Transfer learning was proposed to deal with the aforementioned issues. The motivation is to use previously acquired domain knowledge to solve new problems faster, or yield better solutions [25]. In recent years, transfer learning has been successfully used for supervised and unsupervised learning. In load forecasting, researchers have also explored this technique, where the knowledge is transferred from one region/household to another one [26]. In this case, the source and target domains are the same, which is load and relative features while the task is also the same. The key challenge for applying transfer learning in error correction is incorporating the error information into the target domain without changing

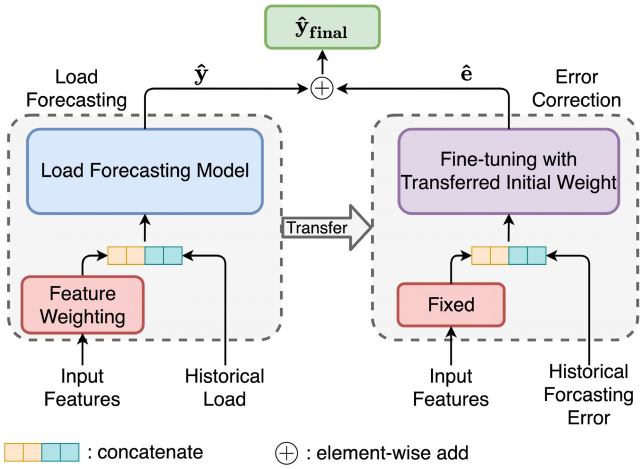


FIGURE 1. The architecture of the proposed framework. The left side is the load forecasting module with an input feature-weighting mechanism designed to weigh the different input features. The right side is the error correction module transferred from the left side model to further enhance forecasting ability. The detailed structures of feature-weighting mechanism and load forecasting module are shown in Fig. 2 and Fig. 3.

the input dimension while capitalizing on previously acquired feature knowledge.

Considering the limitations of the aforementioned prior works, in this paper we propose a novel deep learning framework that incorporates a dynamic feature weighting mechanism and a transferred learning based error correction module. The main contributions of our work are listed as follows:

1) The proposed framework offers a modular and plug-and-play functionality that can be adapted to different types of data and setups in STLF.
2) Compared with classic feature weighting methods, our attention-based time-varying feature weighting mechanism can assign different levels of importance to each feature at each time step, which allows for more dynamic and adaptive feature selection.
3) The hierarchical temporal attention layer captures the similar day and similar hour information, which is a critical factor for accurate STLF.
4) The proposed error correction module leverages transfer learning. This eliminates the need for a new model design and inherits the learned feature knowledge.
5) Extensive experimental results corroborate the merits of our approach, which outperforms existing methods based on various performance metrics.

The remainder of the paper is organized as follows. Section II presents the overall framework with all proposed modules. The simulation setup and results are reported in Section III and IV, respectively. Finally, Section V summarizes this work.

II. THE PROPOSED LOAD FORECASTING FRAMEWORK

In the following section, we introduce the overall load forecasting framework structure as shown in Fig. 1. The framework mainly consists of three modules: (i) the feature-weighting mechanism, (ii) the short-term load forecasting module, and (iii) the error correction module.

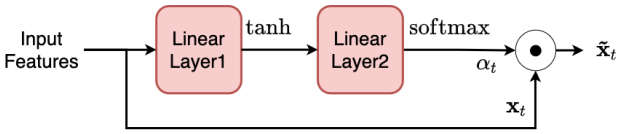


FIGURE 2. The feature-weighting mechanism structure with two linear layers.

A. FEATURE EMBEDDING AND FEATURE-WEIGHTING MECHANISM

Input features can generally be divided into two categories: numeric features and categorical features. As the model requires numeric input, a categorical feature would be transformed into a numeric vector. For STLF, the inputs can contain meteorological conditions (e.g., temperature, humidity, wind speed and direction, etc), time-related features (e.g., indicators of holidays, seasons, etc), and utility discount programs. For those categorical features, we use one-hot encoding in this work. After the embedding, all input features $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n) \in \mathbb{R}^{(T_h+T_f)\times n}$ are the concatenation of encoded categorical features and continuous numeric features. Each row of $\mathbf{X}$, denoted as $\mathbf{x}_t = (x_t^1, x_t^2, \ldots, x_t^n)$, represents all features at time t.

Feature selection plays a crucial role in machine learning methods [18]. Irrelevant features can significantly affect the model's performance. Instead of making a hard feature selection which is a special case of feature weighting mechanism, the proposed model is able to adaptively weigh different features and give more attention to features that contribute more to the target values. In [27], the feature selection layer is entangled in the encoder. Therefore, it is hard to transfer to other load forecasting modules. In order to modularize the proposed framework, we separate the feature selection layer from the encoder, and the weight α_t^k of each feature k at time t is calculated via the softmax operator:

$$\alpha_t^k = \frac{\exp\left(h_t^k\right)}{\sum_{i=1}^{n} \exp\left(h_t^i\right)}, \quad k = 1, 2, \ldots, n, \tag{1}$$

where h_t^k is the k-th entry of the vector $\mathbf{h}_t = \mathbf{V}_\alpha \tanh\left(\mathbf{W}_\alpha \mathbf{x}_t^\top\right) \in \mathbb{R}^n$.

The weight matrices $\mathbf{V}_\alpha \in \mathbb{R}^{n\times d_h^{fw}}$ and $\mathbf{W}_\alpha \in \mathbb{R}^{d_h^{fw}\times n}$ are trained jointly with the proposed model. d_h^{fw} is the number of neurons in the hidden layer and it's a hyper-parameter to tune. We omit the bias term for succinctness. Then, the weighted feature input is given by $\tilde{\mathbf{x}}_t = \alpha_t \odot \mathbf{x}_t$, where $\odot$ denotes the element-wise multiplication. The detailed structure for the feature weighting mechanism is shown in Fig. 2.

The encoder's inputs are the concatenation of embedded categorical features, continuous numeric features, and historical target values (active power demand) at each time step $t \in [t - T_h + 1, t]$. The decoder's inputs are the concatenation

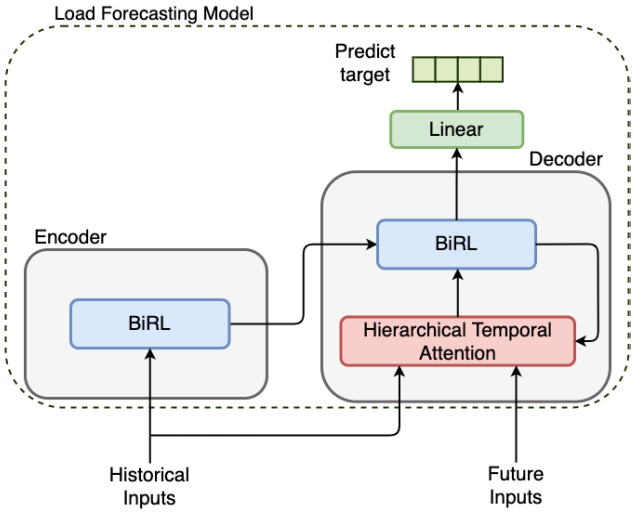


FIGURE 3. The network architecture of the load forecasting model. The hierarchical temporal attention in the decoder focuses on the temporal similarity to incorporate similar day information.

of embedded categorical features and continuous numeric features at each time step $t \in [t+1, t+T_f]$. T_h and T_f are the window size of historical and future data, respectively.

B. SHORT-TERM LOAD FORECASTING MODEL

1) ENCODER-DECODER STRUCTURE

The encoder-decoder structure is a workhorse in state-of-the-art deep neural networks. For time series forecasting, the encoder maps historical input features $\mathbf{x}_t$ and output $y_{h,t}$ at each time step to a hidden vector $\mathbf{h}_t$ that is passed to the decoder. Then, the decoder uses the last step hidden state of the encoder as its initial hidden state, and outputs future target values based on future feature inputs. In this work, a bi-directional recurrent layer (RL) is used for both encoder and decoder. We abbreviate the formulation for bi-directional RL as $\mathbf{h}_t = \text{BiRL}(\cdot)$, where RL can be chosen as recurrent neural network (RNN), LSTM or gated recurrent unit (GRU).

The BiRL consists of two sub-layers in opposite directions which can capture the complete information of the entire input sequence at each time step. Let $\mathbf{h}_t^f, \mathbf{h}_t^b \in \mathbb{R}^{hs}$ denote the hidden state of forward and backward recurrent layer at time t, respectively. Given a sequence of historical weighted feature and target value pairs $(\tilde{\mathbf{x}}_t, y_t)$, the encoder's hidden states are updated from time $t - T_h + 1$ to t as

$$\mathbf{h}_t^f = \text{RL}^f(\mathbf{h}_{t-1}^f, [\tilde{\mathbf{x}}_t; y_t]^\top), \tag{2a}$$
$$\mathbf{h}_t^b = \text{RL}^b(\mathbf{h}_{t+1}^b, [\tilde{\mathbf{x}}_t; y_t]^\top), \tag{2b}$$
$$\mathbf{h}_t^e = \left[\mathbf{h}_t^{f\top}; \mathbf{h}_t^{b\top}\right]^\top, \tag{2c}$$

where $[\mathbf{a}; \mathbf{b}]$ denotes the concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$.

Using future weighted feature $\tilde{\mathbf{x}}_t$ and context vector of hierarchical temporal attention $\mathbf{a}_t$ (see next subsection) as inputs, the decoder updates the hidden state iteratively from time $t+1$ to $t+T_f$ with initial state $\mathbf{h}_t^e$. Hence, we have $\mathbf{h}_t^d = \text{BiRL}(\mathbf{h}_t^e, [\tilde{\mathbf{x}}_t; \mathbf{a}_t^\top])$ with the detailed steps as

$$\mathbf{h}_t^f = \text{RL}^f(\mathbf{h}_{t-1}^f, [\tilde{\mathbf{x}}_t; \mathbf{a}_t^\top]), \tag{3a}$$
$$\mathbf{h}_t^b = \text{RL}^b(\mathbf{h}_{t+1}^b, [\tilde{\mathbf{x}}_t; \mathbf{a}_t^\top]), \tag{3b}$$
$$\mathbf{h}_t^d = \left[\mathbf{h}_t^{f\top}; \mathbf{h}_t^{b\top}\right]^\top. \tag{3c}$$

Finally, a fully connected layer with the rectified linear unit (ReLU) activation function is used to transform the hidden information to the forecast output from time $t+1$ to $t+T_f$:

$$\mathbf{y}_{f,t} = \mathbf{V}_y \text{ReLU}(\mathbf{W}_y \mathbf{h}_t^d), \tag{4}$$

where $\mathbf{V}_y \in \mathbb{R}^{1 \times d_h^o}$ and $\mathbf{W}_y \in \mathbb{R}^{d_h^o \times 2hs}$ are weight matrices.

2) HIERARCHICAL TEMPORAL ATTENTION MECHANISM

Incorporating the information of similar days and hours has been considered in the literature for load forecasting; see e.g., [28] and [29]. However, such information is often treated as additional input features or used to generate separate models. This paper uses a novel hierarchical temporal attention layer designed from our previous work [27], which incorporates a similar day soft selection to re-evaluate the importance of historical information at each time step t.

Consider using previous M days of historical data to forecast the hourly loads for the next day, where each day includes $t_d = 24$ data points. Thus, we have $T_h = M \times t_d$ and $T_f = t_d$. Let $\mathbf{X}_i = (\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^n) \in \mathbb{R}^{t_d \times n}$ and $\mathbf{X}_f = (\mathbf{x}_f^1, \mathbf{x}_f^2, \ldots, \mathbf{x}_f^n) \in \mathbb{R}^{t_d \times n}$ collect the historical features for the day i and the future features of the next day. We use the sum of feature-by-feature dissimilarities $D(\mathbf{X}_i, \mathbf{X}_f) = \sum_{k=1}^n \|\mathbf{x}_i^k - \mathbf{x}_f^k\|_2$ to quantify the distance between all features of those two days. Then, the similar day weight γ_i is calculated as the softmax of the reciprocal of the distance:

$$\gamma_i = \frac{\exp\left(D^{-1}(\mathbf{X}_i, \mathbf{X}_f)\right)}{\sum_{i=1}^M \exp\left(D^{-1}(\mathbf{X}_i, \mathbf{X}_f)\right)}, \quad i = 1, 2, \ldots, M. \tag{5}$$

When forecasting load at time t, not all historical data contribute equally to the model's output. Hence, the attention mechanism facilitates the extraction of historical information that is more important to the current forecast value. Let subscript i denote the i-th day and j for j-th hour. Then, the attention weight $\beta_{i,j,t}$ is given by

$$\beta_{i,j,t} = \frac{\exp(d_{i,j,t})}{\sum_{i=1}^M \sum_{j=1}^{t_d} \exp(d_{i,j,t})}, \tag{6}$$

where $d_{i,j,t}$ is the $(i \times t_d + j)$-th element of vector $\mathbf{d}_t = (d_t^1, d_t^2, \ldots, d_t^{T_h})^\top \in \mathbb{R}^{T_h}$, which is given as

$$\mathbf{d}_t = \mathbf{V}_d \tanh\left(\mathbf{W}_d \left[\mathbf{h}_{t-1}^{d\top}; \mathbf{x}_t\right]^\top\right). \tag{7}$$

The two weight matrices $\mathbf{V}_d \in \mathbb{R}^{T_h \times d_h^{att}}$ and $\mathbf{W}_d \in \mathbb{R}^{d_h^{att} \times (2hs+n)}$ are trained jointly with the proposed model. $\mathbf{h}_{t-1}^d$ is the hidden vector of the decoder BiLSTM at time $t-1$.

To this end, let $\mathbf{h}_{i,j}$ denote the historical hidden state for the j-th hour in the i-th day from the encoder. The context

Algorithm 1 Training Procedure of the Framework

Input : Forecasting module training set $\mathcal{D}_l = \{\mathbf{X}, \mathbf{y}_h, \mathbf{y}_f\}$, Error correction module training set $\mathcal{D}_e = \{\mathbf{X}_e, \mathbf{y}_{e,h}, \mathbf{y}_{e,f}\}$, Number of epochs N_l and N_e, Feature weighting module f_l, Load forecasting module g_l

Output: Trained model $\{f_l, g_l, g_e\}$

Initialize model parameters;
for *epoch* $=$ *1 to* N_l **do**
  for *batch of* $\{\mathbf{X}, \mathbf{y}_h, \mathbf{y}_f\} \in \mathcal{D}_l$ **do**
    $\tilde{\mathbf{X}} \leftarrow f_l(\mathbf{X})$;
    $\hat{\mathbf{y}}_f \leftarrow g_l(\tilde{\mathbf{X}}, \mathbf{y}_h)$;
    Compute training loss L_{LF};
    Compute the gradient of loss;
    Update parameters in $g_l(\cdot)$ and $f_l(\cdot)$;
  end
end
$\hat{\mathbf{y}}_{e,f} \leftarrow g_l(f_l(\mathbf{X}_e), \mathbf{y}_{e,h})$;
$\mathbf{e} \leftarrow \mathbf{y}_{e,f} - \hat{\mathbf{y}}_{e,f}$;
Set feature weighting module $f_e(\cdot) = f_l(\cdot)$;
Set error correction module $g_e(\cdot) = g_l(\cdot)$;
Fixing the feature weighting layer, and train all other layers in the error correction module as follows:
for *epoch* $=$ *1 to* N_e **do**
  for *batch of* $\{\mathbf{X}_e, \mathbf{e}_h, \mathbf{e}_f\}$ **do**
    $\tilde{\mathbf{X}} \leftarrow f_l(\mathbf{X})$;
    $\hat{\mathbf{e}}_f \leftarrow g_e(\tilde{\mathbf{X}}, \mathbf{e}_h)$;
    Compute training loss L_{EC};
    Compute the gradient of loss;
    Update model parameter of $g_e(\cdot)$;
  end
end
Return trained model $\{f_l, g_l, g_e\}$.

vector of hierarchical temporal attention is calculated as $\mathbf{a}_t = \sum_{i=1}^{M} \sum_{j=1}^{t_d} \gamma_i \beta_{i,j,t} \mathbf{h}_{i,j}$.

C. ERROR CORRECTION MODULE

Traditional error correction systems often involve creating a new model to forecast errors, resulting in higher learning costs and the potential loss of learnt knowledge obtained by the original predictive model. To overcome these shortcomings, a transfer-learning-based error correction module is proposed. Transfer learning utilizes previously acquired domain knowledge to solve new problems more efficiently yielding better results. In recent years, transfer learning has succeeded in supervised and unsupervised learning, including load forecasting, where knowledge is transferred between regions or households. However, transferring knowledge in error correction requires incorporating the error information into the target domain without changing the input dimension while leveraging previously learned feature knowledge. The proposed error correction module addresses this challenge and aims to improve prediction accuracy by extracting valuable information from error values with the help of learned hidden features.

The error correction module is trained after the load forecasting module, which is first trained on dataset $\mathcal{D}_l$. Then, based on the error correction dataset $\mathcal{D}_e$, we compute the forecasting error as $\mathbf{e} = \mathbf{y}_{e,f} - \hat{\mathbf{y}}_{e,f}$, where $\mathbf{y}_{e,f}$ is the real output value and $\hat{\mathbf{y}}_{e,f}$ is the predicted value obtained by the forecasting model. The feature weighting module and error correction module are initialized by the forecasting model, with the feature weighting layer fixed and the other layers to be trained. To train the error correction module, a new dataset with feature input and forecasting error is generated and randomly split into training and validation sets. The algorithm computes the training loss and its gradient to update the error correction module via backpropagation. Upon completing the training of the error correction module, it can be used to correct forecast errors and improve forecasting accuracy. The final output is obtained as $\bar{\mathbf{y}} = \hat{\mathbf{y}} + \hat{\mathbf{e}}$, as shown in Fig. 1. The overall training procedure of the framework is summarized in Algorithm 1. The proposed transfer learning based model has several advantages including no need for hyper-parameter tuning and the ability to train with limited data. It also reuses existing knowledge learned by the original model, which results in a faster learning rate.

D. LOSS FUNCTION

For the load forecasting module, we choose the mean squared error (MSE) loss and introduce ℓ_1 regularizer to encourage sparsity. The formulation is given as:

$$L_{\mathrm{LF}} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \left\| \boldsymbol{\alpha} \right\|_1, \tag{8}$$

where λ is the weighting parameter balancing the data fitting loss and the sparsity-promoting ℓ_1 penalty.

For the error correction module, we drop the ℓ_1 regularization term because the feature weighting layer is fixed. Hence, the loss function for training this module becomes

$$L_{\mathrm{EC}} = \frac{1}{N} \sum_{i=1}^{N} (e_i - \hat{e}_i)^2. \tag{9}$$

III. EXPERIMENT SETUP

A. DATA DESCRIPTION

The proposed framework is evaluated using two public datasets: the ISO New England (ISO-NE) dataset[1] and the North-American Utility (NAU) dataset.[2] ISO-NE annually releases reports that provide hourly historical demand and electricity pricing data for its control area and eight load zones. This paper focuses solely on the control area dataset

[1] Available at https://www.iso-ne.com/isoexpress/web/reports/load-and-demand/-/tree/dmnd

[2] Available at https://class.ece.uw.edu/555/el-sharkawi/index.htm

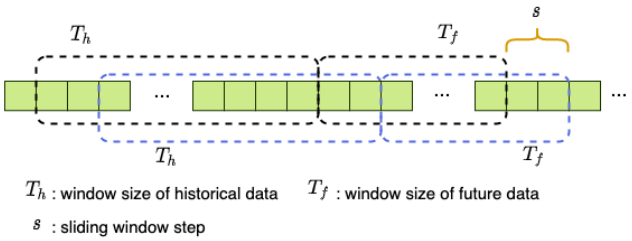


FIGURE 4. Illustration of the sliding window step for data processing.

TABLE 1. Time-related features of the ISO-NE dataset.

Input	Size	Description
$\mathbf{y}_h$	168×1	Historical target values
DaDemd	192×1	Day ahead demand
DryBulb	192×1	Dry bulb temperature
DewPnt	192×1	Dew point temperature
Weekday	192×1	Weekday or weekend indicator
Holiday	192×1	Holiday or non-holiday indicator
Season	192×4	One-hot encoding
Hour of Day	192×24	One-hot encoding
Day of Week	192×7	One-hot encoding
Month of Year	192×12	One-hot encoding

and ignores the price-related features. The input features include day-ahead demand, dry bulb and dew point temperatures (in Fahrenheit), and time-related features. Data from 2015 to 2017 are used to train the forecasting module while 80% of 2018's data are randomly selected to train the error correction module. The remaining 20% data of 2018 are used for validation. The year 2019 is reserved for testing. The NAU dataset provides electricity load, temperature, and time information from January 1, 1985 to October 12, 1992. In our study, temperature and time-related features are considered. We use the data from 1987 to 1989 for training the forecasting model and randomly select 80% of 1990's data to train the error correction module. The remaining 20% of 1990 is for validation while the year 1991 is for testing.

B. DATA PREPARATION

The missing values in the datasets are filled by linear interpolation. We incorporate time-related features such as indicators of weekends and holidays, seasons, hour of day, day of week, and month of year. We embed categorical features via one-hot encoding and standardize numerical features by subtracting their means and dividing by their standard deviations. The framework forecasts next 24 hours load demands using the previous seven days load and features. The next 24 hours features are assumed to be available as model's input. For hourly data, we have $T_h = 168$ and $T_f = 24$. As shown in Fig. 4, the sliding window size is set to be 1 and 24 for training the forecasting module and error correction module, respectively. We list the model inputs from the ISO and NAU datasets in Tables 1 and 2.

TABLE 2. Time-related features of the NAU dataset.

Input	Size	Description
$\mathbf{y}_h$	168×1	Historical target values
Temperature	192×1	Historical and future temperature
Holiday	192×1	Holiday or non-holiday indicator
Season	192×4	One-hot encoding
Hour of Day	192×24	One-hot encoding
Day of Week	192×7	One-hot encoding
Month of Year	192×12	One-hot encoding

C. BASELINE MODELS AND HYPERPARAMETERS

To verify the effectiveness of our proposed framework, we compare four different types of models: classic machine learning models, DBN-based models, RNN-based models, and Transformer-based models. Details are given in below.

- Classic machine learning model: We test SVR [5], RF [6], and GBM [8] using Scikit-Learn 0.23.2. Inputs to the model are historical and feature features and historical load that are flattened as a 1-D vector.
- DBN-based model: DBNs [30] are generative neural networks composed of multiple layers of RBMs. Each RBM layer is pre-trained in an unsupervised manner using the contrastive divergence algorithm, and the overall model is fine-tuned using supervised learning. Rough autoencoder combines rough set theory with DBNs which can effectively handle uncertain and noisy data and learn complex patterns [10].
- RNN-based model: The CNN-LSTM [31] combines the advantages of both CNN and LSTM layers to improve forecasting accuracy. Attention-based load forecasting (ANLF) [27] is based on the encoder-decoder biLSTM architecture and utilizes a dynamic feature selection layer within the encoder. These models have shown promising results in load forecasting and can be used as effective baselines for future research in this field.
- Transformer-based model: Informer is a transformer-based model designed for time-series forecasting as proposed in the 2021 AAAI Best Paper [32]. Unlike the RNN-based model, transformer can handle sequential data in parallel to reduce training time.

The computing environment is a machine with 3.7 GHz Intel Core i7-8700K Six-Core and NVIDIA GeForce GTX 1080 Ti (11GB GDDR5X). Deep learning based models are trained by using Adam optimizer and implemented with PyTorch 1.6.0. The initial learning rate is 0.001 which decays by 0.1 times for every 30 epochs. Early stopping criteria is set with patience 30. All models share the same training, validation, and testing data samples and input features for fair comparisons. We perform a grid search to identify the best hyper-parameter set based on the validation data. The grid search is commonly used for hyper-parameter tuning, which involves setting a range of values for each hyper-parameter and testing all possible combinations. The details of the grid search are given in Table 3.

TABLE 3. The ranges of hyper-parameter tuning. Bold and italic fonts indicate the best values for the ISO-NE and the NAU datasets, respectively.

Model	Hyper-parameter range
SVR [5]	Kernel = (*RBF*, Linear, **Poly**)
	Degree = (2, 3) with Poly
	Gamma = (*auto*, **scale**) with Poly/RBF
	C = (0.1, 1, ***10***)
RF [6]	Number of estimators = (100, *500*, **1000**)
	Maximum depth = (5, 10, ***20*** None)
	Minimum samples split = (***2***, 5)
	Minimum samples leaf = (***1***, 3, 5, 10)
GBM [8]	Loss = (***ls***, lad, huber, quantile)
	Learning rate = (*0.1*, **0.01**)
	Number of estimators = (100, 500, ***1000***)
	Maximum depth = (***5***, 10, 20, None)
	Minimum samples split = (*2*, **5**)
	Minimum samples leaf = (1, 3, *5*, **10**)
DBN [30]	Hidden layer 1 = (*128*, **256**, 512)
	Hidden layer 2 = (*128*, **256**, 512)
	Hidden layer 3 = (***0***, 64, 128)
	Batch = (***64***, 128)
RAE [10]	Hidden layer 1 = (128, ***256***, 512)
	Hidden layer 2 = (128, ***256***, 512)
	Hidden layer 3 = (***0***, 64, 128)
	Batch = (64, ***128***)
CNN-LSTM [31]	Batch = (**64**, *128*)
	Hidden size = (128, 256, ***512***)
	Kernal size = (*3*, 5, **8**)
ANLF [27]	Batch = (64, ***128***)
	Hidden size = (***128***, 256, 512)
Informer [32]	Batch = (***64***, 128)
	Hidden size = (**64**, *128*, 256, 512)
	label length = (0, 24, ***48***)
	number of attention heads = (2, 4, ***8***)
PM-LSTM	Batch = (*64*, **128**)
	Hidden size = (**128**, *256*, 512)
	λ = (0, ***0.001***, 0.01)
PM-GRU	Batch = (64, ***128***)
	Hidden size = (*128*, 256, **512**)
	λ = (***0***, 0.001, 0.01)

D. PERFORMANCE METRICS

The mean absolute error (MAE) and mean absolute percentage error (MAPE) are used to evaluate the forecasting accuracy. They are defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|, \tag{10a}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{10b}$$

where y_i and $\hat{y}_i$ are the i-th true and predicted outputs. n is the number of points in the testing horizon.

IV. SIMULATION RESULTS

In this section, three case studies are carried out to show the effectiveness of the proposed framework. Case 1 shows the ablation study results. Case 2 compares the baseline models in section III-C and our proposed model. Case 3 shows the generalization capability, for which we add the feature weighting mechanism and error correction module to the Informer.

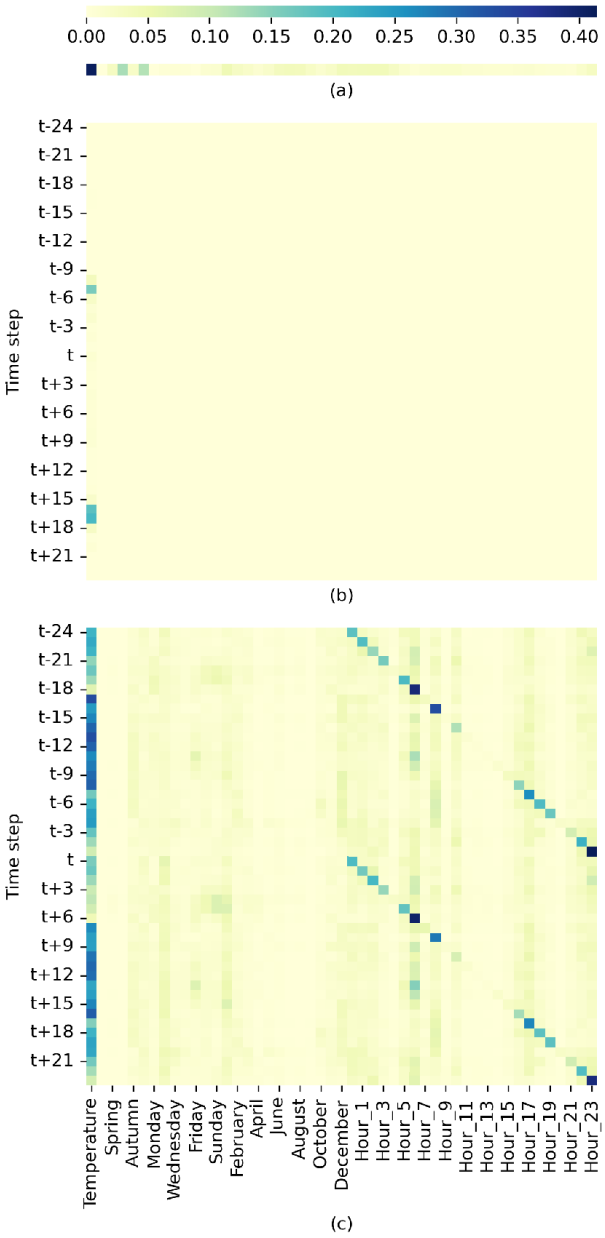


FIGURE 5. Two-day feature weight visualization for the NAU dataset with different approaches: (a) mutual information, (b) random forest, and (c) the proposed feature weighting attention.

A. CASE 1: ABLATION STUDY AND DISCUSSION

An ablation study is conducted based on the NAU dataset. Table 4 presents the MAE and MAPE results. The first row shows the performance of the backbone encoder-decoder based BiLSTM model. We then compare three different approaches of feature weighting: mutual information

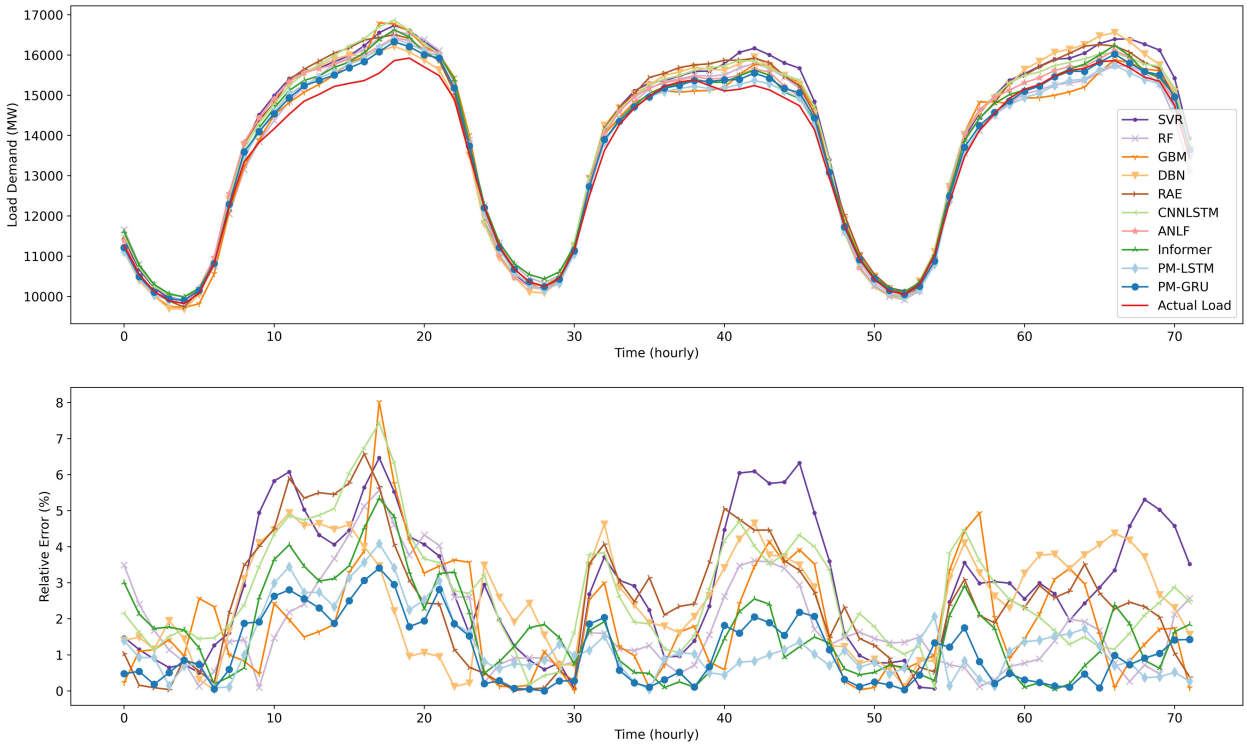


FIGURE 6. Forecasting curve and relative error for the ISO-NE dataset (3 days).

(MI) [19], random forest (RF) [21], and our proposed feature weighting attention (FW). In addition, we evaluate the performance of the backbone model with two types of temporal attention mechanisms: single layer temporal attention (TA) and the proposed hierarchical temporal attention, which incorporates similar day information (SDA). Finally, we include the results for two error correction methods: the baseline ARIMA model (BL) [23] and our proposed feature reinforced error correction model (EC). The results show that the proposed feature weighting and error correction outperform the existing methods. Each individual module improves the accuracy of the backbone model. Moreover, the combination of these modules further enhances the performance. Compared with the other competing alternatives, our proposed framework achieves a significant improvement in accuracy.

The interpretability of the performance improvements can be visualized in Fig 5. First, the proposed weighting attention can identify feature importance in the time domain. Fig 5(c) shows that our method adds time-varying weights on different features while mutual information approach exerts time-invariant weights shown in Fig 5(a). Second, our method puts more accurate weights on each feature compared with RF. Our weight assignments are sparser, with higher weights on temperature and hour [cf. Fig. 5(c)]. In contrast, RF yields similar features weights at approximately 0.02 [cf. Fig. 5(b)]. Third, our approach shows a good response to the input changes while RF is ignorant of different input data. Finally, the clear pattern of feature weights in Fig. 5(c) shows that the temperature from 9 AM to 4 PM is a more critical factor, which reflects the reality.

TABLE 4. NAU dataset: Ablation study for the proposed framework. Acronyms: MI (mutual information feature weight), RF (random forest feature weight), FW (feature weighting attention), TA (temporal attention), SDA (similar day attention), BL (baseline ARIMA error correction) and EC (the proposed error correction).

MI [19]	RF [21]	FW	TA	SDA	BL [23]	EC	MAE	MAPE (%)
-	-	-	-	-	-	-	89.20	3.93
✓	-	-	-	-	-	-	59.93	2.57
-	✓	-	-	-	-	-	63.65	2.76
-	-	✓	-	-	-	-	56.65	2.45
-	-	-	✓	-	-	-	69.10	2.96
-	-	-	✓	✓	-	-	64.29	2.78
-	-	✓	✓	✓	-	-	48.96	2.15
-	-	✓	✓	✓	✓	-	46.36	2.09
-	-	✓	✓	✓	-	✓	**45.70**	**2.00**

B. CASE 2: LOAD FORECASTING MODEL COMPARISON

Besides our proposed model (FW+TA+SDA in Table 4) with LSTM (PM-LSTM) and GRU (PM-GRU) implementations, we test eight baseline models. To have a fair comparison, the error correction module is deactivated because it is not directly applicable to those classic machine learning algorithms such as SVR, RF and GBM. Table 5 and Table 6 show the forecasting error for the two datasets.

Among the three classic machine learning methods, GBM performs the best for the ISO-NE dataset, while SVR

TABLE 5. Forecasting errors over the year 2019 for the ISO-NE dataset.

Model	SVR [5]	RF [6]	GBM [8]	DBN [30]	RAE [10]	CNN-LSTM [31]	ANLF [27]	Informer [32]	PM-LSTM	PM-GRU
MAE	317.49	393.58	277.54	326.45	308.36	309.06	258.70	256.89	**229.47**	**231.84**
MAPE (%)	2.33	2.87	2.00	2.38	2.22	2.27	1.88	1.89	**1.66**	**1.67**

TABLE 6. Forecasting errors over the year 1991 for the NAU dataset.

Model	SVR [5]	RF [6]	GBM [8]	DBN [30]	RAE [10]	CNN-LSTM [31]	ANLF [27]	Informer [32]	PM-LSTM	PM-GRU
MAE	67.97	99.91	83.05	68.33	63.55	61.85	58.65	57.13	**48.96**	**46.42**
MAPE (%)	2.98	4.40	3.61	2.99	2.79	2.73	2.58	2.49	**2.15**	**2.03**

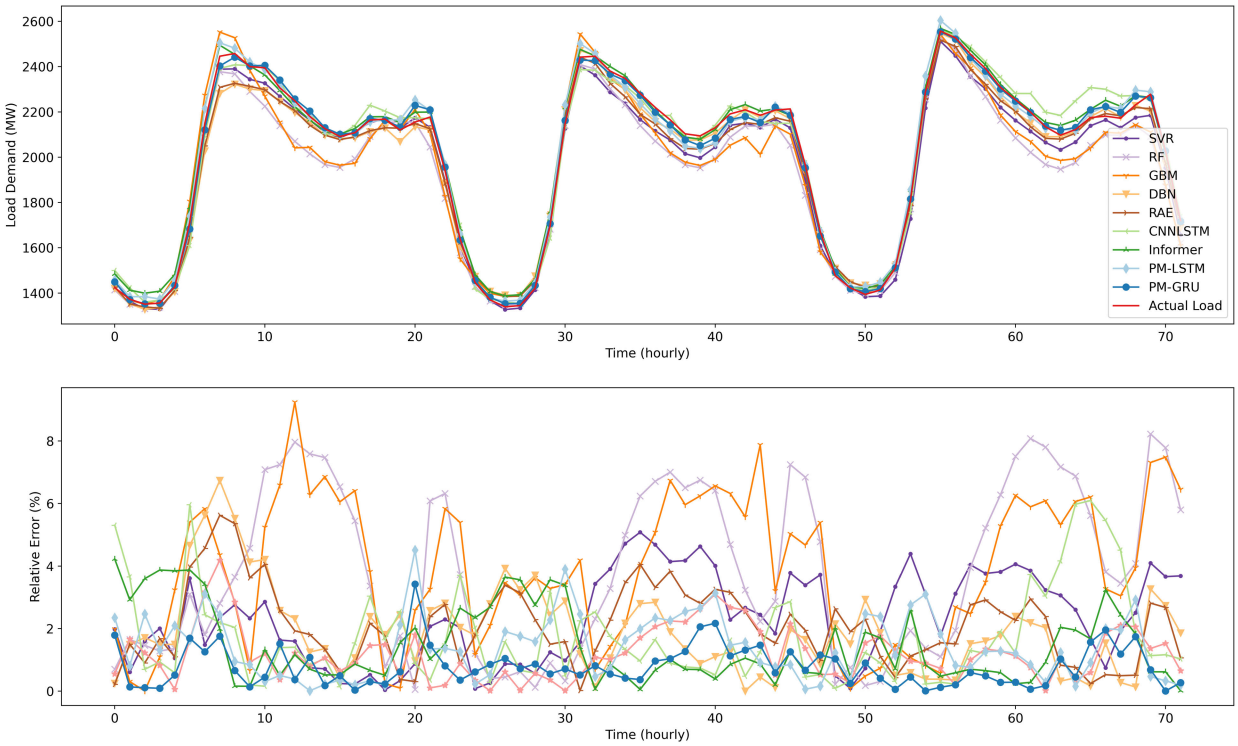


FIGURE 7. Forecasting curve and relative error for the NAU dataset (3 days).

stands out for the NAU dataset. DBN-based models attain results that are comparable to CNN-LSTM for both datasets. Notably, our proposed models PM-LSTM/GRU consistently outperform all the other models for both datasets with the smallest forecasting errors. Moreover, comparing the ANLF model with the proposed ones, extracting the feature weighting layer further improves the accuracy and increases the model's generalization capability. Overall, these findings highlight the effectiveness of our proposed approach. The detailed forecasting performance over three days are given in Fig. 6 and Fig. 7, where the relative error (RE) between the forecast value $\hat{y}$ and the true value y is defined as

$$\mathrm{RE} = \frac{|y - \hat{y}|}{y} \times 100\%. \tag{11}$$

C. CASE 3: GENERALIZATION CAPABILITY

To further show the generalization capability of the proposed framework, we apply both the feature weighting and error correction to the transformer-based Informer. The result is reported in Table 7 for the ISO-NE dataset. We compare the model itself with the model having feature weighting and/or error correction. The forecasting curves and relative errors are shown in Fig. 8. By the ablation study, the model with our proposed feature weighting and error correction mechanisms performs the best. This verifies the merit of integrating the feature weighting to provide more informative features and error correction to further improve the accuracy.

D. CASE 4: COMPUTATIONAL COMPLEXITY

In this section, we provide the big-$\mathcal{O}$ computational complexity analysis for the proposed framework. The feature

TABLE 7. ISO-NE dataset: Ablation study for Informer with feature reinforced error correction (EC) and/or feature weighting attention (FW).

FW	EC	MAE	MAPE (%)
-	-	256.89	1.89
✓	-	249.17	1.81
-	✓	239.35	1.76
✓	✓	**239.23**	**1.74**

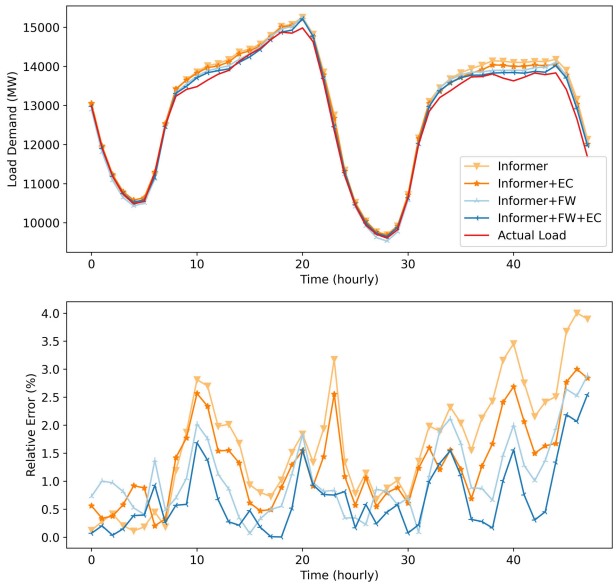


FIGURE 8. ISO-NE dataset: The forecasting curves and relative errors for Informer, Informer with error correction, Informer with feature weighting, and Informer with both feature weighting and error correction (for 2 days).

weighting module has a computational complexity of $\mathcal{O}(nd_h^{fw})$. The load forecasting model consists of the encoder and decoder BiLSTM whose complexity is $\mathcal{O}(T_h h_s(h_s+n))$ and $\mathcal{O}(T_f h_s(h_s+n))$, respectively [33]. The hierarchical temporal attention that forms an additional input to the decoder BiLSTM has a complexity of $\mathcal{O}(T_h d_h^{att} + d_h^{att}(h_s+n))$. The output layer is in the $\mathcal{O}(d_h^o h_s)$. Considering all these components, the overall complexity of the load forecasting model is $\mathcal{O}(T_h h_s(h_s+n)+T_f h_s(h_s+n+T_h d_h^{att}+d_h^{att}(hs+n))+d_h^o h_s)$. If $d_h^{fw} = d_h^{att} = d_h^o =: d_h$, then we have the overall computational complexity $\mathcal{O}(T_h h_s(h_s+n) + T_f h_s d_h(T_h + h_s + n))$.

V. CONCLUSION

This paper develops a unifying deep learning framework for multi-horizon STLF. Three interactive modules are developed with high generalization capability, which includes the feature weighting mechanism, STLF model and error correction module. In the proposed framework, the feature weighting mechanism is designed to provide informative input features for both historical and future time horizons. The STLF model with a hierarchical temporal attention layer decodes the next-day load with the future input features and similar temporal information. The hierarchical temporal attention layer provides a natural way to incorporate similar day information. In addition, the error correction module is developed based on transfer learning. It can reuse the learned hidden feature extraction to reduce the training cost. The modular design of our framework facilitates customization and independent modification. The extensive simulation results tested on the two datasets corroborate the merits of our framework. The codes of this work are available at https://github.com/jxiong22/STLF_framework.

REFERENCES

[1] C. Feng, M. Sun, and J. Zhang, "Reinforced deterministic and probabilistic load forecasting via *Q*-learning dynamic model selection," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1377–1386, Mar. 2020.

[2] X. Kong, C. Li, C. Wang, Y. Zhang, and J. Zhang, "Short-term electrical load forecasting based on error correction using dynamic mode decomposition," *Appl. Energy*, vol. 261, Mar. 2020, Art. no. 114368.

[3] G. Juberias, R. Yunta, J. G. Moreno, and C. Mendivil, "A new ARIMA model for hourly load forecasting," in *Proc. IEEE Transmiss. Distrib. Conf.*, vol. 1, Apr. 1999, pp. 314–319.

[4] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Trans. Power Syst.*, vol. 18, no. 3, pp. 1014–1020, Aug. 2003.

[5] E. Ceperic, V. Ceperic, and A. Baric, "A strategy for short-term load forecasting by support vector regression machines," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4356–4364, Nov. 2013.

[6] G. Dudek, "Short-term load forecasting using random forests," in *Intelligent Systems*. Cham, Switzerland: Springer, 2015, pp. 821–828.

[7] Y.-Y. Cheng, P. P. K. Chan, and Z.-W. Qiu, "Random forest based ensemble system for short term load forecasting," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 1, Jul. 2012, pp. 52–56.

[8] S. Ben Taieb and R. J. Hyndman, "A gradient boosting approach to the kaggle load forecasting competition," *Int. J. Forecasting*, vol. 30, no. 2, pp. 382–394, Apr. 2014.

[9] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2627–2633.

[10] M. Khodayar, O. Kaynak, and M. E. Khodayar, "Rough deep neural architecture for short-term wind speed forecasting," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 2770–2779, Dec. 2017.

[11] M. Khodayar, J. Wang, and M. Manthouri, "Interval deep generative neural network for wind speed forecasting," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3974–3989, Jul. 2019.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 5998–6008.

[16] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proc. SSST-8, 8th Workshop Syntax, Semantics Struct. Stat. Transl.*, 2014, pp. 103–111.

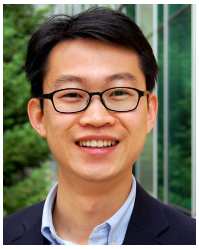
[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[18] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 143–159, Aug. 2002.

[19] N. Ghadimi, A. Akbarimajd, H. Shayeghi, and O. Abedinia, "Two stage forecast engine with feature selection technique and improved meta-heuristic algorithm for electricity load forecasting," *Energy*, vol. 161, pp. 130–142, Oct. 2018.

[20] D. Panday, R. Cordeiro de Amorim, and P. Lane, "Feature weighting as a tool for unsupervised feature selection," *Inf. Process. Lett.*, vol. 129, pp. 44–52, Jan. 2018.
[21] Y. Xuan, W. Si, J. Zhu, Z. Sun, J. Zhao, M. Xu, and S. Xu, "Multi-model fusion short-term load forecasting based on random forest feature selection and hybrid neural network," *IEEE Access*, vol. 9, pp. 69002–69009, 2021.
[22] Y. Deng, B. Wang, and Z. Lu, "A hybrid model based on data preprocessing strategy and error correction system for wind speed forecasting," *Energy Convers. Manage.*, vol. 212, May 2020, Art. no. 112779.
[23] J. Duan, H. Zuo, Y. Bai, J. Duan, M. Chang, and B. Chen, "Short-term wind speed forecasting using recurrent neural networks with error correction," *Energy*, vol. 217, Feb. 2021, Art. no. 119397.
[24] H. Liu and C. Chen, "Multi-objective data-ensemble wind speed forecasting model with stacked sparse autoencoder and adaptive decomposition-based error correction," *Appl. Energy*, vol. 254, Nov. 2019, Art. no. 113686.
[25] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[26] L. Cai, J. Gu, and Z. Jin, "Two-layer transfer-learning-based architecture for short-term load forecasting," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1722–1732, Mar. 2020.
[27] J. Xiong, P. Zhou, A. Chen, and Y. Zhang, "Attention-based neural load forecasting: A dynamic feature selection approach," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Jul. 2021, pp. 1–5.
[28] C. Feng and J. Zhang, "Hourly-similarity based solar forecasting using multi-model machine learning blending," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Aug. 2018, pp. 1–5.
[29] M. Barman, N. B. Dev Choudhury, and S. Sutradhar, "A regional hybrid GOA-SVM model based on similar day approach for short-term load forecasting in Assam, India," *Energy*, vol. 145, pp. 710–720, Feb. 2018.
[30] A. Dedinec, S. Filiposka, A. Dedinec, and L. Kocarev, "Deep belief network based electricity load forecasting: An analysis of Macedonian case," *Energy*, vol. 115, pp. 1688–1700, Nov. 2016.
[31] S. H. Rafi, Nahid-Al-Masood, S. R. Deeba, and E. Hossain, "A short-term load forecasting method using integrated CNN and LSTM network," *IEEE Access*, vol. 9, pp. 32436–32448, 2021.
[32] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI*, 2021, pp. 1–10.
[33] M. Rotman and L. Wolf, "Shuffling recurrent neural networks," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 11, pp. 9428–9435.
[34] G. A. N. Mbamalu and M. E. El-Hawary, "Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation," *IEEE Trans. Power Syst.*, vol. 8, no. 1, pp. 343–348, 1993.
[35] J.-F. Chen, W.-M. Wang, and C.-M. Huang, "Analysis of an adaptive time-series autoregressive moving-average (ARMA) model for short-term load forecasting," *Electric Power Syst. Res.*, vol. 34, no. 3, pp. 187–196, Sep. 1995.
[36] M. Zhang, Z. Yu, and Z. Xu, "Short-term load forecasting using recurrent neural networks with input attention mechanism and hidden connection mechanism," *IEEE Access*, vol. 8, pp. 186514–186529, 2020.
[37] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," *Int. J. Forecasting*, vol. 32, no. 3, pp. 896–913, Jul. 2016.
[38] Z. Li, L. Ye, Y. Zhao, X. Song, J. Teng, and J. Jin, "Short-term wind power prediction based on extreme learning machine with error correction," *Protection Control Modern Power Syst.*, vol. 1, no. 1, pp. 1–8, Dec. 2016.
[39] D. Wu, B. Wang, D. Precup, and B. Boulet, "Multiple kernel learning-based transfer regression for electric load forecasting," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1183–1192, Mar. 2020.
[40] E. Lee and W. Rhee, "Individualized short-term electric load forecasting with deep neural network based transfer learning and meta learning," *IEEE Access*, vol. 9, pp. 15413–15425, 2021.
[41] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.
[42] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–13, 2018.
[43] Q. Cao, B. T. Ewing, and M. A. Thompson, "Forecasting wind speed with recurrent neural networks," *Eur. J. Oper. Res.*, vol. 221, no. 1, pp. 148–154, Aug. 2012.
[44] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[45] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.
[46] U. Stańczyk, "Feature evaluation by filter, wrapper, and embedded approaches," in *Feature Selection for Data and Pattern Recognition*. Berlin, Germany: Springer, Dec. 2014, pp. 29–44.
[47] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943–3952, Jul. 2019.
[48] S. Wang, X. Wang, S. Wang, and D. Wang, "Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting," *Int. J. Electr. Power Energy Syst.*, vol. 109, pp. 470–479, Jul. 2019.
[49] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang, "Multi-horizon time series forecasting with temporal attention learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2527–2535.
[50] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, and V. A. Kamaev, "A survey of forecast error measures," *World Appl. Sci. J.*, vol. 24, no. 24, pp. 171–176, 2013.
[51] C. Feng, M. Cui, B.-M. Hodge, and J. Zhang, "A data-driven multi-model methodology with deep feature selection for short-term wind forecasting," *Appl. Energy*, vol. 190, pp. 1245–1257, Mar. 2017.
[52] Y. Saeys, T. Abeel, and Y. Van de Peer, "Robust feature selection using ensemble feature selection techniques," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, Sep. 2008, pp. 313–325.
[53] K. Chen, Y. Zhang, Q. Wang, J. Hu, H. Fan, and J. He, "Scale- and context-aware convolutional non-intrusive load monitoring," *IEEE Trans. Power Syst.*, vol. 35, no. 3, pp. 2362–2373, May 2020.

JING XIONG received the B.S. and M.S. degrees from the Department of Control and Computer Engineering, North China Electric Power University. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California, Santa Cruz. Her research interests include big data analytics for smart power grids, load forecasting and monitoring, and event identification.



YU ZHANG (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Minnesota. He is currently an Assistant Professor with the Department Electrical and Computer Engineering, University of California, Santa Cruz (UCSC). Prior to joining UCSC, he was a Postdoctoral Researcher with UC Berkeley and the Lawrence Berkeley National Laboratory. His research interests include cyber-physical systems, smart power grids, optimization theory, machine learning and big data analytics. He received the Hellman Fellowship, in 2019. He was a co-recipient of the Early Career Best Paper Award from the Energy, Natural Resources, and the Environment (ENRE) Section, Institute of Operations Research and the Management Sciences (INFORMS), in 2021.