Short-term load demand forecasting through rich features based on recurrent neural networks

Dongbo Zhao | Qian Ge | Yuting Tian | Jia Cui | Boqi Xie | Tianqi Hong

1 Energy Systems Division, Argonne National Laboratory, Lemont, Illinois, USA
2 Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, North Carolina, USA
3 School of Electrical Engineering, Shenyang University of Technology, Shenyang, Liaoning, China
4 School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA

Abstract
With the emerging penetration of renewables and dynamic loads, the understanding of grid edge loading conditions becomes increasingly substantial. Load modelling researches commonly consist of explicitly expressed load models and non-explicitly expressed techniques, of which artificial intelligence approaches turn out to be the major path. This paper reveals the artificial intelligence-based load modelling technique to enhance the knowledge of current and future load information considering geographical and weather dependencies. This paper presents a recurrent neural network based sequence to sequence (Seq2Seq) model to forecast the short-term power loads. Also, a feature attention mechanism, which is along channel and time directions, is developed to improve the efficiency of feature learning. The experiments over three publicly available datasets demonstrate the accuracy and effectiveness of the proposed model.

1 INTRODUCTION

With the continuous development of economy, society and smart grid, the power system makes increasingly higher requirements on the accuracy of load forecasting [1]. An accurate short-term load prediction is an essential backbone of the smart grid operation, which can facilitate the economic dispatch, demand response program etc. [2]. However, the load characteristics of power systems are not the same in different regions and the increasing penetration of distributed renewable energy also brings great unpredictability and uncertainty to the load side of the grid. These place higher challenges to the power load forecasting [3].

Researchers have endeavoured to improve the accuracy and speed of the forecasting methodologies applied in power systems over the past decades [4, 5]. The statistical model is one of the most widely used forecasting methods, which aims to describe the relationship between the time series of the forecasted values and actual history load data [6]. The commonly used statistical methods include the time series model, the regression analysis model and Kalman filter model [7, 8]. Statistical models can solve the problem of forecasting delay effectively, but the accuracy of long-term forecasting is unsatisfactory. Artificial intelligence, another group of methods, is extensively applied for load forecasting [9–11], such as neural networks and support vector machine (SVM).

Neural networks are able to learn complicated non-linear relationship between input and output. They are simple and efficient techniques with good performance and have been widely applied in load and renewable energy forecasting [4]. For instance, authors in [10] present several improved models using artificial neural networks to predict the aggregated load demand with good performance. However, most neural network models require a large training set to prevent over-fitting and the training process is usually unstable [12]. SVMs work well with small training set and are able to model non-linear relationship through the kernel trick, but choosing a good kernel function is non-trivial [13]. Authors in [14] propose a least-squares
SVM for annual electric load forecasting. Recently, deep learning approaches have attracted lots of attentions and have made tremendous achievements in the forecasting area. They have the capability of learning and generalisation on large dataset and have made several breakthroughs in the fields of computer vision and natural language processing [15, 16]. The core idea of deep learning approaches is to model complicated relationship between input and output data by stacking several layers of non-linearities. When applied to load forecasting tasks, instead of designing models manually, these powerful models are able to learn the hidden relationship between prediction and historical data automatically given large enough training data. Recurrent neural networks (RNNs), one of the deep learning models, is designed to model the time dependency of data sequence, which is usually applied in forecasting tasks [17]. However, the vanilla RNNs suffer from gradient vanishing for large recurrent steps, which prevents vanilla RNNs modelling long term time dependency of data. In [18], a long short-term memory (LSTM) cell is proposed to avoid this by applying gates mechanism. Authors in [11] present an LSTM RNN based framework to forecast the electric use of individual residential customers. Reference [19] also applied the RNN with LSTM unit for household load forecasting. In [20], gated recurrent units (GRUs) are proposed with a simpler architecture, but provides comparable performance with LSTM. Authors in [21] have adopted the gated RNN model for day-ahead load forecasting in commercial buildings.

In this paper, we proposed a forecasting framework with the sequence to sequence (Seq2Seq) RNN, which has not been adopted for load forecasting so far. The Seq2Seq model is originally proposed to address the end-to-end sequence learning problem with neural networks [22]. This model is able to handle various length of input and output sequence and provide good performance on machine translation tasks. In [23], an attention mechanism is proposed to let the Seq2Seq model to soft select a set of relevant inputs when predicting each output. This improves the machine translation performance especially for long sentence cases as well as provides interoperability of the model. Besides machine translation and speech recognition [24], Seq2Seq has been successfully applied to several other tasks including image captioning [25], answering questions [26], video representation learning [27] and human motion prediction [28].

Many research achievements on feature selection have been carried out to reduce the computational complexity as well as improve the performance of downstream tasks by removing redundant features [29]. In [30], a subset of weather and historical load features is extracted for load forecasting by ranking candidate features using a conditional mutual information approach. In [31], the optimal feature subset is selected based on the correlation between features and the load. In the proposed model, instead of selecting a fixed set of features for forecasting, different sets of features are selected to predict the power load at different time step through our feature attention mechanism.

The main contributions of this article can be summarised as follows,

1. An RNN-based Seq2Seq model is proposed to predict short-term loads. The Seq2Seq model has rarely been applied in the load forecasting area and this paper will fill the knowledge gap in this domain. The method is suitable for the load forecasting purpose and is demonstrated to be accurate and efficient
2. Feature attention mechanism is involved to learn rich feature representations. Unlike the original Seq2Seq where the decoder is only fed with the previous output, in our model at each output time, the decoder also takes features that are computed from raw input features through a feature attention layer as input. Specifically, a new set of features are computed from the raw features along with both channel and time directions before fed into the decoder

The rest of the paper is organised as follows. Section 2 presents the background knowledge of RNNs and GRUs. Section 3 describes the proposed Seq2Seq method for load forecasting. Section 4 discusses the feature attention mechanism applied in our model. Section 5 documents the solution procedure and the experiment setup. Case studies and results are presented in Section 6. Finally, Section 7 concludes this paper.

2 | BACKGROUND

In this section, we briefly introduce the background knowledge of RNNs [32] and GRUs [20] used in our model and more details can be found in [32, 20].

2.1 | Recurrent neural networks

RNN is a neural network for modelling sequential data by taking an input sequence with length $T$ to generate $T$ steps output sequence and hidden states. At each time step $t$, RNN takes current input $x_t$ and the previous hidden state $h_{t-1}$ to produce $h_t$ and the output $y_t$. Thus, it is able to model the time dependencies of the data and is suitable to capture the temporal relationship between the previous and current states. Traditionally, the RNN cell is defined as [17]:

$$
\begin{align*}
    h_t &= \tanh (W' h_{t-1} + U x_t + b) . \\
    y_t &= V h_t + c .
\end{align*}
$$

where $\tanh ()$ is the hyperbolic tangent function, and $W'$, $U$ and $V$ are weight matrices along with bias vectors $b$ and $c$, which are usually learned through backpropagation through time (BPTT) algorithm [33] during training.

2.2 | Gated recurrent units

The vanilla RNNs can suffer from vanishing/exploding gradients, thus have difficulty to learn long-term dependencies. Some sophisticated recurrent cells including LSTM [18] and GRU [20]
are proposed to avoid this issue by creating paths through time with derivatives that neither vanish nor explode [17]. GRU is applied in our proposed model and the definition is expressed as follows [34],

$$
\begin{align*}
    z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z), \\
    r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\
    f_t &= \tanh(W x_t + U(r_t \odot h_{t-1}) + b_f), \\
    h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot f_t, \\
    y_t &= h_t.
\end{align*}
$$

where $\sigma$ is the logistic sigmoid function, $\odot$ is an element-wise multiplication, $W_z$, $U_z$, $W_r$, $U_r$ and $U$ are weight matrices along with bias vectors $b_z$, $b_r$, and $b_f$ to be learned.

GRU is chosen for our model because we found GRU outperformed LSTM during the experiments. There is no rule that one cell is always better than another one. It depends on the dataset used for modelling. Thus, a practical strategy is to test both cells for a new dataset then choose the one providing the best performance. However, when both cells have similar performance, GRU is usually preferred, as GRU is computationally more efficient.

## 3 LOAD FORECASTING USING RECURRENT NEURAL NETWORKS

Dynamic power load sequence usually can be considered as a time series, thus RNN is a suitable tool for dynamic load forecasting. In our work, the Seq2Seq model is applied and the feature engineering is also involved. In this section, we first introduce the general model of power load. Then we discuss how to use a Seq2Seq framework to learn the load model and forecast the load. Finally, we introduce our feature attention mechanism which reduces the feature engineering efforts.

### 3.1 Power and load model

Let $y_t$ and $y_t^{true}$ be the measured value and true value of a power load at time $t$ respectively. Then $y_t$ can be modelled as

$$
y_t = y_t^{true} + N,
$$

where $N \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian noise with a zero mean and a standard derivation $\sigma$ introduced by the measurement uncertainty.

Let $x_t$ be a subset of features that affects the value of $y_t^{true}$, and $f(x_t, \omega)$ be a load model modelling $y_t^{true}$ for all $t$ with parameters $\omega$. Then the likelihood of $y_t$ given features $x_t$ is

$$
p(y_t | x_t) = \mathcal{N}(y_t; f(x_t, \omega), \sigma_{model}^2),
$$

where $\sigma_{model}$ is a standard deviation with a lower bound $\sigma$ as the load may not be perfectly modelled by only using feature set $x_t$. Assume $\sigma_{model}$ is independent on $x_t$, our goal is to find a function $f(x_t, \theta)$ which approximates the load model $f$ given sufficient observations. In this paper, we use RNNs consisting of an encoder and a decoder as the function approximator and the load model is fitted by the maximum likelihood estimation based on Equation (9). Because of the time dependence property of the load sequence, feature set $x_t$ can include historical load data, weather temperature, and any other external features which can affect the load value at time $t$.

### 3.2 Learning load model through a sequence to sequence model

Seq2Seq [22] is a popular RNN based model widely used in natural language processing tasks [35–37]. It consists of an encoder RNN and a decoder RNN and is able to map an input sequence to a target sequence with different lengths. The encoder first maps an arbitrary length input sequence to a fixed-size vector (the last hidden state of encoder RNN). Then the decoder is initialised by the fixed-size vector encoder provided and generates a target sequence. One advantage of this model for load forecasting is that the entire information of the input sequence is utilised to generate the target sequence. Assuming the load sequence with length $T$ only depends on the previous historical data with length $T+1$ and corresponding external features, the likelihood of the target load sequence given feature set $x_t$ is [15]

$$
p(y_{t+1} \ldots y_{T+1} | x_{t+1} \ldots x_T).
$$

where $t$ is the fixed-size encoded input sequence, $d_t$ is the multi-channel input features at time $t$ and each $p(y_{t+i} | h_{t+i}, y_{t+i-1}, d_{t+i})$ is a Gaussian distribution as discussed in Section 3.1. Unlike the original Seq2Seq where the decoder is only fed with the previous output, in our model at each output time $t$, the decoder also takes features $d_t$ as input. The features are computed from original input features through a feature attention layer, which will be discussed in detail in the next section. The load model then can be learned by maximising the log-likelihood of Equation (12) through a mean squared error (MSE) loss function:

$$
\text{MSE} = \frac{1}{T} \sum_{i=1}^{T} \| y_{t+i} - \hat{y}_{t+i} \|^2.
$$

where $\hat{y}_{t+i}$ is computed from the decoder RNN output followed by a linear fully connected layer. It is observed in [22] that the reversed order of input is preferred for language translation tasks. Inspired by this, the encoder is constructed by a
FIGURE 1 The proposed network for load forecasting

bidirectional RNN [38] to maintain both directions of the input sequence. Figure 1 illustrates the proposed model used for the load forecasting. Note that the encoder is a bidirectional RNN and the decoder is a single directional RNN.

4 | FEATURE ATTENTION MECHANISM

The process of feature engineering which heavily relies on experts’ domain knowledge is often time-consuming and inflexible in adding new types of features. To reduce the efforts on feature engineering, we introduce a feature attention mechanism which is able to learn rich feature representations automatically and efficiently. The only required feature engineering is to group features based on their dependency.

An attention mechanism is proposed for Seq2Seq in [35] to force the model to learn to focus on a specific position of the encoder input sequence. Inspired by this, all the features are fed into the decoder in our model, we introduce an attention mechanism to force the model to learn which input features are important for output at each output time $t$. The input features of our Seq2Seq model is a multiple channel feature sequence including historical load data (encoder input sequence) as well as external features such as weekdays, temperature and any other features affecting the load values at each time step. Though the input sequence is already encoded in the final state of the encoder, we found explicitly including it as an additional feature channel improves the load forecasting performance in our experiment, as the load demands between two successive days at the same time are highly correlated. This can be interpreted as manually specifying the attentions which are learned in [35].

We then assume that at each output time $t$, the model should focus on a specific set of features over time and channels. For example, along with time dimension, the load at time $t$ can depend on previous input features or even the future features. This is reasonable for the load forecasting purpose, for example, the future temperatures can affect the load demand at a particular time of a day because of some human planning behaviours. Similarly, it is also true for the channel dimension.

The feature attention learning is only applied for numerical features. That is, category and ordinal feature channels are fed into the decoder directly. Specifically, let $D^{\text{num}} = [d^1_{\text{num}}, \ldots, d^M_{\text{num}}]$ be the original $T \times M$ input M-channel numerical feature matrix with each column an input numerical feature sequence and $D^{\text{num}} = [d^1_{\text{num}}, \ldots, d^N_{\text{num}}]$ be a $T \times N$ learned N-channel numerical feature matrix. Feature attention learns a set of weights $w$ such that

\[
d_i^n(t) = \sum_{j=1}^{T} \sum_{s=1}^{M} w_{ij}^{ns} (t') d_i^{ns} (t'), n = 1, \ldots, N \text{ and } t' = 1, \ldots, T.
\]

(15)

\[
s.t. \sum_{j=1}^{T} \sum_{s=1}^{M} w_{ij}^{ns} (t') = 1.
\]

(16)

where $d_i^n(t)$ and $d_i^n(t')$ are the $i$-th row and the $t$-th column element of $D^{\text{num}}$ and $D^{\text{num}}$, respectively. In the proposed approach, attention over channels and time are learned separately to reduce the computational complexity.

4.1 | Channel attention

Feature attention over channel is learned based on the feature dependency over channel. One example of channel dependent features are the temperatures obtained from several locally closed weather stations. Feature channels are firstly divided into $K$ groups based on their dependency. Then for each group, the dependency between channels is modelled by weighted summation over all the features within the group. It is desired to learn multiple sets of the weights, since there may exist multiple useful combinations of features across channels. To reduce the computational complexity, we further assume that within each group, features at different time are sharing the same weights. Thus, for each group, the channel attention output can consist of multiple channels, each of which is a weighted average over original feature channels.

Let $\delta_i = \{d_{i1}^{\text{num}}, \ldots, d_i^K\}$ be a set of $T \times 1$ single channel of input numerical feature sequence, $\{k_i, i = 1, \ldots, K\}$ be a partition of index set for $\delta_i$ based on the feature channel dependency, $|k_i|$ be the cardinality of the set $k_i$, and $n_i$ be the number of output channels per group. Then the feature attention over channel for each group is computed as

\[
d_{i1}^{c} = \sum_{s=1}^{n_i} w_{i}^{s} d_{i1}^{s}, i = 1, \ldots, K \text{ and } t' = 1, \ldots, n_i.
\]

(17)

\[
s.t. \sum_{s=1}^{n_i} w_{i}^{s} = 1.
\]

(18)

where $d_{i1}^{c}$ is the $i$-th output channel attention feature vector for $i$-th feature group. The constraint is obtained by a softmax [32] over the weights. Note that for groups with only one feature...
4.2 Time attention

Feature time attention is applied to each individual feature channel separately. Two types of structures are used to model the feature attention over time. Multiple channels output is used in both structures for the same reason as mentioned before. First, we use a linear projection structure to model the time attention weights depending on the output time \( t \). That is, at different output time \( t \), the model should focus on input features with different patterns. Let \( n_t \) be the number of output channels. The time attention for each feature channel \( d^c_{t,l} \) is modelled as

\[
d^c_{t,l}(t) = \sum_{i=1}^{T} w^{c,l}_{i}(t) d^c_{i,l}(t), \quad i = 1, \ldots, L; \text{and } l = 1, \ldots, n_t, \quad \text{s.t. } \sum_{i=1}^{T} w^{c,l}_{i}(t) = 1. \tag{19}
\]

where \( d^c_{i,l}(t) \) and \( d^c_{t,l}(t) \) are the \( t \)-th element of channel attention feature vector \( d^c_{i,l} \) and the corresponding \( t \)-th output time attention feature vector \( d^c_{t,l} \), respectively. The constraint is obtained by a softmax [39] over the weights.

Then we use a multi-scale weight sharing structure inspired by [40] to model the weights which are independent on output time \( t \). That is, at each time \( t \), the model should focus on the same set of features centring at time \( t \), which mimics the process of sliding window feature extraction. A convolutional layer is used to meet the requirement of weight sharing as well as force the model to focus on the features around current time \( t \). The convolutional layer structure used in our experiments is illustrated in Figure 2, which is a modified inception layer proposed in [40]. Three modifications are applied:

1. One dimensional filters are used for one-dimensional sequence data;
2. Pooling is removed because the localisation is critical for attention learning and we want to keep the original sequence length as well;
3. 1x1 filters are removed before larger filters since the dimension reduction is not necessary here (input feature has a single channel). However, we keep the parallel 1x1 filters to add more non-linearity.

4.3 Including original features

To force the model to focus on features at the current time as well, the original features are concatenated with the new features as the final decoder input feature. Thus, the final feature matrix is

\[
D = [D^o_{co}, D^o_{num}, D^o_{num}]. \tag{21}
\]

where \( D^o_{co} \) is the matrix of the original category and ordinal features with each column an individual feature channel. At each time step \( t \), the input feature vector \( d^o \) is the \( t \)-th row of matrix \( D \). With this setting, features can be easily added or removed by adding or removing corresponding channels from the original feature matrix before fed into the feature attention layer.

5 SOLUTION PROCEDURE AND EXPERIMENTS SETUP

5.1 Solution procedure

As illustrated in Figure 1, the encoder takes an arbitrary length of historical load sequence as input. The decoder RNN is initialised by the final state of the encoder and takes a set of feature sequences as input. Features are first fed into a feature attention layer to force the decoder to focus on important features over time and channels at each output time \( t \). Then the decoder generates the load sequence based on the learned features and previous outputs. Note that the input and output sequences can have different lengths. The load model is trained by finding a function \( \hat{f} \) which minimises the distance between the true load and the model predicted load on a training set.

5.2 Experiments

To demonstrate the efficiency of our model, we apply the model on the next day hourly load forecasting task and compare the
performance with some previous models which are designed specifically for load forecasting. It is suitable to have the load forecasting as the task for demonstrating the proposed feature attention learning, because load demand forecasting depends on the historical usage and also heavily depend on external features such as holidays, public events and temperatures [42].

In our experiments, three public datasets are used: Hourly loads of the Polish power system from the period of 2002–2004 (PL) [43]; Global Energy Forecasting Competition 2012 from Kaggle competition (GEFCom2012) [44] and hourly loads of New England states from the period of 2011–2015 (ISO-NE) [45]. Depending on the data availability, historical load demand, weekday and temperature are used as the input features. Historical load demand and temperature are considered as numerical features and scaled to be zero mean and unit standard deviation. Weekday is considered as ordinal feature and scaled to the range [0, 1]. The three datasets correspond to three feature scenarios:

1. Only historical demand data is provided
2. External features are channel dependent
3. All external features are channel independent

5.2.1 Network architecture and training details

The choice of hyper-parameters and model training details are briefly discussed here. The decoder and each direction of the encoder in the Seq2Seq model consist of 2 layers with GRU of 80 hidden units. The number of output channels of inception filters is chosen to be 4 and 1 for other feature attentions.

During the training process, the network takes load demand of the current day and the corresponding features of the next day as input. Then it generates the estimated hourly load demand of the next day. The length of input and output sequences are not necessarily the same. However, in our experiments, no significant improvement is observed when a longer input sequence is provided at the cost of higher computational complexity. We have tested the proposed network using the demand of the previous week, the same day in the previous week, and the current day as input feature, and we found that the current setting provides the best performance. We further assume the next day temperatures can be predicted with high accuracy, so the true temperatures of the next day are used as the feature if applicable [46, 47]. The network is optimised by RMSProp algorithm [48] through BPTT [33]. As discussed in Section 3.2, the model assumes the load demand sequence to be predicted depending only on the input historical sequence of the encoder, but not on preceding historical load. Thus, we can safely shuffle the data of each training epoch to reduce the risk of over-fitting. At the beginning of each epoch, a set of forecasting pairs containing load demand of two successive days and corresponding features are first constructed. Then the forecasting pairs are shuffled as the training set for current epoch. The batch size used for PL, GEFCom2012 and ISO-NE are set to be 16, 8 and 1, respectively. The learning rate starts from 0.001 and is divided by 10 every 50 epochs. The models are trained for up to 150 epochs. The number of layers and hidden units, and the learning rate schedule are found by grid search on a system level load demand of ISO-NE. Other hyper-parameters are manually picked, since this work is not aiming to archive the SOTA performance and the current performance is satisfactory. Higher performance can be expected if all the hyper-parameters are chosen by grid search.

During the test, to obtain higher performance, the final prediction load is obtained by averaging the results of 10 independent experiments under different initialisation and randomisation.

The model is implemented in Python using the TensorFlow framework [49]. The experimental environment is CPU i7-2600 with 8GB RAM. It requires about 10 min, 20 min, and 2 h to train one load demand model for PL, GEFCom2012 and ISO-NE, respectively.

5.2.2 Evaluation metric

Mean absolute percentage error (MAPE) and root mean square error (RMSE) are used as the evaluation metrics in our experiments. Let \( N \) be the total number of samples to be predicted, \( y_i \) be the ground truth and \( \hat{y}_i \) be the predicted value. Then MAPE is defined as

\[
\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|.
\]  

and RMSE is defined as

\[
\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}}.
\]  

Since the true load demand \( y_i \gg 0 \) in all of the three datasets, we can safely use MAPE without numerical instability for forecasting evaluation. For both metrics, the lower value indicates the better performance.

6 | CASE STUDIES AND RESULTS

In this section, three case studies with different datasets are performed. The results demonstrate the effectiveness and the outstanding performance of our proposed model compared with some previous work. Discussions of feature attention learning for load forecasting and the extension of our model to other types of power system load modelling is also presented.

6.1 Case I: PL dataset

In the PL dataset, only load demand data from 1 January 2002 to 31 December 2004 is recorded. Time attention is applied to a single channel historical data. To compare with an artificial immune system based load forecasting model (AISLFS [50]),
TABLE 1  Forecasting performance on three dataset

|               | PL        |             | GEFCom2012  |             | ISO-NE     |             |
|---------------|-----------|-------------|-------------|-------------|------------|-------------|
|               | MAPE      | MAE         | MAPE        | MAE         | MAPE       | MAE         |
| Naive         | 6.19%     | 1506        | 8.40%       | 208         | 5.99%      | 1235        |
| RF [52]       | 1.96%     | 516         | 4.5%        | 113         | 2.56%      | 520         |
| MLP [53]      | 1.93%     | 478         | 3.81%       | 93          | 1.92%      | 398         |
| NN [51]       | 1.44%     | n/a         | n/a         | n/a         | n/a        | n/a         |
| AISLFS [50]   | 1.51%     | n/a         | n/a         | n/a         | n/a        | n/a         |
| ESN [54]      | n/a       | n/a         | 4.60%       | 104         | n/a        | n/a         |
| RNN [55]      | 1.78%     | 462         | 7.72%       | 188         | 2.19%      | 447         |
| Seq2SEquation (proposed) | 1.41% | 353 | 3.32% | 80 | 1.39% | 304 |

we use the same setting as described in their paper. That is, the training samples include load of 2002–2003 and the testing samples include the load of 2004 except days of public holidays. A neural network based forecasting model (NN [51]) is also included because it is reported as the best model on the PL dataset in [50].

The other four general forecasting approaches are included for comparison as well. Persistence forecasting (Naive) which predicts the next day load using the previous day load indicates how different the load demands are between two successive days and provides the lower bound of the performance. Random Forest (RF) [52] uses a rolling window with length 24 as features to predict the load value after 24 points of the window. In our experiments, 500 trees are used to ensure good forecasting performance. Multilayer perceptron (MLP) [53] containing 3 hidden layers with 129, 256, 128 hidden units takes the same input and output setting as the proposed model. To demonstrate the necessity of Seq2Seq model, we also use a single RNN with 2 layers of 150 GRU units to directly predict the load with time lag 24 from historical sequence.

The forecasting performance on the testing set is shown in the first two columns of Table 1, where Seq2Seq is the proposed model. We can see the proposed model has higher accuracy than RNN even with a smaller number of hidden units, because the Seq2Seq structure is able to encode the entire data from the previous day before prediction. Note that the hyper-parameters of RF, MLP and RNN are not fully tuned for the best performance. So better performance can be expected, but less likely to outperform the proposed model without non-trivial modifications. Also note that in Table 1, the performance evaluations of NN and AISLFS are reported in [50], while ESN is reported in [54]. Final results of RNN, MLP and proposed Seq2Seq are averaged from 10 independent experiments.

6.2 Case II: Global energy forecasting competition 2012 from Kaggle competition dataset

We then use the data from GEFCom2012 and train the model for forecasting the load at the system level (sum of all zonal level load demands). The dataset consists of hourly load data and hourly temperatures from 11 locally closed weather stations from 1 January 2004 to 30 June 2008. Thus, the hourly temperatures of 11 weather stations are channel dependent features and applied for channel attention learning. We use the same setting as [54] to compare with their work of fine-tuning parameters of forecasting algorithms through evolution. That is, training samples include data from the year 2004 to 2007 and testing samples include data from the year 2008. Over-fitting is observed when 80 hidden units are used, so 40 hidden units for both encoder and decoder are used for this case.

The forecasting performance on testing data is shown in the third and fourth column of Table 1. ESN is an echo state network based model [56] which is the best model on GEFCom2012 after parameters tuning through evolution reported in [54]. We notice that RNN has difficulty to model the load on this dataset. However, our Seq2Seq model based on RNNs still achieves the best performance.

Figure 3(a) shows the MAPE comparison of 10 independent experiments between using channel attention across 11 weather stations and using averaged temperatures of 11 weather stations. Feature attention across time is applied for both cases. We can see learning the channel attention weights does improve the accuracy. The learned weights for each weather station from 10 independent experiments are shown in Figure 4, where different colours indicate different experiments, and the height of the bar is the weight learned for each weather station. Notice that stations 9, 11, 5, 2 and 3 are always assigned higher weights than other stations, which demonstrates that our model does not simply learn a random set of weights. Thus, other than improving the performance, channel attentions also provide the importance of each weather station on GEFCom2012 forecasting task.

6.3 Case III: ISO-NE dataset

In this case, we use the data from ISO-NE and learn the model for forecasting the load at the system level (sum all zonal level load demand). The dataset consists of hourly historical data.
The comparison of forecasting performance on ISO-NE is shown in the last two columns of Table 1. Figure 6 shows the week of the best and the worst forecasting results over 2015. Each week starts from Monday to Sunday. Solid curves are the ground truth demands and dashed curves are the predicted demands. We can observe the higher errors exist around peak demand. In addition, we can see the trend of the load during one week is correctly predicted (demand is relatively low during weekends), which is the contribution of the weekday feature. Notice that the high load demand on Sunday of the week in Figure 6(a) is because of the high temperature on that day (84°F at the peak) and our model is able to capture this information correctly. Since US federal holidays are not used as the input feature, the model considers 4 July as an ordinary weekend and has a low prediction accuracy. Though adding this feature in our model is straightforward, we did not do this because this paper is not the discussion of feature selection for load demand forecasting and the contribution of the weekday feature has already demonstrated the capability of our model to encode this type of feature.

Figure 3(b) shows the MAPE comparison of with and without time attention from 10 independent experiments on ISO-NE (for both cases, all the hyper-parameters are the same as mentioned in Section 5.2.1 except the batch size is 8 and the number of hidden units is 40). Figure 5 shows the learned time attention weights. The learned weights for computing the new feature of forecast horizon at time $i$ is shown as the $i$-th row of the matrix. The element locating at the $i$-th row and the $j$-th column of the matrix indicates the importance of the feature at time $j$ for forecasting the load demand at time $i$. Figure 5(a) shows the learned time attention weights of dew point temperature. Other than the temperatures at current time, the model surprisingly pay more attention on the temperatures in the future. It is partially because we keep the original temperatures as another feature channel and the decoder itself is able to memorise past input, which lets the model learn the weights to focus on the future features. Another reason is the impact of human behaviours. One can decide whether to use a device based on the forecasted temperatures. Other than the temperatures at current time, the model surprisingly pay more attention on the temperatures in the future. It is partially because we keep the original temperatures as another feature channel and the decoder itself is able to memorise past input, which lets the model learn the weights to focus on the future features. Another reason is the impact of human behaviours. One can decide whether to use a device based on the forecasted temperatures.
learned from GEFCom2012 in Figure 5(b). A similar pattern is observed. However, after 7:00 PM, the model pays more attention to the temperatures in the past few hours. Thus, as well as improving the forecasting performance, the attention learning could guide the manual design of load models.

7 CONCLUSION AND DISCUSSION

In this work, we present a Seq2Seq based framework with GRU units for short-term load forecasting. This proposed framework also leverages feature engineering to improve the accuracy and interpretability of the forecasting model. Unlike the common Seq2Seq where the decoder is only fed with the previous output, the proposed model enables the decoder to accommodate features (as input) that are abstracted through computation from original input via the feature attention layer. Specifically, the new features at each output time are linear combinations of raw features along with both channel and time directions. Three publicly available datasets indicating different feature scenarios have been applied to demonstrate the effectiveness of the proposed model. The method has been compared with other dominant algorithms and the results validate that our proposed Seq2Seq model has a better performance than the state-of-the-art load forecasting models while requiring less effort in model design and feature engineering.

This proposed work can evolve to a generalised platform for load modelling and analysis, with the short-term load forecasting as just one of the typical applications. Load models can be presented and expressed explicitly in some cases when being able to perform component by component analysis, or equivalent representation formatting such as ZIP. But for most of the time, the load models for an aggregated node are not possible to be derived in an explicit manner. Our RNN-based framework learns the load models implicitly and can apply to multiple applications. For example, two Seq2Seq can be used to learn the active and reactive power of an exponential recovery load model [57]. The historical data can be previous active and reactive power sequences and the external feature can be the current...
or voltage sequence, which will result in other major applications in addition to load forecasting.

Our future work will focus on demonstrating the extension of the proposed framework while incorporating other information from power systems operation.

ACKNOWLEDGEMENTS
This work is supported by the U.S. Department of Energy Office of Electricity-Advanced Grid Modeling (AGM) Program. Dongbo Zhao and Qian Ge contributed equally to this work and should be considered as co-first authors.

ORCID
Tianqi Hong https://orcid.org/0000-0003-1307-6808

REFERENCES
1. Rahimi, F., Ipauchi, A.: Demand response as a market resource under the smart grid paradigm. IEEE Trans. Smart Grid 1(1), 82–88 (2010)
2. Zeng, B., et al.: Hybrid probabilistic-possibilistic approach for capacity credit evaluation of demand response considering both exogenous and endogenous uncertainties. Appl. Energy 229, 186–200 (2018)
3. Chan, S.C., et al.: Load/price forecasting and managing demand response for smart grids: Methodologies and challenges. IEEE Signal Process. Mag. 29(5), 68–85 (2012)
4. Antonanzas, J., et al.: Review of photovoltaic power forecasting. Sol. Energy 136, 78–111 (2016)
5. Hernandez, L., et al.: A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings. IEEE Commun. Surv. Tutorials 16.3, 1460–1495 (2014)
6. Wang, B., et al.: Study on weather typing based wind power prediction. Power System Technology 38(1), 93–98 (2014)
7. Pan, D.F., et al.: A wind speed forecasting optimization model for wind farms based on time series analysis and Kalman filter algorithm. Power System Technology 32(7), 82–86 (2008)
8. Zulaaga, C.D., et al.: Short-term wind speed prediction based on robust Kalman filtering: An experimental comparison. Appl. Energy 156, 321–330 (2015)
9. Kankal, M., et al.: Modeling and forecasting of Turkey’s energy consumption using socio-economic and demographic variables. Appl. Energy 88(5), 1927–1939 (2011)
10. Hernandez, L., et al.: Experimental analysis of the input variables’ relevance to forecast next day’s aggregated electric demand using neural networks. Energies 6, 2927–2948 (2013)
11. Kong, W., et al.: Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans. Smart Grid 10.1, 841–851 (2017)
12. Wu, Y., et al.: Short-term load forecasting based on improved grey relational analysis and neural network optimized by bat algorithm. Autom. Electr. Power Syst. 42, 67–74 (2018)
13. Bai, X., et al.: A spatial load forecasting method based on multilevel clustering analysis and support vector machine. Autom. Electr. Power Syst. 39(12), 56–61 (2015)
14. Li, H., et al.: Annual electric load forecasting by a least squares support vector machine with a fruit fly optimization algorithm. Energies 5, 4430–4445 (2012)
15. Yin, B., et al.: Review of deep learning. Journal of Beijing University of Technology 41(1), 48–59 (2015)
16. Yu, K., et al.: Deep learning: Yesterday, today, and tomorrow. Journal of computer Research and Development 50(9), 1799–1804 (2013)
17. Goodfellow, I., et al.: Deep Learning. MIT Press, Cambridge (2016)
18. Schmidhuber, J., Hochreiter, S.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997)
19. Shi, H., et al.: Deep learning for household load forecasting—A novel pooling deep RNN. IEEE Trans. Smart Grid 9.5, 5271–5280 (2017)
20. Cho, K., et al.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint (2014), arXiv:1409.1259
21. Cai, M., et al.: Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. Appl. Energy 236, 1078–1088 (2019)
22. Sutskever, I., et al.: Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems 27, 3104–3112 (2014)
23. Bahdanau, D., et al.: Neural machine translation by jointly learning to align and translate. arXiv preprint (2014), arXiv:1409.0473
24. Prabhavalkar, R., et al.: A comparison of sequence-to-sequence models for speech recognition. Interspeech 939–943 (2017)
25. Xu, K., et al.: Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning, 2048–2057 (2015)
26. Malinowski, M., et al.: Ask your neurons: A neural-based approach to answering questions about images. In: 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 1–9 (2015)
27. Srivastava, N., et al.: Unsupervised learning of video representations using lstms. In: International Conference On Machine Learning, 843–852 (2015)
28. Martinez, J., et al.: On human motion prediction using recurrent neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 4674–4683 (2017)
29. Xu, J., et al.: Semisupervised feature selection based on relevance and redundancy criteria. IEEE Transactions on Neural Networks and Learning Systems 28(9), 1974–1984 (2017)
30. Huang, N., et al.: Short term electrical load forecasting using mutual information based feature selection with generalized minimum-redundancy and maximum-relevance criteria. Entropy 18(9), 330 (2016)
31. Kouti, S., et al.: A new short-term load forecast method based on neuro-evolutionary algorithm and chaotic feature selection. Int. J. Electr. Power Energy Syst. 62, 862–867 (2014)
32. Rumelhart, D.E., et al.: Neurocomputing: Foundations of research. ch. Learning Representations by Back-propagating Errors, 096–099 (1988)
33. Rumelhart, D. E.: Parallel distributed processing: Explorations in the microstructure of cognition. Learning Internal Representations by Error Propagation 1, 318–362 (1986)
34. Chung, J., et al.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint (2014), arXiv:1412.3555
35. Luong, M.T., et al.: Effective approaches to attention-based neural machine translation. arXiv preprint (2015), arXiv:1508.04025
36. Wu, Y., et al.: Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint (2016), arXiv:1609.08144
37. Hermann, K. M., et al.: Teaching machines to read and comprehend. Advances in Neural Information Processing Systems 28, 1693–1701 (2015)
38. Schuster, M., Paliwal, K. K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45(11), 2673–2681 (1997)
39. Bishop, C.M., Pattern Recognition and Machine Learning. Springer, New York (2006)
40. Szegedy, C., et al.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 1–9 (2015)
41. Nair, V., Hinton, G. E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), 807–814 (2010)
42. Cui, J., et al.: Intelligent load pattern modeling and denoising using improved variational mode decomposition for various calendar periods. Appl. Energy 236, 1078–1088 (2019)
43. Czestochowa University of Technology http://gdudek.el.pcz.pl/varia/
44. http://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting
45. https://www.iso-ne.com/nsoexpress/web/reports/load-and-demand/tree/zone-info
46. Buitrago, J., Asfour, S.: Short-term forecasting of electric loads using nonlinear autoregressive artificial neural networks with exogenous vector inputs. Energies 10(1), 40 (2017)
47. Aprillia, H., et al.: Optimal decomposition and reconstruction of discrete wavelet transformation for short-term load forecasting. Energies 12(24), 4654 (2019)
48. Hinton, G.: Neural Networks for Machine Learning. Coursera, video lectures
49. Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. Software available from tensorflow.org, (2015)
50. Dudek, G.: Artificial immune system with local feature selection for short-term load forecasting. IEEE Trans. Evol. Comput. 21(1), 116–130 (2017)
51. Dudek, G.: Forecasting time series with multiple seasonal cycles using neural networks with local learning. In: International Conference on Artificial Intelligence and Soft Computing, Berlin, Heidelberg, 52–63 (2013)
52. Liaw, A., Wiener, M.: Classification and regression by random forest. R News 2(3), 18–22 (2002)
53. Haykin, S., Neural Networks and Learning Machines. Pearson Education India, (2010)
54. Hoverstad, B. A., et al.: Short-term load forecasting with seasonal decomposition using evolution for parameter tuning. IEEE Trans. Smart Grid 6(4), 1904–1913 (2015)
55. Mandic, D., Chambers, J.: Recurrent neural networks for prediction: Learning algorithms, architectures and stability. John Wiley & Sons, Inc., New Jersey (2001)
56. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304(5667), 78–80 (2004)
57. Arif, A., et al.: Load modeling—A review. IEEE Trans. Smart Grid 9(6), 5986–5999 (2018)

How to cite this article: Zhao D, Ge Q, Tian Y, Cui J, Xie B, Hong T. Short-term load demand forecasting through rich features based on recurrent neural networks. IET Gener Transm Distrib. 2021;15:927–937. https://doi.org/10.1049/gtd2.12069