KIPS Trans. Softw. and Data Eng.
Vol.9, No.12 pp.419~430
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2020.9.12.419

Double Encoder-Decoder Model for Improving the Accuracy of the Electricity Consumption Prediction in Manufacturing

Yeongchang Cho† · Byung Gill Go†† · Jong Hoon Sung††† · Yeong Sik Cho††††

ABSTRACT

This paper investigated methods to improve the forecasting accuracy of the electricity consumption prediction model. Currently, the demand for electricity has continuously been rising more than ever. Since the industrial sector uses more electricity than any other sectors, the importance of a more precise forecasting model for manufacturing sites has been highlighted to lower the excess energy production. We propose a double encoder-decoder model, which uses two separate encoders and one decoder, in order to adapt both long-term and short-term data for better forecasts. We evaluated our proposed model on our electricity power consumption dataset, which was collected in a manufacturing site of Sehong from January 1st, 2019 to June 30th, 2019 with 1 minute time interval. From the experiment, the double encoder-decoder model marked about 10% reduction in mean absolute error percentage compared to a conventional encoder-decoder model. This result indicates that the proposed model forecasts electricity consumption more accurately on manufacturing sites compared to an encoder-decoder model.

Keywords : Time-Series Forecasting, Deep Learning, Machine Learning

제조업 전력량 예측 정확성 향상을 위한 Double Encoder-Decoder 모델

조 영 창† · 고 병 길†† · 성 종 훈††† · 조 영 식††††

요 약

본 연구는 기존 전력량 예측 모델의 구조를 변경하여 모델의 예측 능력을 향상 시킬 수 있는 방법에 관하여 연구하였다. 전기에 대한 수요는 그 어느 때보다 증가하고 있다. 산업 부문에서는 그 어느 부문 보다 전기 소모량이 많음으로, 더욱 정확한 공장 지역의 전력량 소모 예측 모델이 잉여 에너지 생산을 줄이기 위해 주목을 받고 있다. 우리는 2개의 개별 encoder와 한개의 decoder를 사용하여, 장기와 단기 데이터를 모두 사용하는 double encoder-decoder 모델을 제안한다. 우리는 제안된 모델을 세홍(주)의 생산 구역에서 2019년 1월 1일부터 2019년 6월 30일 까지 모집된 전력 소모량 데이터에서 평가 하였다. double encoder-decoder 모델은 기존의 encoder-decoder 모델을 사용했을 때와 비교하여 약 10 %의 평균 절대 비율 오차의 감소를 기록 하였다. 본 결과는 제안한 모델이 encoder-decoder 모델에 비해 생산 지역의 전력 사용량의 예측을 더 정확하게 하는 모델임을 보여준다.

키워드 : 시계열 예측, 딥러닝, 머신러닝

1. Introduction

Explosive growth in the population caused ever-increased usage of electricity consumption. Considering that large proportions of primary energy supply still rely on fossil fuels such as coal, petroleum, and natural gas [1], increased demands on energy supply raise tremendous amounts of problems on environmental pollution. The demand for electricity has increased significantly during the last 10 years [1], and therefore the accurate forecasting methods on electricity consumption become more important than ever before for the efficient planning of energy production, which may facilitate the deduction of the wasted energy.

※본 연구는 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구과제임(No.20172010000260).
† 비 회 원 : (주)에스더블유엠 부설연구소 연구원
†† 준 회 원 : (주)에스더블유엠 부설연구소 전임연구원
††† 정 회 원 : (주)에스더블유엠 부설연구소 이사
†††† 비 회 원 : (주)AMEP 기술연구소장
Manuscript Received : August 21, 2020
First Revision : September 22, 2020
Accepted : October 14, 2020
* Corresponding Author : Byung Gill Go(byunggill.go@swm.ai)

Currently, the vast majority of research papers focus on energy demand forecasting on the smart grid system [2,4,5]. They forecast overall electricity consumption in a certain area for the grid. While these approaches account previous patterns on overall electricity consumption, they do not consider individual patterns that individual factories or households possess. Since 61.7% of final energy consumptions are from the industrial sector [1], it would be beneficial for the efficient usage of electrical energy throughout the flexible energy storage system or demand response system [3], if precise forecasting of individual factories' electricity consumptions is possible.

Electricity demand forecasting belongs to the time-series forecasting problem. Traditional forecasting methods, such as Box-Jenkins Autoregressive Integrated Moving Average (ARIMA) [6] and state-space models, were often applied to electricity load forecasting problem [7], but due to the intrinsic features of electricity consumption datasets, it often failed to provide accurate predictions for electricity demand. These datasets do not have stationarity nor linearity [2]. These datasets often have shifted distributions over time, which violates the assumptions of stationarity, which is required for traditional ARIMA approaches. Another significant problem in applying ARIMA on electricity forecasting is the existence of non-seasonality. The Traditional Box-Jenkins approach expects to have seasonal patterns on the dataset by fitting the ARIMA model onto the time-series dataset.

While traditional statistical models have encountered difficulties on a dataset which may not be suitable for their model assumptions, machine learning approaches, particularly Deep Neural Networks (DNN), have started showing their adequateness on varieties of tasks [8-12] with the advancement in computing hardware such as graphic processing unit (GPU) and tensor processing unit (TPU) which facilitates faster matrix computation for such models. Deep learning methods achieved state-of-the-art performance on several tasks such as image classification [8,10], object detection [9], statistical machine translation [11,12,22,23], and natural language understanding [24,25], but for time-series forecasting, they have not reached the same level of maturity.

There have been several improvements in the implementation of machine learning techniques on time series forecasting in the last two decades. Before the emergence of deep learning, support vector regression (SVR) [14] and random forest (RF) [15] have been studied for time series forecasting and achieved preferable results compared to traditional models. More recently, the applicability of multilayer perceptron, which is also known as DNN, on time series forecasting has been investigated. However, DNN does not have a recursive structure, which is essential for the time-series forecast. Recurrent neural network (RNN), which is a variant of a multi-layer perceptron, has the structure of propagating signals through the time axis, which makes it more suitable for the sequential learning dataset [17]. RNN and its variants, such as long short-term memory (LSTM) [18] and Gated recurrent unit (GRU) [12], are capable of learning long-term dependencies between each observation and therefore frequently used for the time series forecasting modeling [2-5].

In this paper, we examine the effectiveness of recent advancements in deep learning approaches on electricity demand forecasting tasks. We firstly adapt the encoder-decoder model with LSTM layers as proposed in [11,12], and then make some modifications on encoder units for better forecasting. Since encoder-decoder architecture was originated from NLP tasks, some features of encoder-decoder are designated more towards NLP tasks. For instance, neural machine translation (NMT) tasks, where a translation of one language to another language is required, force the encoder to receive source sentences in one language as input and the decoder to output target sentences in another language. However, time-series forecasting task is not alligated to such requirements like one-to-one sentence mapping. Therefore we change the structure of the encoder-decoder model for time-series to have two separate encoder units, where each of them is responsible for learning different types of data. The first encoder takes relatively long-term data with a moderate time interval between measurements, and the second encoder extracts information from more finely recorded short-term data. These extracted features are then used for one decoder unit to produce a fixed-length sequence of forecasts. Secondly, we apply the attention mechanism [19,20] on the outputs from encoder units, to focus on the parts that possess more relevant information when predictions are made. The evaluation of the performance

comparison between conventional encoder-decoder and our modified model with and without attention is carried out.

The rest of the sections is constructed as follows. In section 2, we review the fundamental researches on RNN and its variants and then delineate encoder-decoder architecture which uses RNN or its variants. Furthermore, we review the ML application on the electricity demand forecasting task. In section 3, our proposed model, double encoder-decoder, is discussed in detail with additional explanations on the attention mechanism. Throughout section 4, our experimental setups and results are presented to evaluate our proposed approach compared to the standard encoder-decoder model, and then we make conclusions in section 5.

2. Related Work

In this section, we describe RNN and LSTM models and then delineate the encoder-decoder model, which uses RNN or LSTM cells in the computation. After that, we review researches specific to electricity load forecasting tasks both with ML approaches.

2.1 Background: Recurrent Neural Network and LSTM

Since the 1950s, neural network [19] has been studied for artificial intelligence, motivated from neurons in an actual brain. After its first proposal, the neural network was applied to various tasks but was not popularized as much as now, due to the tremendous amount of required computation. However, as the computing power of hardware increased, with distinct advantageous characteristics of neural networks, such as universal function approximator theorem [20] or representation learning [21], neural networks gained popularity again recently.

The conventional feed-forward neural network receives an input data vector and predicts a corresponding output based on the following equation.

$$\hat{y}_i = f(x_i;\theta) \tag{1}$$

where f is an approximation function that maps input to output. $\hat{y}_i$ is a prediction by f given x_i with parameters θ. When an approximation function f is chosen as neural networks, f can be decomposed

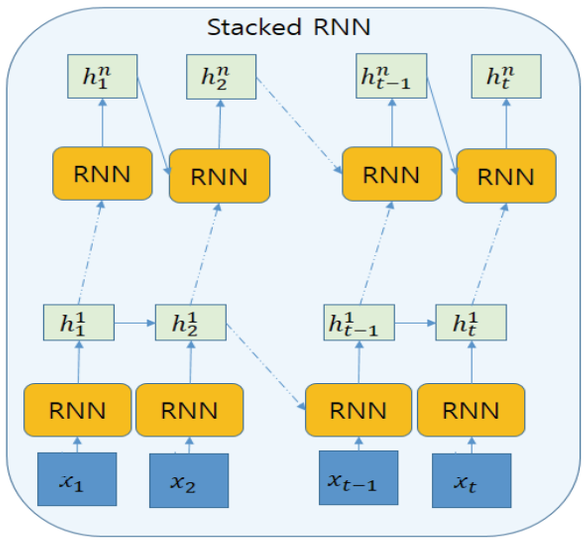

Fig. 1. Stacked RNN Computation Graph. Each Arrow Represents the Computational Flow. Dashed Arrows Represent Skipped Computation in the Graph. Inputs are used in RNN Computation to Produce hidden States. Superscript in hidden States Notation Represent the Order of the Layer, and Subscript Represents the Time Step

into two separate parts, one for the computation of hidden states and the other for the computation of the output as following equations:

$$h_i = \sigma(W^T x_i + b) \tag{2}$$

$$\hat{y}_i = I(h_i) \tag{3}$$

where h_i is a hidden state, and W is a weight vector. I is a function that maps hidden states in the last layer to output. Its choice is mostly dependent on the task. σ can be any nonlinear function. Common choices for neural networks are sigmoid:

$$f(x) = \frac{1}{1+e^{-x}} \tag{4}$$

or rectified linear unit (ReLU):

$$f(x) = \max(0,x) \tag{5}$$

One of the most important assumptions on the neural network and general regression is the independence of each observation in the dataset. For instances, $x_i \in \boldsymbol{X}$ for $i=1,\ldots,n$ is i.i.d. But this assumption is often violated in sequence learning since the observations are correlated. This is referred to as autocorrelation between

observations. In order to acclimate autocorrelated observations, the recursive structure of the learning algorithm, such as RNN [17], is typically required.

RNN propagates hidden states computed from previous time steps to hidden states in the current time step with hidden-to-hidden weight vector, W_{hh}. By adopting this additional structure, the computation of hidden states can be modified as follows:

$$h_i = \sigma(W_{ih}^T x_i + W_{hh}^T h_{i-1}) \tag{6}$$

where W_{ih} represents the input-to-hidden weight vector. As shown in Fig. 1, RNN and its variants are often stacked [28] on each other. This architecture makes a deep model by connecting each RNN layer's hidden states on top of each other. This enables RNN to learn more sophisticated representations of a dataset.

However, due to its recursive computation, propagated gradients are easily exploded or vanished. This problem yields to the modification of the architecture. In [18], LSTM was proposed to learn long-term dependencies among sequential datasets. LSTM computes hidden states and cell states using cell and three gates called input, forget, and output gates. Firstly, LSTM computes internal vectors using a current input x_i and a previous hidden state h_{i-1} by the following equations:

$$\begin{aligned} f_i &= \sigma(W_{if} x_i + U_{hf} h_{i-1}) \\ u_i &= \sigma(W_{iu} x_i + U_{hu} h_{i-1}) \\ o_i &= \sigma(W_{io} x_i + U_{ho} h_{i-1}) \\ \tilde{c}_i &= \sigma(W_{ic} x_i + U_{hc} h_{i-1}) \end{aligned} \tag{7}$$

where f_i, u_i, o_i, $\tilde{c}_i$ represent output vectors computed by forget, update, output and cell weight matrices. Using these 4 vectors, LSTM yields the current cell state c_i and the current hidden state h_i as follows:

$$\begin{aligned} c_i &= f_i \odot c_{i-1} + u_i \odot \tilde{c}_i \\ h_i &= o_i \odot \sigma(c_i) \end{aligned} \tag{8}$$

Operation $\odot$denotes an element-wise product between vectors or matrix. By using these gated operations, LSTM is enabled to control the past information flows to the current one by manipulating propagated gradients.

2.2 LSTM-RNN-based Encoder-Decoder

With the popularity of RNN in a recent decade, many researchers proposed improvements in model architectures for various tasks. Encoder-decoder [11, 12] is one of the proposed architectures to overcome bottlenecks of RNN in sequence-to-sequence tasks, such as semantic learning and variable-size inputs.

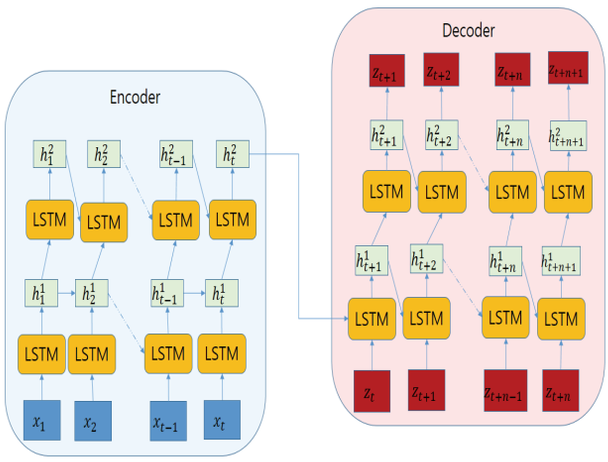

Fig. 2. Flow Diagram of Encoder-decoder Model, Blue Squares Represent Inputs to the Encoder, and Red Squares Represent the Output of the Decoder. With the Stacked RNN Computation, Inputs to the Encoder are Transformed to Hidden States, which is then used as Initial Hidden State of the Decoder. Decoder Forecasts Current Output Given Previous Hidden States and Previous Output Value

In the encoder-decoder structure, the encoder unit is responsible for compressing relevant information from the input sequence to predict output sequences. Instead of bypassing current input values to current hidden states for the output, by limiting the length of the vector to the decoder, the encoder learns the most informative part of input sequences in order to predict the output, even if the most relevant input is far from current time step. This is analogous to seasonal ARIMA, which incorporates distanced autocorrelation between inputs and outputs.

As shown in Fig. 2, the encoder outputs hidden states of the last layer as a context vector [11,12] with some transformation function, and these vectors are used in the decoder. The common model architecture of encoder units is stacked RNN or stacked LSTM as represented in Fig. 1, and the context vector is usually computed as follows:

$$c = q(\boldsymbol{h}_{1:T}) \tag{9}$$

where q is a transformation function that maps $\boldsymbol{h}_{1:T}$,

a set of hidden states in case of RNN or a set of hidden states and cell states in case of LSTM, to a fixed-length vector, c. In [12], the context vector was computed as:

$$q(\boldsymbol{h}_{1:T}) = h_T \tag{10}$$

where h_T is the last time step element of $\boldsymbol{h}_{1:T}$.

The decoder unit uses the context vector c, either as input [11] or initial states of the networks [12]. The same model architecture as the encoder unit is mostly chosen. With the context vector from the encoder unit, the decoder unit also uses hidden states of previous time steps of the decoder unit as conventional RNN or LSTM:

$$h_t^{dec} = g(y_{t+1}, c, h_{t-1}^{dec}) \tag{11}$$

where g is the decoder unit that forecasts outputs given previous hidden states and context vectors. We specially denote hidden states or hidden states and cell states in the decoder as, h_t^{dec}, to distinguish it from ones from the encoder. This overall architecture makes encoder-decoder differentiable and enables continuous learning from standard backpropagation algorithms [29,30].

While the adaptation of encoder-decoder architecture advanced sequence-to-sequence forecasting, it still suffered to the prediction of longer sequences, where the location of the semantic token in input and output sequence was often not aligned together. The proposed new mechanism called attention [22, 23] gives weight to the context vector from the encoder to facilitate better matching between input and output sequences.

2.3 ML Applications on Electricity Consumption Forecasting

As efficient usage of electricity becomes a major concern of the wide public, there have been numerous attempts on electricity load forecasting with machine learning approaches. Perhaps the two most investigated approaches in machine learning were random forest [15] and support vector regression [14]. Random Forest grows individual trees from a bootstrapped dataset, and average the outcome of an individual. With this procedure, the random forest can achieve more accurate predictions by lower the variance of outcome [26], which is a problematic feature of the decision tree. In [15], random forest achieved lower mean absolute percentage error (MAPE) compared to exponential smoothing and ARIMA model and compatible result to the artificial neural network (ANN). Support vector regression [27], which is a variant of support vector machine, computes the weight vector from the data points above the pre-defined threshold. This procedure makes SVR more robust to outliers and easily learns non-linearity in a dataset. With some slight modifications, in [14], locally weighted SVR outperformed over locally weighted regression on MAPE loss by 24.72% in forecasting maximum daily load.

More recently, RNNs have been applied to electricity load forecasting tasks. Authors of [5] used LSTM for long-term electricity load forecasting, which was beneficial to utility suppliers. They trained the LSTM model using 10 years of data and evaluated the forecasting ability on 5 years of data by MAPE loss. Over the 5 years, MAPE was 6.54%, which showed that forecasts from LSTM could be a good indicator of future electricity load even in long periods. More similar to our tasks, in [2,3], analysis on the applicability of LSTM on short-term forecasting of electricity usage in the residential sector and smart-grid was carried out. In [2], LSTM was used to calculate a short-term load for the smart grid. Their results showed that LSTM achieved far better accuracy compared to both conventional machine learning approaches, such as neural network, SVR, and traditional time series forecasting methods like seasonal ARIMA. In contrast to [2], where the study was performed from the perspective of overall electricity usage in a certain area, the study of electricity consumption of individual households was conducted in [3]. Empirical mean, k-nearest neighbor, neural networks, and LSTM were benchmarked, and LSTM yielded the best accuracy in terms of MAPE. One of the interesting results from [3] was that forecasting the aggregated electricity usage yielded higher MAPE than summing results from forecasting individuals' usage. This result advocates the need for learning individual electricity consumption patterns, which aligns with our study.

3. Methodology

In this section, we firstly describe the architecture of our proposed model, double encoder-decoder (DED), which is composed of two encoders and one decoder. Our proposed model is expected to incorporate with both long term and short term measurements in order to predict accurate forecasts based on both long-term trend and short-term momentum of the dataset. Secondly, we apply the attention mechanism [22,23] on our proposed model to align two context vectors from encoders for decoder by assigning appropriate weights.

3.1 LSTM-based Double Encoder-Decoder Model

As shown in Fig. 3, our proposed model consists of two encoders and one decoder. One encoder is for long-term effects, and the other encoder is for short-term effects. Let enc^{long} denote encoder for long-term effects and enc^{short} denote encoder for short-term effects. The role of both encoders is to produce a context vector, c^{long} for the long-term and c^{short} for the short-term, which is used as initial hidden states in the decoder. Context vectors are computed as follows:

$$h_t^{long} = enc^{long}(x_t, h_{t-i}^{long}) \\ c^{long} = h_T^{long} = q(\boldsymbol{h_{1:T}^{long}}) \tag{12}$$

$$h_t^{short} = enc^{short}(x_t, h_{t-1}^{short}) \\ c^{short} = h_T^{short} = q(\boldsymbol{h_{1:T}^{short}}) \tag{13}$$

where h_t^{long} denotes a hidden state for the long- term encoder, enc^{long}, and h_t^{short} denotes for short- term. $\boldsymbol{h_{1:T}^{long}}$ and $\boldsymbol{h_{1:T}^{short}}$ represent a set of hidden states in every time step, where measurements take. i indicates the time lag between measurements in the input sequence. Our enc^{long} receives input sequence which is measured with the time interval of i units between each measurement. In contrast, our enc^{short} takes data, which is measured in 1 unit time interval. Context vectors for our first proposed model are chosen to be the last hidden state as shown in equation (10).

Even though a input dataset for long-term data is measured more sparsely, we used the same number of measurements on long-term and short-term encoder for our model specification. However, since the time intervals of measurement for the enc^{long} is greater than enc^{short}, a longer period of time can be captured by enc^{long}. For instance, when the time interval i equals 4, if the short-term encoder receives input of 120s, the long-term encoder receives input of 480s.

The decoder unit receives both c^{long} and c^{short} and

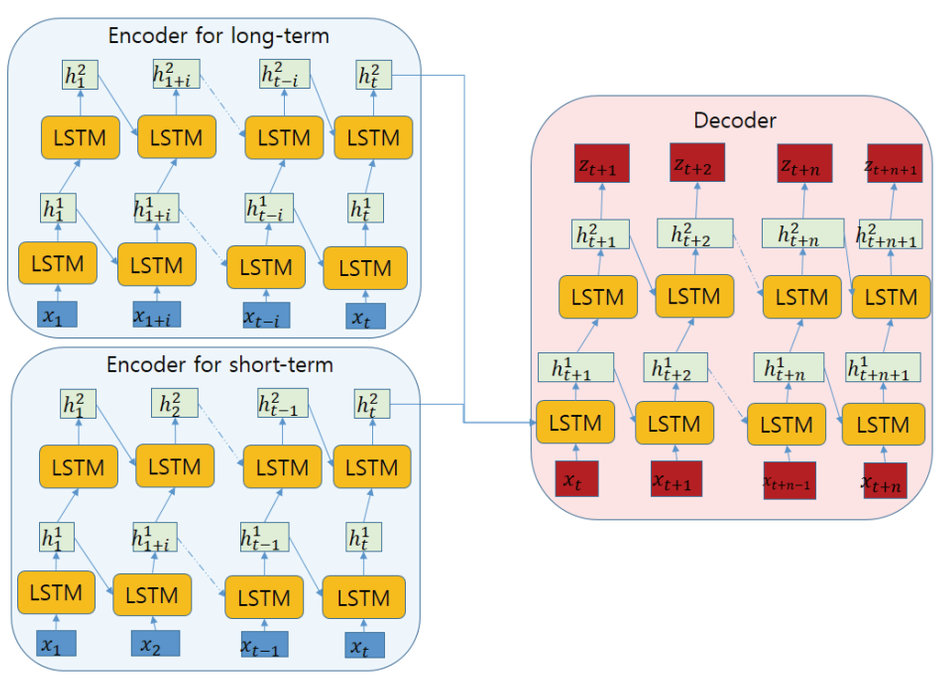

Fig. 3. Computational Graph of Double Encoder-decoder. It is the Same as Fig. 2, Except that it has One Additional Encoder Unit. The Last Hidden States from both Encoders are used as Initial Hidden States in the Decoder after Concatenation

uses as initial hidden states after concatenation. By using these initial hidden states and decoder input vector at the previous time step t-1. The sequence of output can be computed as follows:

$$\hat{y_t} = dec(c^{long}, c^{short}, y_{t-1}) \quad (14)$$

For instance, with the RNN structure, computation of hidden states in the decoder, equation (11), can be re-expressed as follows:

$$h_t^{dec} = \sigma(W_{ih} y_t + W_{hh} h_{t-1}^{dec}) \quad (15)$$

where z_t denotes the decoder input vector at the time step t.

3.2 Attention Mechanism-based DED Model

Rather than using the last hidden states from encoder units as context vectors, we may change the operation conducted by function q in equation (10). The attention mechanism produces a weighted sum of the sequence of hidden states, where the weight is assigned based on the alignment between input and output sequence. Here, we briefly describe the attention mechanism applied to our proposed model. A more detailed description is referred in [22,23].

Instead of using the last hidden state as the context vector, we replace equation (10) as follow:

$$c_i = q(\boldsymbol{h}) = a_i \boldsymbol{h} \quad (16)$$

where

$$\boldsymbol{h} = [\boldsymbol{h}_{1:T}^{long}; \boldsymbol{h}_{1:T}^{short}],$$
$$a_i = [a_{i1}, \ldots, a_T]$$

It should be noted that we concatenate hidden states of the last encoder layer from enc^{long} and enc^{short}, which is denoted as $[\boldsymbol{h}_{1:T}^{long}; \boldsymbol{h}_{1:T}^{short}]$ in equation (16). a_{ij} refers to a weight for the hidden state at jth time step in the encoder for ith time step hidden states in the decoder. Weight a_{ij} is computed by

$$a_{ij} = \frac{\exp(s_{ij})}{\sum_{k=1}^{T}} \quad (17)$$

where s_{ij} represents the score in between h_i^{dec} and, h_j, jth element in $\boldsymbol{h}$. If h_j is important when predict the y_i, high score would be assigned, which results in greater weight for that element. There are several functions for scoring. Frequently used functions [11] are:

$$s_{ij} = \begin{cases} h_i^{dec} h_j & \text{dot} \\ \mathrm{h_i^{dec}} \mathbf{W_a} \mathrm{h_j} & \text{general} \\ \mathrm{v_a tanh}(\boldsymbol{W_a}[h_i^{dec}; h_j]) & concat \end{cases} \quad (18)$$

By conditioning on context vector based attention, the identically structured decoder unit can be used to produce a forecast in our model.

4. Experiment and Results

We examined our proposed approaches to our factory electricity consumption data set. In this section, we first outline how our data set was collected and used for our model training. After that, we briefly mention the tuning and scoring procedure for model comparison. Secondly, in the following subsection, we report our proposed models' results with a comparison to result from the baseline model and then discuss the meaning of our results.

4.1 Dataset and Experimental Setup

Our Sehong data set is a recording of electricity usage of a factory for 6 months from January 2019 to September 2019 in South Korea. Each measurement was firstly recorded with a minute time interval, separately on every electrical socket used in every room, and then summed in a group to show electricity usage for each individual room. The unit of measurements for electricity consumption was in kWh, with 3 decimal places. In the factory, there were 12 different rooms. Among 12 separate rooms, 11 of them were office area, and only one room was for manufacturing, which was the main source of electricity consumption. In addition to that, manufacturing machines' internal state measurements such as air compression rate, and external factors like temperatures were also acquired. Measuring frequencies of external variables were sometimes set less densely. These measurements were interpolated using a simple mean when it was necessary. The number of sample points, measuring units, and descriptions of individual variables are represented in Table. 1. As shown in Table. 1, the

Table 1. Descriptions of Sehong Dataset

Variable Names	Sample Points			Units	Independent/ Dependent	Description
	Train	Validation	Test			
Manufacturing Room	138500	23090	69250	kWh	Independent	Electricity consumption measured in manufacturing Room
staff lounge				kWh	Dependent	Electricity consumption measured in staff lounge
Quality Control Dept.				kWh		Electricity consumption measured in quality control department
R&D Lab.				kWh		Electricity consumption measured in R&D laboratory
Marketing Support Dept.				kWh		Electricity consumption measured in marketing support department
Exhibit Hall				kWh		Electricity consumption measured in exhibit hall
Business Development Dept.				kWh		Electricity consumption measured in business development department
Marketing Dept.				kWh		Electricity consumption measured in marketing department
Management Planning Dept.				kWh		Electricity consumption measured in management planning department
President's office				kWh		Electricity consumption measured in president's office
Executive Suites				kWh		Electricity consumption measured in executive suites
Conference Room				kWh		Electricity consumption measured in conference room
Library				kWh		Electricity consumption measured in library
P1				kPa		Internal air compressure of the 1st manufacturing machine in the manufacturing room.
P2				kPa		Internal air compressure of the 2nd manufacturing machine in the manufacturing room.
P3				kPa		Internal air compressure of the 3rd manufacturing machine in the manufacturing room.
P4				kPa		Internal air compressure of the 4th manufacturing machine in the manufacturing room.
Temperature	23270	3880	11638	°C		Temperature of Suwon, South Korea, where Sehong is located.
Humidity				%		Relative humidity of Suwon, South Korea, where Sehong is located

dataset was split into train, validation, test dataset. The proportion between train, validation, and test was set approximately as 0.6, 0.1, and 0.3.

Since our variables had different scales and units, we needed to transform our data set to prevent typical gradient vanishing problem in LSTM-RNN models. We computed mean on individual variables with variance, and then applied standardization with the following equation:

$$\bar{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{19}$$

where $\bar{x}_{ij}$ is ith observation of jth variable after standardization, and μ_j and σ_j represent the mean and standard deviation of jth variable in the input.

We used the electricity usage in the manufacturing room as a target variable, since our main object of the research was to forecast electricity usage on the manufacturing site based on previous historical observations. As shown in Table. 1, electricity usage in other rooms in the same building were also used as dependent variables. In addition to that, manufacturing machines' internal state measurements such as air compression rate, and external factors like temperatures were also used as dependent variables. These features were selected by exploring correlation coefficients [36] and principal component analysis [37]. We used the measurements of selected dependent variables and target variable that were picked in every 20 minutes as long-term encoder's inputs, and corresponding measurements that were collected in every 1 minute were fed into the short-term encoder, enc^{short}. The number of measurements for both long-term and short-term encoder was chosen to be the same, so we used the last 10 hours of data for enc^{long} and 30

minutes of data for enc^{short}. Target observed value of the previous time step was used for decoder input during training. Our models outputted 3 hours ahead target values with time steps of 1 minute.

We trained all our models with LSTM cells using the train dataset. When training the model, several hyperparameters needed to be tuned. For the hidden unit size of each LSTM layer, we examined the hidden unit size between 64, 128, and 256 units, with two layers for each encoder and decoder unit. In order to optimize the parameters in the LSTM layers, we chose the ADAM optimizer [29] with the learning rate exponentially decaying from 0.0001. Hyperparameters were tuned using the validation dataset.

In order to fit models, loss functions needed to be selected along with the optimizer function. A commonly used loss function for time-series forecasting is Mean Absolute Error (MAE):

$$MAE = \sum_{i=1}^{N} \frac{|\hat{y_i} - y_i|}{N} \tag{20}$$

or Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{1}{n} \sum_{i=1}^{N} \frac{|\hat{y_i} - y_i|}{y_i} \tag{21}$$

where $\hat{y_i}$ is a prediction of true output y_i. We fitted our parameters with MAE loss, but also tracked other metric, MAPE, for convenient interpretation between models.

Two parallel RTX 2080 GPUs with 8 GB memory and 6123 GFLOPS per each under Ubuntu 16.04 OS were used for both training and testing of models with python 3.6 and Tensorflow 2.2 [35].

4.2 Performance Evaluation for Double Encoder-Decoder

The comparison of prediction accuracy between our proposed models and baseline models is shown in Table. 2 with MAE and MAPE scores. Numbers after model names represent the hidden unit size used for LSTM cells when the model was constructed. Bold numbers show the best results among the same models. The unit of MAE is the same as the original electricity measurements, kWh, after standardization, and the unit of MAPE is in percentage. Our results were calculated using only the test dataset. For the baseline model, we used the conventional encoder-decoder model [11,12] as a baseline model.

Table 2. Comparison between Double Encoder-decoder and Double Encoder-decoder with Attention

	MAE	MAPE
Encoder-Decoder-64	0.4522	54.30
Encoder-Decoder-128	0.4357	50.47
Encoder-Decoder-256	0.4348	53.50
Double Encoder-Decoder-64	0.4645	61.23
Double Encoder-Decoder-128	0.4233	45.15
Double Encoder-Decoder-256	0.4635	67.41
Double Encoder-Decoder with Attention-64	0.4358	50.28
Double Encoder-Decoder with Attention-128	0.4752	53.43
Double Encoder-Decoder with Attention-256	0.4515	52.82

The double encoder-decoder model recorded 0.4233 kWh and 45.15% in MAE and MAPE metrics with the best-tested hyperparameters. These values represent the one minute mean of absolute deviance between observed electricity consumption and its predicted values when the next 3 hours' electricity consumptions are forecasted. Similarly, MAPE represents one minute mean of absolute deviance divided by observed value at each time step. Perhaps, all our reported measures in Table. 2, MAE and MAPE, are calculated on the standardized values. Therefore, both target value and predicted value need to be transformed by the inverse function of equation (19), when the interpretation of the error measures in the original scale is required.

Our DED model achieved about 10 percent reduction on percentage error compared to the baseline model's MAPE, which scored 50.47% in their best hyperparameter settings. Also, DED scored lower MAE compared to the baseline model's MAE, 0.4348 kWh. This is an indication of the increment of the forecasting power of the proposed DED model in contrast to the encoder-decoder model with one encoder.

On the other hands, the DED model with attention mechanism scored 50.28 % and 0.4358 kWh in MAPE and MAE under the best hyperparameter setting. These are insignificantly different from the baseline model, and produced worse test results contrary to

the DED model without attention. This result demonstrates that the attention mechanism does not bring improvements on the predictive performance of the encoder-decoder model, and causing degeneration of the forecasting power DED model.

5. Conclusion

Forecasting electricity consumption is a challenging time-series regression task, but many traditional statistical models do not adequately fit the datasets due to the violation of model assumptions. Many ML and DNN approaches have been investigated on the electricity consumption forecasting task, but their results are not as satisfactory as in other domains. Their recent developments were tailored for NLP tasks, and therefore, modifications to the model structures are required to adapt to time-series forecasting tasks. We proposed the double encoder-decoder model, which uses two encoders, instead of one, in order to fit two different types of data, sparsely and finely measured data. We observed that the double encoder-decoder model brought the reduction in several loss metrics compared to the standard encoder-decoder in our Sehong dataset. This result would demonstrate that an improvement in forecasting power can be achieved by ensembling results from two encoders. Moreover, we tested the effect of attention mechanisms on our double encoder-decoder, model but did not observe any significant advancements. Rather, the accuracies of the model are degraded on the test dataset. Recently, many researchers [33,34] proposed a modification of attention for the application to time-series forecasting. Some of these researches pointed out that attention mechanisms on time-series forecasting are not as effective as on NLP tasks. Therefore, in future studies, further investigations on whether attention could bring meaningful effects on the double encoder-decoder model with recently proposed modified attention mechanisms.

References

[1] Y. Cho, Energy info. Korea. Kyonggi-do, Korea: Korea Energy Economics Institute, 2018.

[2] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network," *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, 2017.

[3] 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 Transactions on Smart Grid*, Vol.10, No.1, pp.841-851, 2019.

[4] J. Bedi and D. Toshniwal, "Deep learning framework to forecast electricity demand," *Applied Energy*, Vol.238, pp. 1312-1326, 2019.

[5] R. K. Agrawal, F. Muchahary, and M. M. Tripathi, "Long term load forecasting with hourly predictions based on long-short-term-memory networks," *2018 IEEE Texas Power and Energy Conference (TPEC)*, 2018.

[6] G. E. P. Box and G. M. Jenkins, Time series analysis: forecasting and control. Oakland: Holden-Day, 1976.

[7] E. Erdogdu, "Electricity demand analysis using cointegration and ARIMA modelling: A case study of Turkey," *Energy Policy*, Vol.35, No.2, pp.1129-1146, 2007.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le, "Sequence to sequence learning with neural networks," In *Advances In Neural Information Processing Systems*, pp.3104-3112. 2014.

[12] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Cambridge, MA: MIT Press, 2017.

[14] E. E. Elattar, J. Goulermas, and Q. H. Wu, "Electric Load Forecasting Based on Locally Weighted Support Vector Regression," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol.40, No. 4, pp.438-447, 2010.

[15] G. Dudek, "Short-Term Load Forecasting Using Random Forests," *Advances in Intelligent Systems and Computing Intelligent Systems 2014*, pp.821-828, 2015.

[16] T. He, Z. Dong, K. Meng, H. Wang, and Y. Oh, "Accelerating Multi-layer Perceptron based short term demand forecasting using Graphics Processing Units," *2009 Transmission & Distribution Conference & Exposition: Asia and Pacific*, 2009.

[17] A. Graves, Supervised sequence labelling with recurrent neural networks. Berlin: Springer, 2012.

[18] Hochreiter, Sepp, and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation,* Vol.9, No.8, pp.1735-1780, 1997.

[19] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, Vol.65, No.6, pp.386-408, 1958.

[20] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks,* Vol.6, No.6, pp.861-867, 1993.

[21] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.35, No.8, pp.1798-1828, 2013.

[22] T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," International Conference on Learning Representations, 2015.

[24] J. Cheng, L. Dong, and M. Lapata, "Long Short-Term Memory-Networks for Machine Reading," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[25] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A Decomposable Attention Model for Natural Language Inference," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[26] L. Breiman, "Random Forests," *Machine learning*, Vol.45, No. 1, pp. 5-32, 2001.

[27] Drucker, Harris, Christopher JC Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik, "Support vector regression machines," In *Advances in Neural Information Processing Systems*, pp. 155-161. 1997.

[28] El Hihi, Salah, and Yoshua Bengio, "Hierarchical recurrent neural networks for long-term dependencies," In *Advances in Neural Information Processing Systems*, pp.493-499. 1996.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *3rd International Conference for Learning Representations*, 2015.

[30] L. Bottou, "On-line Learning and Stochastic Approximations," *On-Line Learning in Neural Networks*, pp. 9-42, 1999.

[31] T. Hastie, J. Friedman, and R. Tisbshirani, The Elements of statistical learning: data mining, inference, and prediction. New York: Springer, 2017.

[32] D. Barber, Bayesian reasoning and machine learning. Cambridge: Cambridge University Press, 2018.

[33] S.-Y. Shih, F.-K. Sun, and H.-Y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, Vol.108, No.8-9, pp.1421-1441, 2019.

[34] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

[35] M. Abadi, "TensorFlow: learning functions at scale," *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming - ICFP 2016*, 2016.

[36] Hall, Mark A. "Correlation-based Feature Selection for Machine Learning," PhD diss., The University of Waikato, 1999.

[37] Song, Fengxi, Zhongwei Guo, and Dayong Mei. "Feature selection using principal component analysis," In *2010 International Conference on System Science, Engineering Design and Manufacturing Informatization*, Vol.1, pp.27-30. IEEE, 2010.

Yeongchang Cho
https://orcid.org/0000-0001-8546-1508
e-mail : yeongchang.cho@swm.ai

He received a BSc degree in Statistics from University College London in 2019. He has been a research engineer at SWM from 2019. His research interests are in the area of Time-series Forecasting, Machine Learning, Reinforcement Learning.

Byung Gill Go

https://orcid.org/0000-0002-5520-468X
e-mail : byunggill.go@swm.ai
He received a master's degree in Computer Engineering from Kyung Hee Univ. in 2014. He has been a Associate research engineer at SWM from 2019. His research interests are in the area of Machine Learning and AI.

Jong Hoon Sung

https://orcid.org/0000-0003-0586-7116
e-mail : jhsung@swm.ai
He received a master's degree in Electrical and Computer Engineering from Ajou Univ. in 1998. He has been a director at SWM from 2015. His research interests are in the area of Machine Learning and AI.

Yeong Sik Cho

https://orcid.org/0000-0002-0622-9493
e-mail : pobicho@amep.co.kr
He received a master's degree in electrical engineering from Myongji Univ. in 2016. He has been a research- director at Amep from 2016. His research interests are in the area of Big Data, Energy Saving, AI.