ARCHIVES OF ELECTRICAL ENGINEERING VOL. 70(1), pp. 25–36 (2021)

DOI 10.24425/aee.2021.136050

Grid Search of Convolutional Neural Network model in the case of load forecasting

THANH NGOC TRAN

Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City
12 Nguyen Van Bao, Ward 4, Go Vap District, Ho Chi Minh City, Vietnam
e-mail: tranthanhngoc@iuh.edu.vn

(Received: 14.03.2020, revised: 23.07.2020)

Abstract: The Convolutional Neural Network (CNN) model is one of the most effective models for load forecasting with hyperparameters which can be used not only to determine the CNN structure and but also to train the CNN model. This paper proposes a framework for Grid Search hyperparameters of the CNN model. In a training process, the optimal models will specify conditions that satisfy requirement for minimum of accuracy scores of Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). In the testing process, these optimal models will be used to evaluate the results along with all other ones. The results indicated that the optimal models have accuracy scores near the minimum values. Load demand data of Queensland (Australia) and Ho Chi Minh City (Vietnam) were utilized to verify the accuracy and reliability of the Grid Search framework.

Key words: load forecasting, Grid Search, Convolutional Neural Network

1. Introduction

Load forecasting plays an important role in the electricity system, including the generation, transmission, distribution and retail of electricity. Depending on the period of prediction time, load forecast problems can be divided into 4 groups: very short-term, short-term, medium-term and long-term load forecasting [1–4]. Recently, many techniques and methodologies have been applied to forecast electricity load. These forecasting techniques are mainly classified into two classes: artificial intelligence methods (Support Vector Machine, Artificial Neural Networks, etc.) and statistical methods (Multiple Regression, Exponential Smoothing, ARIMA and Seasonal ARIMA, etc.) [5, 6]. Recent developments in artificial neural networks, especially Deep Learning (DL) neural networks, have been becoming one of the most active technologies in load forecasting. DL contains different models: Long Short-Term Memory Networks, Deep Belief Networks, Deep Boltzmann Machine, Convolutional Neural Network (CNN), etc. [7–9]. Among them, the CNN is a neural network using convolution operation to extract features in at least one of their layers. The architecture of the CNN consists of convolution, pooling and fully connected layers. Several hyperparameters of the CNN structure and CNN training are also used such as the number of convolutional filters, size of the kernel, pooling layer, batch size/training epochs, optimization algorithm, etc. that strongly influence the quality of the CNN model. In this regard, finding out good hyperparameters is an important procedure when using the CNN model for load forecasting. There have been numerous works devoted to optimizing the configuration hyperparameters of the CNN network including the number of filters, kernel size [10, 11] as well as the number of epochs, batch size, optimizers [12, 13]. In the present paper, the CNN Grid Search methodology is proposed with the consideration of not only hyperparameters that determine the network structure and training, but also the pre-processing input data as the number of input and differencing data. Load demand data of Queensland (Australia) and Ho Chi Minh City (Vietnam) were analyzed through training and testing processes to verify the accuracy and reliability of the Grid Search framework. The number of accuracy scores such as Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) were also used. In the training process, the optimal models with certain hyperparameters were obtained to meet the minimum requirement of the accuracy scores. In the testing process, these optimal models will be compared to all other ones to evaluate the Grid Search model. The experiments were implemented using the Keras library with TensorFlow as the backend in the Python environment with Google Colab, a free GPU on the cloud for running large-scale machine learning projects [14–18].

This paper is organized as follows. In Section 2, a brief introduction to CNN modelling along with proposing the Grid Search of 1D CNN methodology is presented. In Section 3, we focus on experiments and analysis of the obtained results. The conclusions are given in Section 4.

2. Research method

2.1. CNN for time series forecasting

Deep learning techniques have attracted a lot of attention in machine learning issues such as classification, clustering and regression. The CNN is known as one type of Deep Learning networks. As shown in Figure 1, the CNN architecture consists of convolutional, pooling and fully connected layers. In the convolutional layer, the input data is convolved with a number of filters and the feature map is created. The main function of the pooling layer is to reduce the resolution of feature maps in order to aggregate the input features. Finally, fully connected layers process output of the convolutional layers [10–13, 19, 20].

Usually, in regression and time series forecasting problems, a large amount of data is stored in the form of time series: stock indices, weather measurements, electricity load, etc. Time series is a sequence of values that depend on time and can be defined by Equation (1), where N is the number of observation values.

$$x(t) = \{x(t_1), x(t_2), \ldots, x(t_N)\}. \tag{1}$$

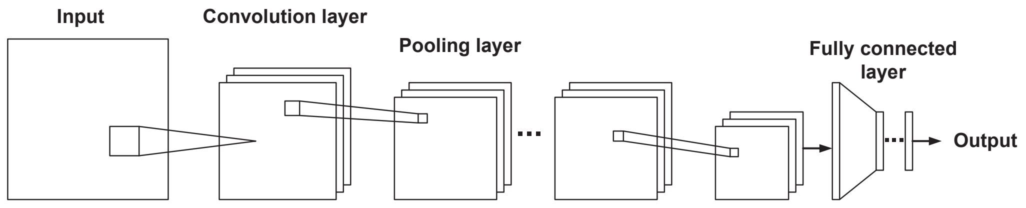
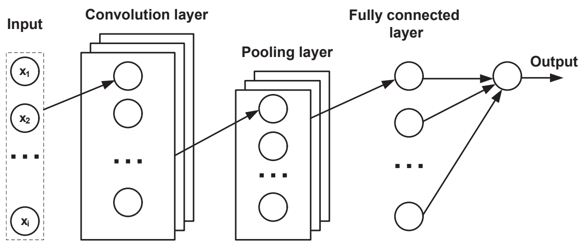


Fig. 1. The architecture of CNN

One of the important tasks in time series processing is predicting future values using past values of time series as described in Equation (2), note that i represents the number of lag observation used as input.

$$x(t_N) = f(x(t_{N-1}), x(t_{N-2}), x(t_{N-i})). \tag{2}$$

To perform the time series forecasting, a one-dimensional CNN (1D CNN) is applied. A 1D CNN is a CNN model that has a convolutional hidden layer operating over a 1D sequence. A typical 1D CNN model with input, one convolutional layer, one pooling layer and one fully connected layer is shown in Figure 2(a) [21–23]. Deep Learning libraries such as PyTorch, Keras, and TensorFlow can support well for a 1D, 2D, and 3D CNN [24]. In the paper, we used Keras, Python to implement a 1D CNN for time series prediction. The definition of the typical 1D CNN including one convolutional layer, one pooling layer and one flatten in Keras is given in Figure 2(b) as below:



(a)

Pseudocode of 1D CNN in Keras, Python

```
Model=Sequential()
model.add(Conv1D(filters =...,
kernel_size= ..., input_shape=(..., 1)))
model.add(MaxPooling1D(pool_size =  ....)

        model.add(Flatten())
model.add(Dense(1))
model.compile(loss='...', optimizer=....)
```

(b)

Fig. 2. The definition of typical 1D CNN: architecture (a); pseudocode (b)

2.2. The hyperparameters of 1D CNN

There are many factors that could influence the quality of the 1D CNN. Input parameters are such ones that define the data (features of input data, differencing of input data, etc.), or parameters of the CNN configuration (number of layers, filters, pooling, fully connected layers, etc.), as well as parameters in the training process (such as optimization algorithms, batch, epoch, etc. [10–14]. In our study, we focus on the following hyperparameters:

Number of input (i)

The number of input (i) units is pre-specified by the available data that represents the number of lag observations used as input data and defined as the dimension of input features as well. Because of the seasonality of load times series, the number of input (i) units can be chosen as the multiples of the period of time series. For instance, with the period of hourly time series of 24, the period of weekly time series is 7, the period of yearly time series is 12, and so on.

Differencing data (d)

The CNN can handle raw data with little pre-processing such as differencing the input data. The differencing of time series data can be the first differencing or the seasonal differencing as shown in Equation 3, where d is the seasonal period of time series data:

$$\begin{aligned} dy(y) &= y(t) - y(t-1), \\ dy(y) &= y(t) - y(t-d). \end{aligned} \tag{3}$$

Number of filters (f)

The first required CNN parameter is the number of filters that the convolutional layer will learn. In a CNN, a convolution filter slides over all the elements of the input taking convolution operation to extract features of the input. Commonly, the number of filters is of 32, 64, 128, etc.

Size of kernel (k)

The kernel size (k) specifies width and height of the convolution window. The kernel size must be an odd integer with the typical values of (3, 3), (5, 5), (7, 7), etc.

Number of batch (b)

The batch size is the number of samples that will be propagated through the network to update weight values. Advantages of using a batch are related to requiring less memory as well as improving the training speed of networks after each propagation.

Number of epoch (e)

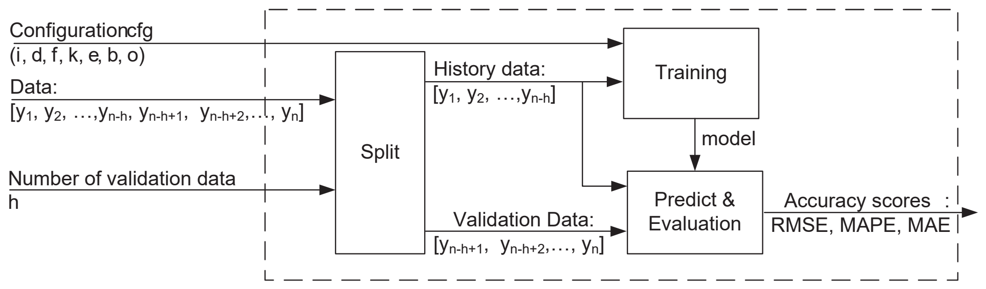
The number of epochs is the number of times that the learning algorithm will work through all training datasets. An epoch is comprised of one or more batches.

Optimization algorithms (o)

The optimization algorithm is used to iteratively update network weights based on training data. Some of optimization algorithms include SGD (Stochastic Gradient Descent), RMSProp (Root Mean Square Propagation), Adagrad (Adaptive Gradient Algorithm), Adadelta, Adam, Adamax and Nadam.

2.3. The 1D CNN methodology

Based on the 1D CNN structure and hyperparameters, the framework of 1D CNN methodology is proposed in Figure 3. Firstly, the data ($[y_1, y_2, \ldots, y_{n-h}, y_{n-h+1}, y_{n-h+2}, \ldots, y_n]$) is split into history data ($y_1, y_2, \ldots, y_{n-h}$) and validation data ($y_{n-h+1}, \ldots, y_n$) using the split function, the validation and history data have lengths of h and n-h, respectively. In the training stage, the 1D CNN model was obtained after the training process by history data and the combination of hyperparameters ***cfg*** (i, d, f, k, e, b, o). In the prediction and evaluation stage, the accuracy scores (RMSE, MAPE, MAE) between prediction and validation values are calculated. The formulas for RMSE, MAPE and MAE are shown in Equation (4) [16, 26, 27]. The pseudocode of training and prediction as well as the evaluation stage are shown in Figure 4.



Configurationcfg
(i, d, f, k, e, b, o)
Data:
[y_1, y_2, …,y_{n-h}, y_{n-h+1}, y_{n-h+2},…, y_n]
Number of validation data
h
Split
History data:
[y_1, y_2, …,y_{n-h}]
Training
model
Predict & Evaluation
Validation Data:
[y_{n-h+1}, y_{n-h+2},…, y_n]
Accuracy scores :
RMSE, MAPE, MAE

Fig. 3. The framework of 1D CNN methodology

Pseudocode of training stage

```
Input:
  – History data $[y_1, y_2, \ldots, y_{n-h}]$
  – cfg: combination of tuning hyperparameters
    $(i, d, f, k, e, b, o)$

1: Differencing data:
   If $d > 0$ :
   $History\ data = History\ data - History\ data[d]$
2: Transform data into supervised format
   Transform History data into input X_train and output
   Y train according to the number of input i
3: Define CNN model:
   $Model = Sequential(\,)$
   $model.add(Conv1D(filters = f,$
   $kernel\ size = \text{k}, input\_shape = (i, 1)))$
   $model.add(MaxPooling1D(pool\_size = 2))$
   $model.add(Flatten(\,))$
   $model.add(Dense(1))$
   $model.compile(loss = mse', optimizer = o)$
4: Training model
   $model.fit(X\_train, Y\_train, epochs = e,$
   $batch\ size = b)$

   Output: model
```

(a)

Pseudocode of predicting & evaluation stage

```
Input:
  – History data: $[y_1, y_2, \ldots, y_{n-h}]$
  – Validation data: $[y_{n-h+1}, y_{n-h+2}, \ldots, y_n]$
  – model: model from training stage

1: $t = 1$
   Repeat
   a: Difference history data and calculate offset
      If $d > 0$ :
         $History\ data = History\ data - History\ data[d]$
         $Offset = History\ data[d]$
   b: Obtain the input $X\_test$, the length of $X\_test$ is the
      number of input $i$.
      $X\_test = History\ data\ [-i\ :]$
   c: Predict the first value and then add offset
      $\hat{y}_1 = model.predict(X\_test)$
      $\hat{y}_1 = \hat{y}_1 + offset$
   d: Add actual observation to history for the next prediction
      value
      $History\ data$: $[y_1, y_2, \ldots, y_{n-h}, y_{n-h+1}]$
      $t = t + 1$
   $Until\ t = h$
2: Calculate accuracy scores
   Calculate the accuracy scores between validation val-
   ues $[y_{n-h+1},\ y_{n-h+2},\ \ldots,\ y_n]$ and prediction values
   $[\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_h]$

   Output: RMSE, MAPE, MAE
```

(b)

Fig. 4. The 1D CNN pseudocode: training stage (a); prediction and evaluation stage (b)

$$\begin{aligned}
\mathrm{RMSE} &= \sqrt{\frac{1}{h}\sum_{i=1}^{h}\left(y_{n-h+i}-\hat{y}_i\right)^2},\\
\mathrm{MAPE} &= \frac{1}{h}\sum_{i=1}^{h}\left|\frac{y_{n-h+i}-\hat{y}_i}{y_{n-h+i}}\right|,\\
\mathrm{MAE} &= \frac{1}{h}\sum_{i=1}^{h}\left|y_{n-h+i}-\hat{y}_i\right|.
\end{aligned} \tag{4}$$

2.4. The 1D CNN Grid Search methodology

Based on the tuning hyperparameters and framework of 1D CNN methodology as described above, the Grid Search model of 1D CNN methodology was established as shown in Figure 5.

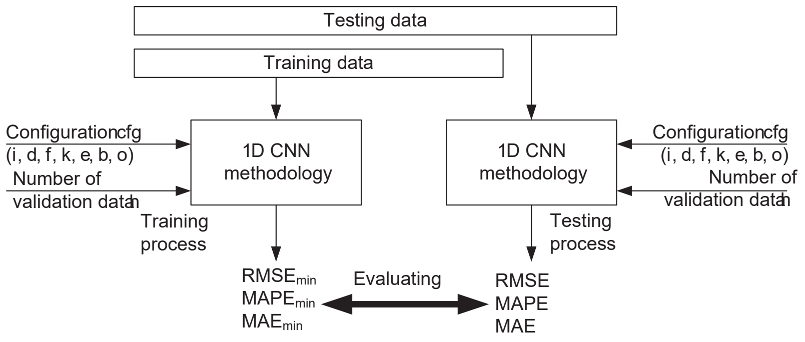


Fig. 5. The 1D CNN grid search methodology

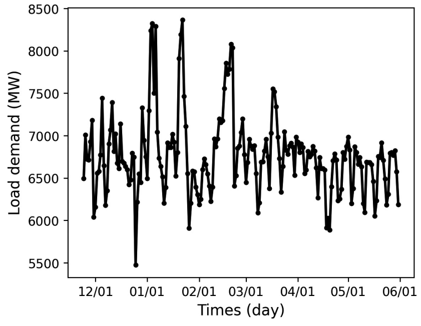
The training and testing processes have the same combination of tuning hyperparameters ***cfg*** and the same number of validation data h. For the training data, the optimal models satisfying minimum accuracy scores ($\mathrm{RMSE}_{\mathrm{min}}$, $\mathrm{MAPE}_{\mathrm{min}}$, $\mathrm{MAE}_{\mathrm{min}}$) are obtained. In the testing process, these optimal models will be compared to all other models according to their accuracy scores in order to evaluate reliability of the Grid Search model of 1D CNN methodology.

3. Experiment results and analysis

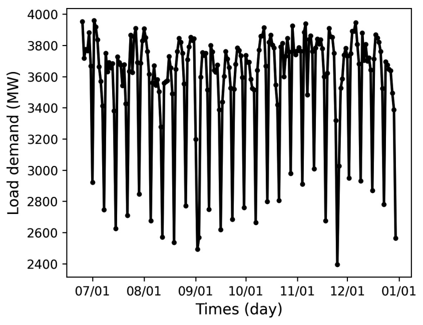
3.1. Data description and tuning hyperparameters

In order to enhance the reliability of experiment results, load demand data of Queensland (Australia) and Ho Chi Minh City (Vietnam) were studied in our experiments. The Queensland load demand dataset provides peak daily electricity demand in MW from 2013.11.24 to 2014.05.31 [25], and the Ho Chi Minh City load demand dataset – from 2018.06.25 to 2018.12.30 as presented in Figure 6.

Setup values of the tuning hyperparameters for Queensland and Ho Chi Minh City load demand data are also listed in Table 1. Because of weekly seasonality, there is a numeric value of 7 assigned to the value of input i and the value of differencing d. Combining all tuning



(a)



(b)

Fig. 6. The experimental data: Queensland load demand data (a); Ho Chi Minh City load demand data (b)

hyperparameters gives 384 cases corresponding to 384 possible models of 1D CNN methodology. The values of the number of validation data h is 7 (one week).

Table 1. The values of tuning hyperparameters

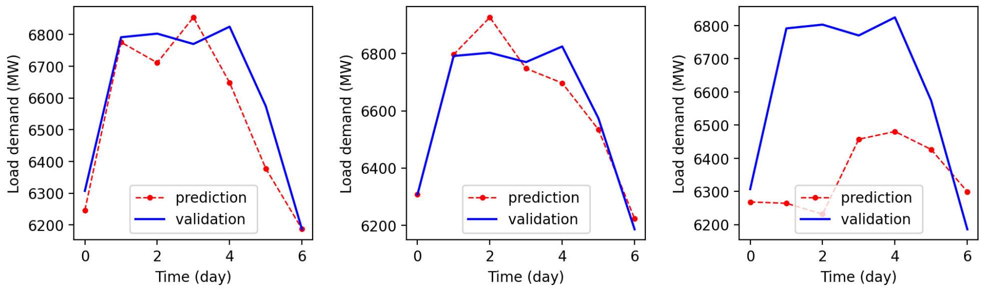
Items	Queensland	Ho Chi Minh City
input i	7, 14	7, 14
differencing d	0, 7	0, 7
number of filters f	32, 64	32, 64
kernel size k	3, 5	3, 5
number of epochs e	100, 500	100, 1 000
number of batch b	1, 100	1, 150
types of optimizer o	"RMSprop", "Adagrad", "Adadelta", "Adam", "Adamax", "Nadam"	"RMSprop", "Adagrad", "Adadelta", "Adam", "Adamax", "Nadam"
Number of possible models	384	384

3.2. The results in case of Queensland load demand data

Table 2 shows the results of the training and testing processes. For the training process, the optimal model was selected according the minimum values of accuracy scores RMSE, MAPE and MAE. Obviously, we have the same optimal models in the case of RMSE and MAPE, and another one in the case of MAE. For testing process, the column "Optimal" shows the accuracy scores for the optimal model, and the columns "Min", "Average" and "Max" – the minimum, the average and the maximum values for all possible models that can be generated. Figures 7(a), 7(b) and 7(c) give the prediction and validation series of the optimal, minimum and maximum models for the RMSE case in the testing process, respectively. Figure 8 indicates the distribution of accuracy scores for the testing process. Figure 8(a) presents the box plot for the RMSE component with the first column for the distribution of all possible models and the second column for the optimal model. The same distributed data are plotted in Figure 8(b) and 8(c).

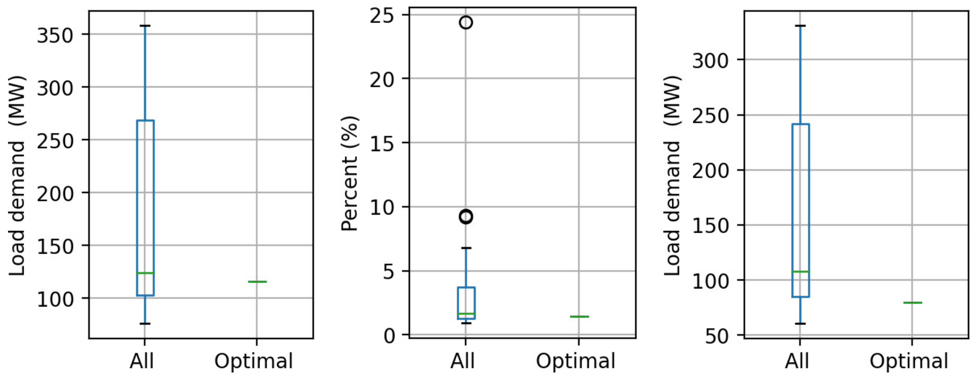
Table 2. The results of training and testing process in case of Queensland load demand data

Accuracy scores	Optimal models of training process (*i, d, f, k, e, b, o*)	Accuracy scores of testing process			
		Optimal	Min	Average	Max
RMSE (MW)	[14, 7, 32, 3, 100, 100, “Adam”]	115.60	76.22	178.08	358.25
MAPE (%)	[14, 7, 32, 3, 100, 100, “Adam”]	1.42	0.93	2.51	24.40
MAE (MW)	[14, 7, 32, 5, 100, 100, “Adamax”]	80.05	60.79	155.44	330.94



(a) (b) (c)

Fig. 7. The prediction and validation series for testing process in the case of Queensland load demand data for RMSE case: optimal model (a); minimum model (b); maximum model (c)



(a) (b) (c)

Fig. 8. The box plot of accuracy scores for testing process in the case of Queensland load demand data: RMSE (a); MAPE (b); MAE (c)

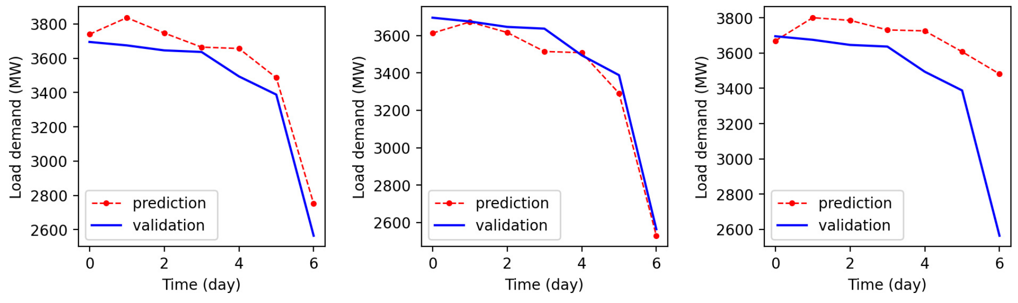
3.3. The results in the case of Ho Chi Minh City load demand data

Table 3 shows the results of training and testing processes. Obviously, the minimum of RMSE, MAPE and MAE gives the different optimal models in the training process. For the testing process, the column “Optimal” shows the accuracy scores for the optimal model, and the columns “Min”, “Average” and “Max” – the minimum, the average and the maximum values for all possible models that can be generated.

Table 3. The results of training and testing process in the case of Ho Chi Minh City load demand data

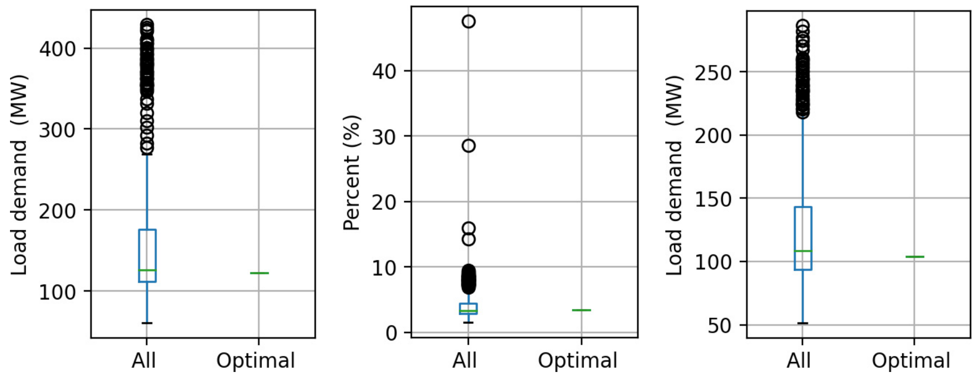
Accuracy scores	**Optimal models of training process (*i, d, f, k, e, b, o*)**	**Accuracy scores of testing process**			
		Optimal	**Min**	**Average**	**Max**
RMSE (MW)	[14, 7, 32, 5, 100, 150, “Adadelta”]	122.74	60.96	165.80	428.99
MAPE (%)	[7, 0, 64, 3, 1000, 150, “RMSprop”]	3.46	1.53	4.25	47.59
MAE (MW)	[14, 7, 32, 5, 100, 150, “Adadelta”]	103.72	51.52	126.97	286.51

Figures 9(a), 9(b) and 9(c) give the prediction and validation series of the optimal, minimum and maximum models for the RMSE case in the testing process, respectively. Figure 10 indicates the distribution of accuracy scores for the testing process. Figure 10(a) presents the box plot for the RMSE component with the first column for the distribution of all possible models and the second column for the optimal model. The same distributed data are plotted in Figures 10(b) and 10(c).



(a) (b) (c)

Fig. 9. The prediction and validation series for testing process in the case Ho Chi Minh City load demand data for RMSE case: optimal model (a); minimum model (b); maximum model (c)



(a) (b) (c)

Fig. 10. The box plot of accuracy scores for testing process in the case of Ho Chi Minh City load demand data: RMSE (a); MAPE (b); MAE (c)

3.4. Evaluation

As described above, the optimal models in the training process are determined by minimizing the accuracy scores such as RMSE, MAPE and MAE. Analysis of the results listed in Table 2 and Table 3 shows the existence of the optimal model that satisfies the minimum criteria for accuracy scores of RMSE, MAPE, and MAE. In the case of Queensland load demand data, there is a single model that satisfies all RMSE, MAPE and the other ones for MAE. In the case of Ho Chi Minh City load demand data, three different optimization models were found to meet RMSE, MAPE and MAE, respectively.

The optimal model obtained in the training process does not guarantee the best results in the testing process. Let us analyze Table 2 in the case of Queensland load demand data. When using the optimal model in the testing process, the accuracy scores of RMSE, MAPE and MAE are 115.60 MW, 1.42% and 80.05 MW, respectively. Meanwhile, there are other models that give better results with the minimum values of RMSE, MAPE, MAE of 76.22 MW, 0.93%, 60.79 MW, respectively. However, compared to the average values of all models (178.08 MW, 2.51%, 155.44 MW obtained respectively for RMSE, MAPE, MAE values) and to maximum values of the models of RMSE (358.25 MW), MAPE (24.40%), MAE (330.94 MW), the accuracy scores of the optimal model are considerably low. In addition, analyzing the boxplot of accuracy scores shown in Figure 8 makes it clear that the accuracy scores of the optimal models are close to the minimum value of all other models. Moreover, the forecast values shown in Figure 7(a) are very consistent with the validation values for the optimal model. Similar results were also obtained for the Ho Chi Minh City load demand data case. These results clearly show that the optimal model received during the training by applying 1D CNN Grid Search model will give good values in the testing process. In this regard, it is promising for applying the Grid Search model of 1D CNN methodology in any load demand times series.

4. Conclusions

Based on the 1D CNN structure and hyperparameters, a framework for the Grid Search model of the 1D CNN method has been proposed. In the training process, minimum accuracy scores of RMSE, MAPE, and MAE were applied to specify the optimal model. In the testing process, accuracy scores were used to compare the optimal model with all other ones. Both Queensland and Ho Chi Minh City load demand were used for the analysis. The results indicated the existence of an optimal model that satisfies the minimum requirement for accuracy scores. During the testing process, accuracy scores of the optimal model gave good values close to the minimum and much lower than the average. The positive results obtained in this study show an effective way to forecast load demand.

References

[1] Walther J., Spanier D., Panten N., Abele E., *Very short-term load forecasting on factory level – A machine learning approach*, Procedia CIRP, vol. 80, pp. 705–710 (2019).

[2] Aydarous A.A., Elshahed M.A., Hassan M.M.A., *Short-Term Load Forecasting Approach Based on Different Input Methods of One Variable: Conceptual and Validation Study*, 2018 Twentieth International Middle East Power Systems Conference (MEPCON), Cairo, Egypt, pp. 179–184 (2018).

[3] Raza M.Q., Khosravi A., *A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings*, Renew. Sustain. Energy Rev., vol. 50, pp. 1352–1372 (2015).

[4] Walther J., Spanier D., Panten N., Abele E., *Very short-term load forecasting on factory level – A machine learning approach*, Procedia CIRP, vol. 80, pp. 705–710 (2019).

[5] Khan S., Javaid N., Chand A., Abbasi R.A., Khan A.B.M., Faisal H.M., *Forecasting day, week and month ahead electricity load consumption of a building using empirical mode decomposition and extreme learning machine*, 2019 15th International Wireless Communications and Mobile Computing Conference (IWCMC), Tangier, Morocco, pp. 1600–1605 (2019).

[6] Joshi M., Singh R., *Short-term load forecasting approaches: A review*, International Journal of Recent Engineering Research and Development (IJRERD), no. 01, pp. 9–17 (2015).

[7] Cao Z., Member S., Wan C., Zhang Z., *Hybrid Ensemble Deep Learning for Deterministic and Probabilistic Low-voltage Load Forecasting*, IEEE Trans. Power Syst., p. 1 (2019).

[8] Yu Y., Ji T.Y., Li M.S., Wu Q.H., *Short-term Load Forecasting Using Deep Belief Network with Empirical Mode Decomposition and Local Predictor*, 2018 IEEE Power and Energy Society General Meeting (PESGM), Portland, OR, pp. 1–5 (2018).

[9] Yang J., Wang Q., *A Deep Learning Load Forecasting Method Based on Load Type Recognition*, 2018 International Conference on Machine Learning and Cybernetics (ICMLC), Chengdu, pp. 173–177 (2018).

[10] Krishnakumari K., Sivasankar E., Radhakrishnan S., *Hyperparameter tuning in convolutional neural networks for domain adaptation in sentiment classification (HTCNN-DASC)*, Soft Comput., vol. 24, no. 5, pp. 3511–3527 (2020).

[11] Subramanian S.V., Rao A.H., *Deep-learning based time series forecasting of go-around incidents in the national airspace system*, AIAA Model. Simul. Technol. Conf. 2018, no. 209959 (2018).

[12] Zahid M. *et al.*, *Electricity price and load forecasting using enhanced convolutional neural network and enhanced support vector regression in smart grids*, Electronics, vol. 8, no. 2, pp. 1–32 (2019).

[13] Nurshazlyn Mohd Aszemi, Dhanapal Durai Dominic Panneer Selvam, *Hyperparameter optimization in convolutional neural network using genetic algorithms*, Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 6, pp. 269–278 (2019).

[14] Brownlee J., *Deep Learning for Time Series Forecasting*, Ebook (2019).

[15] https://keras.io/optimizers/

[16] Brownlee J., *Deep Learning with Python*, Ebook (2019).

[17] Chen K., Chen K., Wang Q., He Z., Hu J., He J., *Short-Term Load Forecasting With Deep Residual Networks*, IEEE Transactions on Smart Grid, vol. 10, no. 4, pp. 3943–3952 (2019).

[18] Jojo Moolayil, *Learn Keras for Deep Neural Networks: A Fast-Track Approach to Modern Deep Learning with Python*, Apress (2018).

[19] Xishuang Dong, Lijun Qian, Lei Huang, *Short-term load forecasting in smart grid: A combined CNN and K-means clustering approach*, IEEE Int. Conf. Big Data Smart Comput., pp. 119–125 (2017).

[20] Dong X., Qian L., Huang L., *A CNN based bagging learning approach to short-term load forecasting in smart grid*, 2017 SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI, San Francisco, CA, 2017, pp. 1–6 (2017).

[21] Voß M., Bender-Saebelkampf C., Albayrak S., *Residential Short-Term Load Forecasting Using Convolutional Neural Networks*, 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Aalborg, 2018, pp. 1–6 (2018).

[22] Amarasinghe K., Marino D.L., Manic M., *Deep neural networks for energy load forecasting*, IEEE Int. Symp. Ind. Electron., pp. 1483–1488 (2017).

[23] Koprinska I., Wu D., Wang Z., *Convolutional Neural Networks for Energy Time Series Forecasting*, Proc. Int. Jt. Conf. Neural Networks, pp. 1–8 (2018), DOI: 10.1109/IJCNN.2018.8489399.

[24] Valentino Zocca *et al*., *Python Deep Learning*, Packt Publishing (2019).

[25] https://www.aemo.com.au/

[26] Haiqing Liu, Weijian Lin, Yuancheng Li, *Ultra-short-term wind power prediction based on copula function and bivariate EMD decomposition algorithm*, Archives of Electrical Engineering, vol. 69, no. 2, pp. 271–286 (2020).

[27] Wang Y., Ma X., Wang F., Hou X., Sun H., Zheng K., *Dynamic electric vehicles charging load allocation strategy for residential area*, Archives of Electrical Engineering, vol. 67, no. 3, pp. 641–654 (2018).