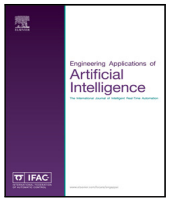
Engineering Applications of Artificial Intelligence 109 (2022) 104645

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Deep-learning-based short-term electricity load forecasting: A real case application

Ibrahim Yazici [a,1], Omer Faruk Beyca [a,1], Dursun Delen [b,c,*,1]

[a] Industrial Engineering Department, Faculty of Engineering, Istanbul Technical University, Macka, Istanbul, 34367, Turkey
[b] Department of Management Science and Information Systems, Spears School of Business, Oklahoma State University, Stillwater, OK, USA
[c] School of Business, Ibn Haldun University, Istanbul, Turkey

ARTICLE INFO

Keywords:
Data science
Time-series forecasting
Short term electricity demand prediction
Deep learning
One-dimensional CNN

ABSTRACT

The rising popularity of deep learning can largely be attributed to the big data phenomenon, the surge in the development of new and novel deep neural network architectures, and the advent of powerful computational innovations. However, the application of deep neural networks is rare for time series problems when compared to other application areas. Short-term load forecasting, a typical and difficult time series problem, is considered as the application domain in this study. One-dimensional Convolutional Neural Networks (CNNs) use is rare in time series forecasting problems when compared to Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU), and the efficiency of CNN has been rather remarkable for pattern extraction. Hence, a new method that uses one-dimensional CNNs based on Video Pixel Networks (VPNs) in this study, in which the gating mechanism of Multiplicative Units of the VPNs is modified in some sense, for short term load forecasting. Specifically, the proposed one-dimensional CNNs, LSTM and GRU variants are applied to real-world electricity load data for 1-hour-ahead and 24-hour-ahead prediction tasks which they are the main concerns for the electricity provider firms for short term load forecasting. Statistical tests were conducted to spot the significance of the performance differences in analyses for which ten ensemble predictions of each method were experimented. According to the results of the comparative analyses, the proposed one-dimensional CNN model yielded the best result in total with 2.21% mean absolute percentage error for 24-h ahead predicitions. On the other hand, not a noteworthy difference between the methods was spotted even the proposed one-dimensional CNN method yielded the best results with approximately 1% mean absolute percentage error for 1-h ahead predictions.

1. Introduction

Electricity load forecasting is of great importance to researchers, public administrators, and electricity producers and distributors (Berk et al., 2018; Cancelo et al., 2008; Djukanovic et al., 1995; Soares and Medeiros, 2008). Operational, tactical, and strategic impacts of electricity load forecasting make the issue even more critical for the power supplier firms. These impacts include operational cost, management challenges, safety issues, and lack of sustainability of power supply system in the short and mid-term (Hu et al., 2017; Kavousi-Fard et al., 2014; Osman et al., 2009; Zeng et al., 2017), and market share shrinkage and weakening competition in the electricity marketplace in the long term (Ceperic et al., 2013).

Due to transitions to deregulated electric supply markets emergence (Cancelo et al., 2008; Hooshmand et al., 2013), multifaceted contributions of forecasts to the power system's operations (Amato et al., 2020; Dordonnat et al., 2016; Sudheer and Suseelatha, 2015), the effect of handling the uncertainty in load forecasting (Santos et al., 2007), and big data surge in electricity markets that requires efficient and complex forecasting models (Hewamalage et al., 2020; Salinas et al., 2020), enhancement of smart systems use in the power supply systems (Amato et al., 2020; Wang et al., 2019), intelligent systems emergence in power supply systems (Cheng and Yu, 2019; Ozcanli et al., 2020), new forecasting method developments are needed to get more accurate results, and this, in turn, will contribute to different management levels of the power supply system. By considering these aspects, the design and development of novel methods for electricity load forecasting to reduce prediction errors and to propose new forecasting engines for businesses are the key motivations behind most of the recent research studies in this area, as is the case for the current study.

Electricity load forecasting is a typical, yet challenging, time-series forecasting problem oftentimes studied by academics and practitioners alike. In any time series forecasting, the forecasting time horizon is one

* Correspondence to: Department of Management Science and Information Systems, Spears School of Business, Oklahoma State University, USA.
E-mail addresses: iyazici@itu.edu.tr (I. Yazici), beyca@itu.edu.tr (O.F. Beyca), dursun.delen@okstate.edu, dursun.delen@ihu.edu.tr (D. Delen).
URL: http://spears.okstate.edu/delen (D. Delen).

[1] All authors have contributed collectively and equally to the original and revised version of this manuscript.

https://doi.org/10.1016/j.engappai.2021.104645
Received 25 December 2020; Received in revised form 23 July 2021; Accepted 26 December 2021
Available online xxxx
0952-1976/

of the integral parts of the analysis alongside the input–output relations, stationarity, and periodicity of load data. In the literature, forecasting horizons are classified into four categories; very short-term, short-term, mid-term, and long-term forecasting horizons (Zheng et al., 2017). The timespan for Very Short Term Load Forecasting (VSTLF) is from a few minutes up to one hour-ahead (Guan et al., 2013). It is from one hour-ahead up to one week ahead for Short Term Load Forecasting (STLF). On the other hand, periods for Medium Term (MTLF) and Long Term Load Forecasting (LTLF) are from several weeks up to several months, and from one year up to several years ahead, respectively (Din and Marnerides, 2017). In this study, STLF which is important for electricity markets, and the shareholders for spotting the hourly electricity market price changes is considered (Chen et al., 2009).

Energy market dynamics show significant differences for each country based on governmental policies and regulations. For instance, energy market establishments in Turkey have been mostly implemented through the market regulations put in place since 2001. In this market, mechanisms such as day-ahead market planning, hourly pricing, and financial consolidation were implemented in 2009, two years after these implementations, down-payment and assurance, day-ahead market consolidations, and incentives to support renewable energy programs initiated. Legislative requirements for Energy Market Enterprise Corporation (EPİAŞ) were mostly completed in 2013, and then, the Corporation was founded in order to regulate the Turkish electricity market in 2015; during that year, the intraday market was opened as well. This market regulated by EPİAŞ comprises day-ahead, intraday, and balancing market mechanisms, which are essential parts in operating a non-governmental electricity market. Day-Ahead market, intraday market and balancing market mechanisms play important roles for efficient system operation in Turkish electricity market regulated by EPİAŞ. Before EPİAŞ foundation in the market, prediction horizons for the provider firms ranging from one week ahead to several months ahead are prevalent, and these forecasting horizons were facilitating more flexibility in prediction error margins for the firms compared with the introduced regulated market. The flexibility in prediction error margins for the providers enabled the providers to compromise their excessive and/or deficient amounts in the long horizon with less price charge. After the market's shift to a regulated market, day-ahead and intraday markets, in which one-day-ahead and one-hour-ahead predictions are respectively the most essential parts of them, have played a landmark role for the electricity providers because of that the mentioned prediction error margins are narrowed. This fact has in turn increased the significance of the one-hour ahead and one-day-ahead predictions in the market. After these shifts, innovations and competition increase in the market, overestimation or underestimation of electricity load demands incurs extra costs for electricity provider firms in the market, and hence one-hour-ahead and 24-hour-ahead load predictions has emerged as the major concerns for the providers. Thus, one-hour-ahead and 24-hour-ahead load predictions are the focal point of this study as their details will be explained in the latter sections.

In general, reducing operational costs, stabilizing power supply scheduling, efficiently coordinating the load management, increasing the safety and security of the power supply systems can significantly benefit from accurate forecasting, especially from short term forecasting (Djukanovic et al., 1995; Hooshmand et al., 2013; Santos et al., 2007; Sudheer and Suseelatha, 2015; Wang et al., 2019). Technological advancements such as distributed generations (DGs), and smart device use in smart grid environments, accuracy, quick response, intelligence for STLF emerge as essential issues in forecasting (Wang et al., 2019). With the aforementioned advancements, intelligence is becoming an integral part of the power supply system management tools, such as market mechanism tools and advanced algorithms for forecasting. In addition to that, hiring a team of forecasting experts shows the importance of forecasting in the power supply system management (Duan et al., 2017). In brief, the technological advancements such as the Internet of Things, the energy Internet, Energy 5.0, Smart Grid Plus (Cheng and Yu, 2019) in electricity markets, the surge of the big data regime in this market, and the necessity of speed, accuracy of the forecasting entail more robust and intelligent forecasting techniques in the load forecasting area. In turn, intelligent and robust forecasting techniques contribute to power supply system management at different levels (Hooshmand et al., 2013; Ozcanli et al., 2020; Sudheer and Suseelatha, 2015; Wang et al., 2019). In addition, intelligence in addressing energy-related problems are also increasing (Guyot et al., 2019; Lee et al., 2020; Mishra et al., 2020). From the perspective of model development, more complex models rather than simple machine learning and statistical models can significantly benefit from the so called the big data regime, and hence neural networks, which are the most prominent complex models for the massive amount of data, have been applied by researchers across many areas as in short term load forecasting recently (Hewamalage et al., 2020; Ozcanli et al., 2020). By considering all of these aspects, a new deep learning model for short term load forecasting is proposed, and compared with some other deep neural networks as deep learning models supersede conventional time series forecasting and machine learning methods. Deep learning methods, which are the state-of-the-art methods as per the recent literature, have currently been gaining popularity as a new tool to use in STLF applications (Mishra et al., 2020). Hence, they have become the new forecasting trend/method for STLF applications.

To create a comparative analysis of deep learning methods, and to prove the applicability of the proposed method for the major concerns of the providers that are one-hour-ahead and 24-hour-ahead predictions, a real-world application case for İstanbul is used to make load predictions. At the most conceptual level, the underlying study aims to make the following contributions to STLF field:

- 1-D CNN is rarely used for time series prediction when compared to the use of LSTM and GRU methods. Although some studies used 1-D CNNs for the STLF are existent in the literature, to the best of our knowledge, this study is the first one that introduces a novel method for STLF based on Video Pixel Networks (VPN). With comparative analyses, the VPN based 1-D CNNs method is a very competitive method in time series forecasting.
- By providing comparative analyses on a real dataset, this study provides insight for the managerial decision-makers in STLF in utilizing off-the-shelf deep learning methods and their variants for STLF problems,
- All of the methods utilized in this paper are examples of end-to-end learning which do not use any combined/hybridized heuristic techniques. Hence, by keeping things genuine, it makes understanding and application of these methods by managerial decision-makers in STLF simpler and more intuitive compared to the application of any combined deep learning-based methods.

The content of the paper is organized as follows. A brief introduction and motivation about STLF and the use of deep learning methods for STLF problems are given in Section 1. Section 2 is dedicated to the review of the extant literature on the use of deep learning methods for STLF. Section 3 provides a succinct description of the data, and the proposed and utilized deep learning methods. Section 4 provides the application details, and presents the comparative results and their discussion. The last section, Section 5, summarizes the study and its findings, and provides some concluding remarks.

2. Literature review

Prominent STLF studies date back to the 1950s (Hu et al., 2017). Many, including regression and other statistical (conventional time series techniques), classical machine learning, and deep learning-based methods, have been used for STLF. Some of them used hybridization of several methods (Bahrami et al., 2014; Hajirahimi and Khashei, 2019; Hernández et al., 2014; Liu et al., 2014), while others chose to use individual methods (Amjady and Keynia, 2011; Dudek, 2015;

Fan and Hyndman, 2012; Zhang et al., 2013). Numerous studies exist in the extant literature as examples of using regression and other statistical approaches as well as machine learning-based prediction methods for STLF problems. Artificial Neural Network (ANN) (Amjady and Keynia, 2011), ANN combined with Wavelet Transform (WT) and evolutionary algorithm (Amjady and Keynia, 2009), Support Vector Regression (SVR) combined with Particle Swarm Optimization (PSO) (Ceperic et al., 2013; Selakov et al., 2014), Autoregressive Integrated Moving Average (ARIMA) with SVR and Cuckoo Search Algorithm in the hybrid form (Kavousi-Fard and Kavousi-Fard, 2013), Extreme Learning Machine (ELM) (Zhang et al., 2013), Self-Organizing Maps with *k*-means and Multi-layer Perceptron (MLP) in the hybrid form (Hernández et al., 2014) are some the noteworthy machine learning-based studies for STLF. Fuzzy interaction regression (Hong and Wang, 2014), WT with ARIMA in the hybrid form (Lee and Ko, 2011), a semi-parametric additive model (Fan and Hyndman, 2012), Multiple Linear Regression (MLR) (Amral et al., 2007), ensemble learning-based regression trees (Dudek, 2015), Empirical Mode Decomposition (EMD) combined with Extended Kalman Filter (EKF), and ELM with kernel and PSO (Liu et al., 2014), WT combined with the gray algorithm, and PSO (Bahrami et al., 2014) are some of the many regression and statistical-based methods used for STLF.

Deep learning methods are applied to time series in a variety of application domains such as energy and fuels, image and video, finance and banking, among others (Torres et al., 2021). Recently, numerous artificial intelligence methods have been utilized and published for STLF applications (Mishra et al., 2020). Especially, the-state-of-the-art deep learning methods, which are advanced versions of shallow neural networks, have been utilized, mostly attributed to the emergence of the big data phenomenon, new and advanced algorithms, availability of powerful computational resources, and hardware advancements. It could be said that these methods have already superseded conventional time series and machine learning methods for STLF applications due to their efficient feature extraction ability and high prediction performances. Autoencoders (AEs), Recurrent Neural Network (RNN), LSTM and GRU, CNN, Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN), and Deep Boltzmann Machine (DBM) are frequently utilized as the-state-of-the-art deep learning methods for STLF (Almalaq and Edwards, 2017).

Some examples of LSTM and GRU for STLF are as in following studies: Wang et al. (2019) proposed a novel model based on Attention Mechanism (AM) and Rolling Update (RU) with BiDirectional-LSTM (Bi-LSTM). Du et al. (2020) used an attention mechanism based LSTM for STLF (a six-time step ahead forecasting) for their study as well. The method was compared with several methods including some time series methods, RNN variants, classical sequence to sequence (seq2seq) variants by use of LSTM and bi-LSTM. Fan et al. (2019) considered several models with several inference scenarios for forecasting strategies. The models are mainly based on recurrent neural network versions. Bedi and Toshniwal (2019) used LSTM for prediction and k-means for segmenting load data seasonally in hybrid form for Multi-Input Multi-Output (MIMO) strategy for forecasting task. Kong et al. (2017) used clustering and deep learning in hybrid form as well. They used LSTM with the Density-based Spatial Clustering of Applications with Noise (DBSCAN) method for the Australian Smart Grid Smart City dataset. The DBSCAN method was used for detecting differences between aggregated and individual loads, and then the predictions were made by LSTM method with different prediction aggregation levels. Memarzadeh and Keynia (2021) proposed a hybrid method for STLF and electricity price forecasting. WT, feature selection and LSTM were used in hybrid form in this study by comparing it with different methods by utilizing two different datasets.

Shi et al. (2017) used both one layer RNN and deep RNN for two different datasets to show efficacy of RNNs in STLF. Another LSTM used study belongs to Narayan and Hipel (2017) where the authors utilized LSTM for STLF with a 10-year electricity load that belongs to Ontario, Canada. Kumar et al. (2018) aimed to show the efficiency of uses LSTM and GRU methods for STLF where GRU was found superior to LSTM for the utilized load data. Zheng et al. (2018) utilized LSTM and GRU to show the efficacy of the-state-of-the-art methods under different scenarios by considering different time steps and varying input combinations in addition to feature engineering with some machine learning methods in the study. Another study that compares LSTM and GRU in STLF was done by Xiuyun et al. (2018). The authors compared LSTM and GRU methods, GRU in turn outperformed LSTM method in this study. Xu et al. (2020) proposed a method based on residual networks and ensembling, and compared their methods with several different RNN variants by utilizing two public available datasets. According to the results obtained, their method outperformed the remaining methods. Skomski et al. (2020) utilized sequence to sequence (seq2seq) RNNs for short term load forecasting. They utilized several datasets and made different predictions for different time resolutions through the used method. Chitalia et al. (2020) compared nine different RNN/CNN methods and their hybrid forms. They used five different datasets for their study. Choi et al. (2018) performed preprocessing on input data by transforming one-dimensional data into two-dimensional one by utilizing the methodology proposed in Amral et al. (2007), and Lee and Ko (2011). ResNet with LSTM was proposed, wherein ResNet extracted the latent features in the load data while LSTM performed the forecasting tasks. Wen et al. (2020) proposed GRU based deep Recurrent Neural Network (DRNN-GRU) for one step ahead prediction of residential hourly load data in Texas, USA by considering different aggregation level of dataset. The proposed method was compared to several Deep Neural Network (DNN) and time series methods. Muzaffar and Afshari (2019) used LSTM for 1–2 days ahead, one-week, and one-month ahead predictions, and compared the method several conventional time series forecasting methods.

Some CNN applications for STLF are as in following studies: Cai et al. (2019) used Gated CNN (GCNN) and Gated RNN (GRNN) models, and these models were compared with Seasonal Autoregressive Integrated Moving Average with External Inputs (SARIMAX) method under direct and recursive forecasting scenarios. The performances of the methods were compared in terms of accuracy, robustness, and generalization ability. He (2017) used parallelly structured CNN-RNN since the load data in the research had dynamic patterns and periodicity. Parallelly structured CNN modules for the extraction of valuable features and RNN for capturing temporal dynamics of historical load data were used. Dong et al. (2017a) proposed an integrated model by combining *k*-means with CNN. *k*-means is used to group the data collected from different regions, and predictions were performed by CNN method. Dong et al. (2017b) proposed a CNN model based on a bagging strategy. They transformed one-dimensional sequential data into such an image to utilize correlation between input sequences, and then, convolutional neural network makes forecasting. Another study of one-dimensional data transformation into two dimensions for forecasting can be found in Han et al. (2019). In this study, the authors proposed time-dependent CNN (TD-LSTM) by exploiting the feature extraction ability of CNNs, Cycle LSTM (C-LSTM), to extract time dependence in sequential data. Sadaei et al. (2019) used the newly proposed method which combines fuzzy time series and CNN for one hour ahead prediction. They used the hourly load of a power supply system in Malaysia and made a benchmark of the proposed method with several conventional time series methods, LSTM, and some of its variants. Kim et al. (2019) proposed the Recurrent Inception Convolutional Neural Network (RICNN) by combining RNN with one dimensional CNN. They used half-hourly power usage data in South Korea to make 24-h ahead prediction besides 3, 5 and 7 day-ahead predictions. The authors made comparisons of the proposed method with MLP, RNN, and 1-D CNN, and the proposed method outperformed the compared methods.

Some application examples of deep neural network and its variants are as in following studies: Guo et al. (2018) proposed an integrated

method by combining deep neural networks with kernel density estimation and quantile regression. The method outperformed gradient boosting and random forest models according to obtained results in the paper. Qiu et al. (2017) presented a hybrid model by combining EMD with RBM. A comparative analysis was performed for several models, including Factored Conditional Restricted Boltzmann Machine (FCRBM), RBM, SVR, ANN, and RNN, under different scenarios with several time horizons and time resolutions by Mocanu et al. (2016). Din and Marnerides (2017) applied deep feed-forward neural networks and recurrent deep neural network to a publicly available dataset to show the effectiveness of the state-of-the-art deep learning methods for STLF by the models (i) incorporating only time relevant features, (ii) incorporating both time and frequency relevant ones. Hui et al. (2017) used a hybrid algorithm by combining stacked RBMs with a genetic algorithm. Four concatenated RBMs were used for prediction, while the genetic algorithm was used for optimizing the weights and thresholds of deep neural networks. Fan et al. (2020) proposed a deep belief network based on EMD and ensembling in hybrid form by utilizing Australian Energy Market Operator (AEMO) data for proving the applicability of the method. Yu et al. (2018) proposed an integrated model that consists of EMD for decomposition of load data, DBN for training the weights of stacked RBMs, and Local Predictor (LP) for constructing neighbor sets among forecast samples in hybrid form. Yang et al. (2019) proposed a deep ensemble learning method based on clustering and the Lasso method for both 1-h and 24-h ahead predictions for a real dataset from China. The proposed method was compared with some quantile machine learning methods and LSTM. Hafeez et al. (2020) used an integrated method by combining Modified Mutual Information, Factored Conditional Boltzmann Machine (FCBM), and the newly proposed genetic wind-driven optimization algorithm.

Tong et al. (2018) used a hybrid model combining stacked denoising autoencoders and SVR, and the proposed model is shown to outperform SVR and ANN. Chen et al. (2018) proposed a novel model with residual connections and dense networks. Extensive comparisons with publicly available datasets are performed that validated the high-performance of the proposed model. Li et al. (2020) proposed a new loss function in their study, and the authors compared the performances of CNN, LSTM, and multilayer perceptron with different metrics. They utilized the same datasets in Chen et al. (2018), and found their methods superior to compared methods in the paper. Fang and Yuan (2019) investigated performance enhancing techniques for deep learning methods used in time series forecasting by using different datasets and different forecasting horizons. Liu et al. (2020) made comparative analyses for 24-hour ahead prediction using a real dataset obtained from Jiangsu Province, China. Time series, classical regression, and deep learning methods were chosen for the application in the paper.

Deep learning architectures are successfully utilized for different time series domains, and their advantages and limitations are extensively studied by Torres et al. (2021), where detailed information and best practices about deep learning for time series forecasting is also provided.

Brief representation of mentioned literature is given in the following table (see Table 1).

In the literature gleaned, to the best of our knowledge, there is no application for VPN in time series forecasting domain. Hence, it was proposed to apply VPN to a time series problem to show the efficacy of CNNs in sequential data modeling as well.

3. Methodology

In this section, we explain data collection and preprocessing, and sequence-to-sequence deep learning methods (LSTM, GRU, and their variants) along with the proposed VPNs based 1-D CNN method for STLF and its application details.

3.1. Data collection and pre-processing

In this study, the hourly collected electricity load and temperature data pertaining to the 2015–2017 time period, from one of the largest metropolitan cities in the world, Istanbul, Turkey is used. After signing a confidentiality and nondisclosure agreement, the data were extracted, organized/tabulated, and given to us in a transactional/raw format by the electricity provider firm, CK Bogazici Elektrik. The granularity of the time series data consisted of hourly electricity load demands and corresponding temperature values, as it was used by the firm in their forecasting tasks. An extensive data preprocessing that included tasks like data normalization, lag identification, and dummy variable inclusion were performed.

For data normalization, since feature normalization contributes to fast convergence and numerical stability of neural networks training, a zero-mean normalization procedure is applied to the load and temperature variables in the study. Normalization is performed according to Eq. (1),

$$x_i' = \frac{x_i - \mu}{\sigma} \tag{1}$$

where x_i', μ and σ correspond to zero mean value of ith example, x_i, in a time series dataset, the mean, and standard deviation of the dataset, respectively. Test data were normalized using the mean and standard deviation of the training data. For final predictions, denormalization for the test data outcomes.

The lag value for the time series load data is identified as 168 time-steps back according to Partial Autocorrelation Function (PACF) results. This means that the load value at time step t is affected by previous 168-time steps' values. When load characteristics for each day are examined in the dataset, different characteristics of each day, which are especially salient between weekdays and weekends, were identified. Hence, we included dummy variables to account for these characteristics, as most of the previous STLF studies have been done and reported in the literature (Chen et al., 2018; Guo et al., 2018; Liu et al., 2020).

3.2. LSTM and GRU

RNNs enable long-term computing dependencies of sequential data via recurrences of feedback within layers; thus, they overperform conventional feed-forward neural networks because of this time delay ability (Wen et al., 2020). In an RNN, the current time step output is computed by the current time step input, and output conveyed from previous time steps by the recurrence. Eq. (2) shows the value of an RNN hidden state at time step t,

$$h_t = f\left(w_i x_t + w_h h_{t-1} + b\right) \tag{2}$$

where h_t corresponds to output at time step t, f is any activation function, x_t and h_{t-1} correspond to input at time step t and the value of the output one time step before, respectively. w_i and w_h also denote weights of inputs at time step t and output of the previous time step, respectively. b denotes the bias term for the hidden layer.

Long term dependency in data causes gradient vanishing and/or explosion in backpropagation, which is called *exploding and vanishing gradient* in the literature, resulting in degradation of RNN efficiency. The vanishing gradient problem is much more complicated than the exploding gradient problem. To address the "exploding gradient problem", gradient clipping was proposed by Pascanu et al. (2013). On the other hand, LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) which use gating mechanisms, was proposed as a solution for the vanishing gradient problem, and they have been producing promising results. In an LSTM cell, there are input, output, and forget gates, each of which conveys partial information of the LSTM cell. An LSTM cell is shown in Fig. 1.

In Fig. 1, input x_t and hidden state at previous time step h_{t-1} are concatenated. Then, the concatenated input is processed by a sigmoid

Table 1
Brief representation for the relevant literature.

Study	*Method(s)*	*Used dataset(s)*	*Forecast horizon(s)*
Tong et al. (2018)	Denoising Autoencoder + SVR	A US dataset	
Wang et al. (2019)	Attention Bi-LSTM	Australian Electricity Data	
Fan et al. (2019)	RNN + 1-D CNN	Hong Kong electricity data	
Bedi and Toshniwal (2019)	k-means + LSTM	Data of Union Territory Chandigarh	
Li et al. (2020)	MLP, CNN, LSTM	ISO New England and North American Utility datasets	24-h ahead
Choi et al. (2018)	ResNet + LSTM	Korea Electric Power Cooperation dataset	
Xu et al. (2020)	Newly proposed model based on DNNs	ISO New England and North American Utility datasets	
Liu et al. (2020)	CNN	A dataset belonging to Jiangsu province in China	
Wen et al. (2020)	RNN + GRU	A dataset belonging to Texas in US	
Dong et al. (2017a)	k-means + CNN	A big electricity dataset from a power industry	1-h ahead
Sadaei et al. (2019)	Fuzzy time series + CNN	Load dataset of a power supply system in Malaysia	
Qiu et al. (2017)	EMD + RBM	AEMO dataset	Half-an-hour & 24-h ahead
Yang et al. (2019)	Deep NN + clustering + Lasso	A real dataset from the Irish Commission for Energy Regulation	
Cai et al. (2019)	Gated RNN + Gated CNN	A US dataset	
Chitalia et al. (2020)	CNN + LSTM and LSTM variants	Several different load datasets	1-h & 24-h ahead
Chen et al. (2018)	DNN	ISO New England and North American Utility datasets	
He (2017)	Parallely structured CNN-RNN	A city of North China dataset	
Zheng et al. (2018)	GRU	A Chinese dataset belongs to Nanjing Huafu	One-step ahead
Din and Marnerides (2017)	RNN	ISO New England dataset	
Hafeez et al. (2020)	Modified mutual information + FCRBM + Genetic wind-driven optimization algorithm	PJM electricity market dataset	24-h & one-week ahead
Guo et al. (2018)	DNN + Quantile regression + kernel density estimation	A dataset belongs to Jiangsu province in China	
Shi et al. (2017)	One layer RNN and deep RNN	New England system and Irish datasets	
Narayan and Hipel (2017)	LSTM	10-year load dataset for Ontario, Canada	
Hui et al. (2017)	RBM + genetic algorithm	Load data for an electric power system	
Dong et al. (2017b)	CNN	A power industry system dataset, New York ISO dataset, Electric Reliability Council of Texas dataset	Multi-step ahead
Du et al. (2020)	Attention LSTM	Individual household electric power consumption dataset from UCI repo	
Kumar et al. (2018)	LSTM and GRU	A household load dataset	
Fan et al. (2020)	DBN + EMD	AEMO dataset	
Xiuyun et al. (2018)	LSTM and GRU	A load dataset for a Chinese region	
Kim et al. (2019)	Recurrent Inception Convolutional Neural Network (RICNN)	Half-hourly power usage data in South Korea	
Kong et al. (2017)	Density-based Spatial Clustering of Applications with Noise + LSTM	Australian Smart City dataset	Several multi-step aheads
Muzaffar and Afshari (2019)	LSTM	A one-year load dataset	
Yu et al. (2018)	EMD + DBN + LP	AEMO dataset	1-h and several multi-step aheads
Mocanu et al. (2016)	Factored Conditional Restricted Boltzman Machine (FCRBM)	Individual household electricity consumption in UCI (University of California,Irvine) repo	15-min, 30-min and one-day ahead
Han et al. (2019)	Time dependent CNN + Cycle LSTM	Two datasets belonging to Hangzhou in China and Toronto in Canada	One-week & two-weeks ahead
Memarzadeh and Keynia (2021)	WT + LSTM	Pennsylvania–Jersey–Maryland and Spanish datasets	One-week ahead
Skomski et al. (2020)	RNN	A US office building dataset	15-min and 1-h ahead

function that forms forget gate f_t, which determines what information is to be preserved and forgotten. Forget gate is computed by Eq. (3).

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \tag{3}$$

In Eq. (3), σ denotes sigmoid activation function while W_f and b_f correspond to weights and bias terms for the gate in order, and the square bracket symbolizes concatenation operation. Next, cell state that stores relevant information for successive layer, c_t, is determined by utilization of f_t, c_{t-1}, i_t and $\check{C}_t$ where c_{t-1}, i_t and $\check{C}_t$ denote cell state at the previous time step, input gate and candidate values conveyed for the next cell state, respectively. Computations for input gate and candidate values are given in Eqs. (4) and (5).

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{4}$$

$$\tilde{C}_t = T\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) \tag{5}$$

In Eqs. (4) and (5), W_i, W_c denote weights for input gate and matrices of candidate values while b_i and b_c correspond to bias terms for them, respectively. T corresponds to the hyperbolic tangent *(tanh)* activation function. In the next equation (Eq. (6)), proceeding cell state computation is performed, where $\odot$ symbolizes the element-wise

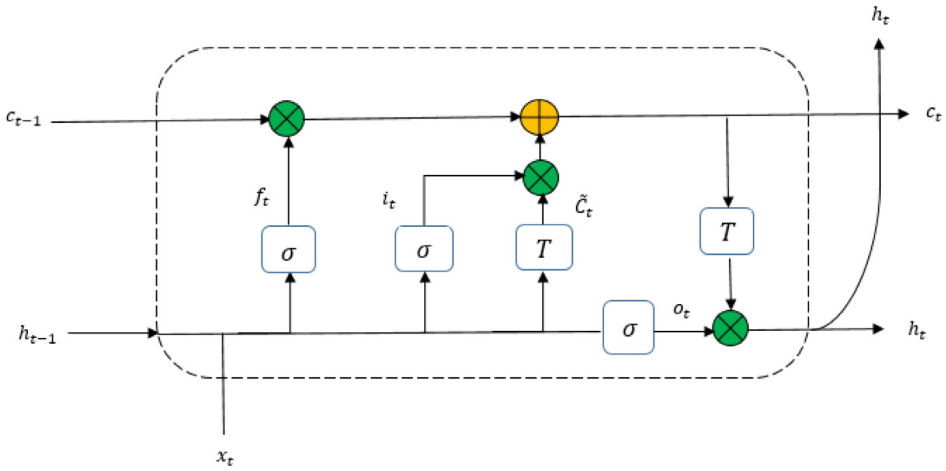


Fig. 1. An illustrative example of an LSTM cell.

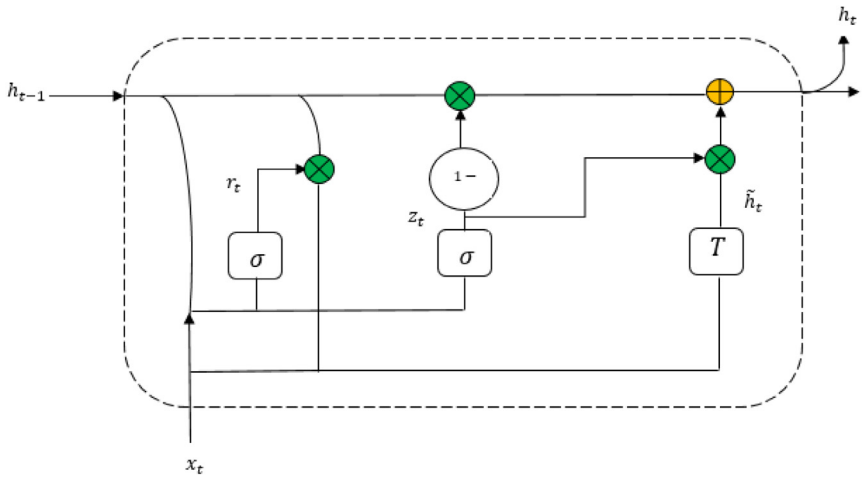


Fig. 2. An illustrative example of a GRU cell.

multiplication.

$$c_t = f_t \odot c_{t-1} + i_t \odot \check{C}_t \tag{6}$$

In Eqs. (7)–(8), output gate o_t, which conveys the filtered information by means of forgetting and updated cells, and h_t computations are given.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{7}$$

$$h_t = o_t \odot T\left(c_t\right) \tag{8}$$

LSTM has some variants, which are constructed by some merging and modifications in gating mechanisms such as Clockwork RNN, Depth-Gated LSTM, and GRU. GRU is one of the most frequently used LSTM variant in the literature. A schematic for GRU is depicted in Fig. 2.

The forget and input gates are merged into a single update gate, z_t, along with merging the cell state and hidden state. Unlike LSTM, GRU consists of two gates, which are called update and reset gates. The update gate determines what information will be conveyed from previous time steps to next time steps while the reset gate, r_t, determines what information will be forgotten. Computations for update and reset gates, current memory content, and final memory content are given in the following equations, respectively.

$$z_t = \sigma\left(W_z\left[h_{t-1}, x_t\right]\right) \tag{9}$$

$$r_t = \sigma\left(W_r\left[h_{t-1}, x_t\right]\right) \tag{10}$$

$$\check{h}_t = T\left(W\left[r_t * h_{t-1}, x_t\right]\right) \tag{11}$$

$$h_t = \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \check{h}_t \tag{12}$$

In this study, LSTM and GRU with 3 layer stacks, and encoder–decoder methods with LSTM and GRU are used alongside the 1-D CNN. Simple stacked LSTM and encoder–decoder methods can be seen as in Fig. 3 and Fig. 4, respectively.

In Fig. 3, each rectangle box contains the LSTM cell which its details was depicted in Fig. 1. Each x_t is the input of relevant time step *t*. Dense box refers to a fully connected layer, and $\hat{y}$ is the predicted value of the method. In Fig. 4, the encoder state corresponds to the output of the last LSTM cell stored to be fed into the first LSTM cell of the decoder part of the method. As it will be mentioned in the application settings, since 168 time lags will be utilized in our modeling, x_t will correspond to 168thinput instance in simple LSTM and GRU, Autoencoder LSTM and GRU methods.

In the stacked and encoder–decoder GRU methods, LSTM cells are replaced by GRU cells; thus, the only difference in these methods from their LSTM counterpart is GRU cell utilization. Therefore, we did not include their figures in the interest of the brevity of the paper. Encoder–decoder models make some abstraction of input, and store the extracted information in an encoder part. Then, a decoder part is fed

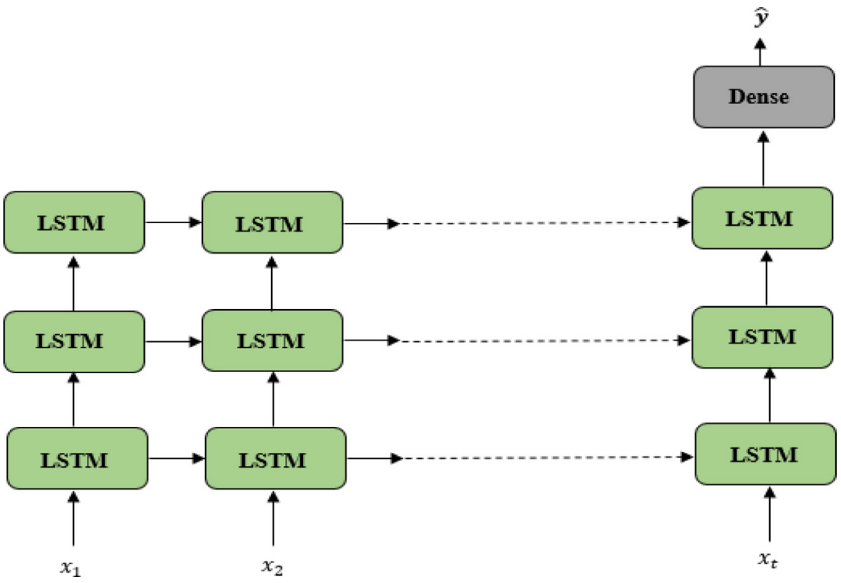


Fig. 3. Stacked LSTM.

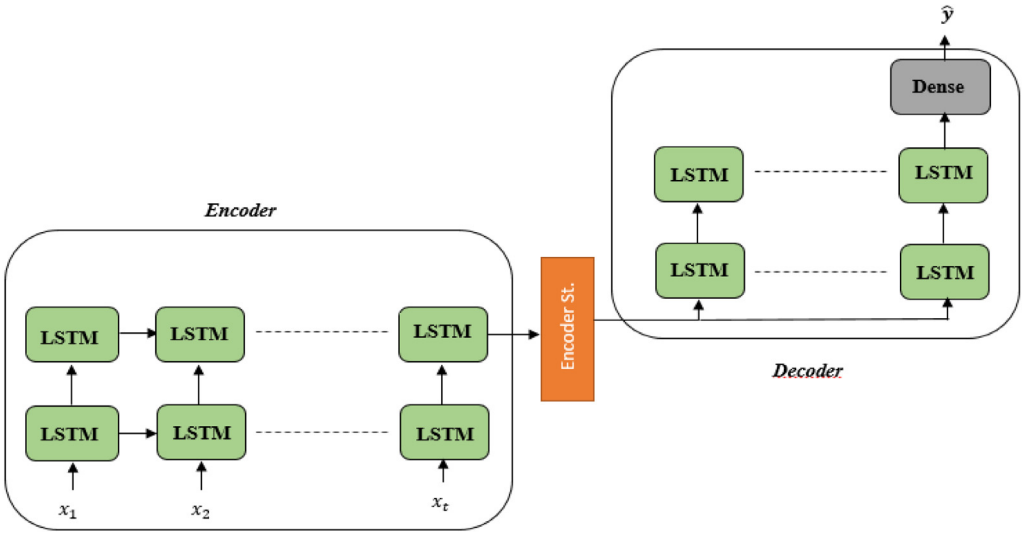


Fig. 4. Encoder–Decoder LSTM.

by the stored information within the encoder part, thereby performing regression or classification tasks with powerful representation. Since encoder–decoder architecture has proven its efficiency in representation learning with respect to simple or stacked LSTM/GRU methods, the encoder–decoder architecture is included for comparisons in the study.

3.3. Proposed 1-D CNN method

CNN is a kind of deep neural network architecture that uses shared parameters by local connectivity, which is different from conventional feed-forward networks. CNNs have many application areas such as computer vision and object tracking (Pang et al., 2017), sensory fault detection and identification (Dong, 2019), image processing and classification (Kara et al., 2020), medical image classification (Yang et al., 2021), human activity recognition (Gil-Martín et al., 2020), writer identification (Javidi and Jampour, 2020). Although their use for time series is rare, CNNs are gaining attention in time series regression problems recently (Binkowski et al., 2018; Dong et al., 2017a). Since the use of CNNs for time series problems has been emerging only recently, in this study, we chose to utilize them for the comparative analysis of deep learning methods with a real case application to show their feasibility and efficiency in addressing time series forecasting problems.

A CNN applies convolution operation, which nothing but a kernel with a fixed length, k, slides over the input x, thus, produces a representative output as a result of this overlapping operation. The convolution operator, which makes element-wise multiplication over the input patches of k with a kernel by a pre-defined stride and/or dilation rate and sums up the results, generates a feature map at the end of the operation. The number of filters used in convolutions determines the depth of output, while stride and padding determine the spatial dimensions of the feature map(s) (Gasparin et al., 2019).

In this study, we used a one-dimensional CNN that is deemed to be suitable for time series. The convolution operation for the one-dimensional time series input $x \in R^d$, where the weight for the one-dimensional kernel ($w \in R^f$ for the ith element of the convolution) between input and the kernel, is given in Eq. (13).

$$g_{(i)} = (x * w)(i) = \sum_{k=0}^{f-1} x(i-k).w(k) \tag{13}$$

where $g \in R^{d-f+1}$ with the same padding method used.

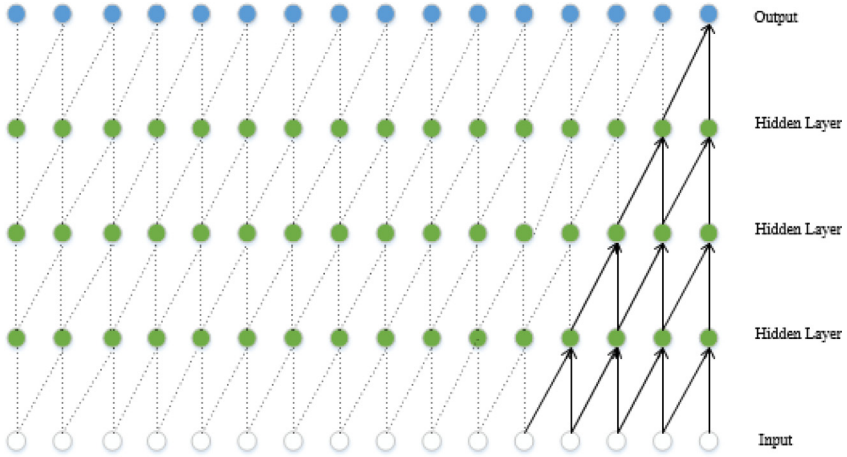


Fig. 5. A stack of dilated causal convolutional layers.

A causal convolution means that the convolution restricts a model not to violate the ordering of the modeled data. Thus, it does not allow the dependence of prediction at time t on future values (Van Den Oord et al., 2016). An illustrative example of a stacked causal convolution can be seen in Fig. 5.

Causal convolutional layers are preferable to RNNs since they have no recurrent connections; thus, they take less computational time for training a network. This fact is more salient when modeling of very long sequences is performed. However, to augment the receptive fields of a network, requirements of many layers or large filters in causal convolutions pose a challenge in the training phase. Dilated convolutions can be used in 1-D CNNs to address this challenge. A dilated convolution performs convolution by skipping inputs with certain strides. This behavior is similar to pooling or strided convolutions except for the fact that dilated convolution yields the same sized output with input (Van Den Oord et al., 2016).

A dilated causal convolution operation for an input $x \in R^d$ where the weight for one-dimensional kernel $w \in R^f$ with dilaton rate d is given in Eq. (14).

$$h(i) = \left(x *_d w\right)(i) = \sum_{k=0}^{f-1} x\left(i - d_k\right) w(k) \tag{14}$$

While the receptive field, r, grows exponentially by $r = 2^{L-1}k$, this growth is linear for causal convolutions with the depth of the network that is computed by $r = k(L-1)$. Here L denotes the length of the modeled sequence (i.e., time-series data length). This exponentially grown receptive field enables effectively modeling through long-range temporal dependencies in sequences. In addition, this feature contributes to preserve the resolution of input and provide computational efficiency by fewer parameters in the network (Van Den Oord et al., 2016). This network structure is used by Bai et al. (2018) as Temporal Convolutional Networks (TCN) for sequence modeling, and by Van Den Oord et al. (2016) for sequence modeling. An illustrative example of a dilated causal convolution structure can be seen in Fig. 6.

Another important property for deep CNNs is to use of residual connections. In deep neural networks, depth has landmark importance for the extraction of low/mid/high-level features and abstractions by contributing to the performance of the networks. Hence, deep architectures of the neural networks yielded promising results by utilizing this property (Simonyan and Zisserman, 2014; He et al., 2015, 2016; Szegedy et al., 2015). On the other hand, very deep networks are hard to train due to the degradation problem of deep networks. A deeper model suffers from degradation problems, which is a result of adding more layers to a network rather than being resultant of overfitting problems. While the depth of a network increases, accuracy saturation and rapid degradation issues emerge, and they, in turn, result in higher training error. To mitigate these issues, identity mapping of added layers is proposed throughout a network that allows efficient training of very deep networks (He et al., 2016). With the deep residual learning, H_x, which corresponds to the desired underlying mapping, is rearranged as $F(x) := H(x) - x$. The recasting of the original mapping into $F(x)+x$ enables identity mapping throughout a network (He et al., 2016). By adding these identity learning mappings progressively, very deep networks can be designed without degradation problems. Such an identity block building is depicted in Fig. 7:

In addition to residual connections and dilation use with 1-D CNNs, another main ingredient used in our method is Residual Multiplicative Block (RMB) and Multiplicative Unit (MU) modules that were originally proposed for video processing task (Kalchbrenner et al., 2017). We adopt MU and RMB ideas in our representation and modeling of the STLF problem with some modifications in their gating mechanism.

A Multiplicative Unit consists of some gates that are merged into a convolutional layer. This layer contains a given input h of size T x c, where T and c correspond to the length of the input and the size of channels, respectively. Firstly, update denoted by u and three gates g_{1-3} are constructed by passing h into the MU, then element-wise multiplication and summation operations yield the output of the MU (Kalchbrenner et al., 2017). Update and three gates, and resultant output computations are given in Eq. (15).

$$\begin{aligned} g_1 &= \sigma\left(W_1 h\right) \\ g_2 &= \sigma\left(W_2 h\right) \\ g_3 &= \sigma\left(W_3 h\right) \\ u &= ReLU\left(W_4 h\right) \\ MU(h; W) &= g_1 \odot ReLU(g_2 * h + g_3 \odot u) \end{aligned} \tag{15}$$

where $\odot$ is the element-wise multiplication.

MU in the proposed method is somewhat different from the original *MU* proposed by Kalchbrenner et al. (2017) that replaces the tanh activation function in the original paper with the *ReLU* activation function, as seen in the last two computations in Eq. (15). To avoid vanishing of the gradients, and the outputs to be fed into consecutive layers through the *MU*s, replacing tanh with ReLU activation function contrary to the original concept of VPNs is proposed. As a result, the proposed method yields better results than the original concept since *ReLU* activation does not decrease the magnitude of the inputs in a *MU* contrary to tanh activation. So, the proposed method moderately contributes to performance of VPNs in time series forecasting domain by boosting its performance. Some part of the paper's novelty stems from this contribution along with the first VPNs application to time series forecasting domain.

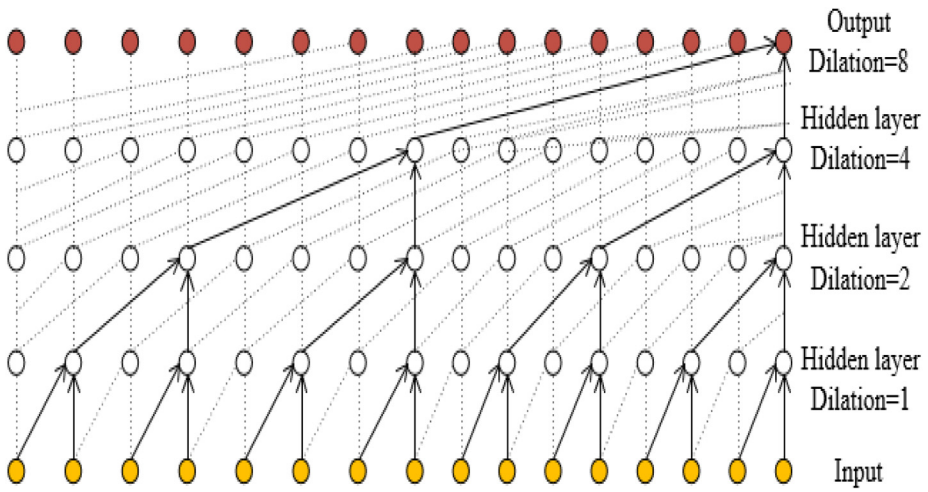


Fig. 6. A stack of dilated causal convolutional layers.

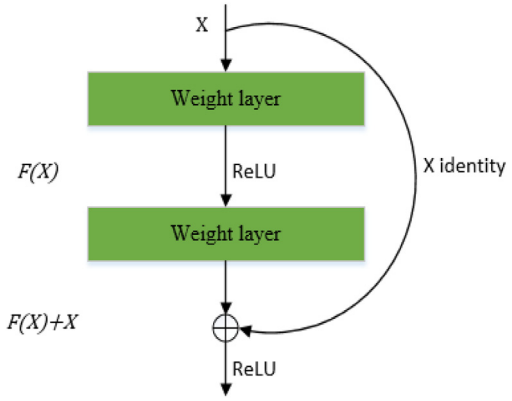


Fig. 7. A building block for residual learning.

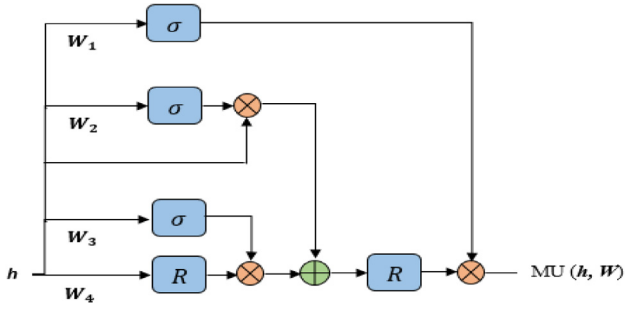


Fig. 8. MU structure.

In the case of *MU*, no distinction is available between memory and hidden states when compared to LSTM networks. It is different from Highway Networks and Grid LSTM, where the *MU* applies a nonlinearity to the input h (Kalchbrenner et al., 2017). An illustrative example of the changed *MU* is given in Fig. 8, where R denotes the Rectified Linear Unit (*ReLU*) activation function, and σ denotes the sigmoid activation function, as mentioned before.

Another ingredient for the proposed method is Residual Multiplicative Block (RMB), which is an essential part of a VPN. Two *MU* layers are stacked subsequently to enable easy gradient propagation through many layers with a residual connection of input to output in an RMB. Firstly, the size of channels of input h are halved at the entry of the first MU by applying 1×1 convolution with the linear transformation (i.e., no activation is applied), then, the output of the first *MU* is fed into the second *MU* and processed in the second *MU*. Finally, a linear projection by a 1×1 convolutional layer is applied to the output of the second *MU* to double its size of channels. After doubling the size of channels of this output, the input h is added to the output by using the residual connection (Kalchbrenner et al., 2017). Computations for an RMB module are given in Eq. (16).

$$\begin{aligned} h_1 &= W_1 * h \\ h_2 &= MU(h_1; W_2) \\ h_3 &= MU\left(h_2; W_3\right) \\ h_4 &= W_4 * h_3 \\ RMB(h; W) &= h + h_4 \end{aligned} \tag{16}$$

The *RMB* module structure is shown in Fig. 9, and the proposed 1-D CNN architecture is graphically depicted in Fig. 10.

3.4. Application case

In this section, the application case's implementation details by describing the method level specification with their parameter settings are presented. In all stages of experiments and related computations, Keras 2.2.2 with Tensorflow 1.10.0 as backend in Python 3.6 environment is used. A desktop workstation with Intel® 12 Core™ i7-5820K CPU and an NVIDIA GeForce GTX 1080 8 GB graphical processing unit (GPU) is utilized for the computations.

A graphical illustration of the used data, loads between 2015–2018, can be seen in Fig. 11 which depicts the hourly loads demand for three years.

As seen from Fig. 11, the time series has a stationary mean. Seasonal, weekly and daily cycles are evident in the series, thus we considered these cycles in the modeling stage. Weekly cycles for a month is depicted in Fig. 12.

Adding seasonal indicators as dummy variables did not improve the prediction accuracy. This outcome may be attributable to the fact that the effects of seasons on the short-term load demand do not warrant explicit consideration. However, inserting the effects of daily cycles by dummy variables into the input set improved the prediction results because each day exhibited distinct patterns for the load consumption. Hence, dummy variables for days were included in the input set, as given in Eq. (17). Dummy variables, presented as one-hot-encoding of days by examining their patterns, are created, as shown in Table 4. Different daily load consumption characteristics of day-of-the-week are shown in Fig. 13.

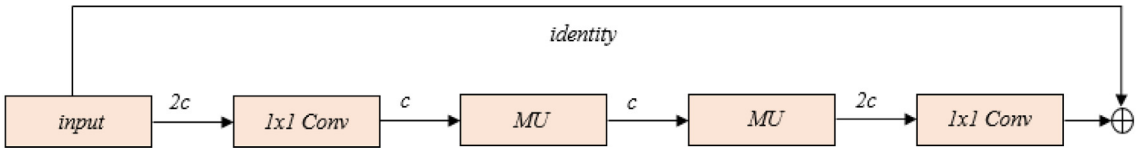


Fig. 9. The RMB module structure.

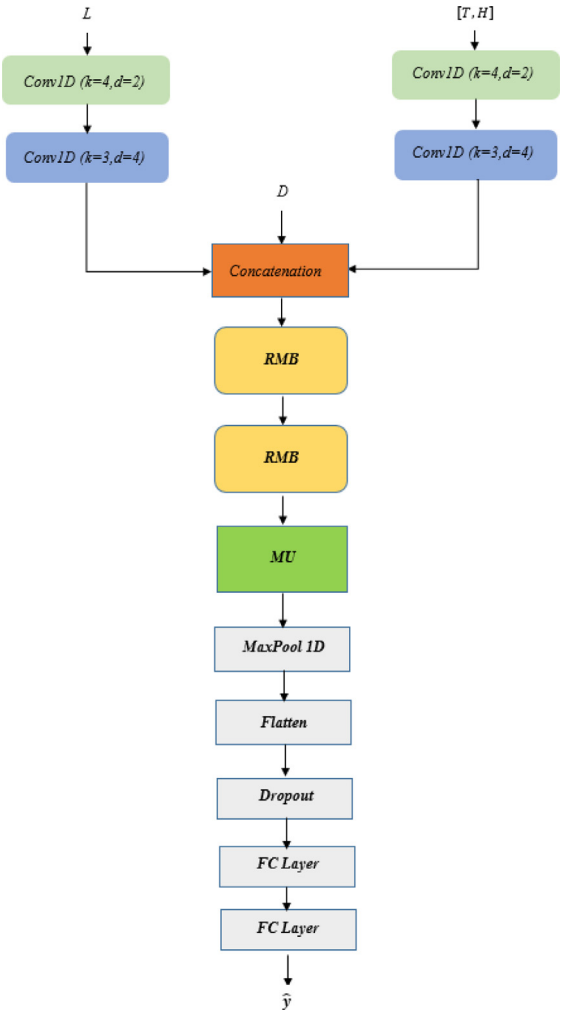


Fig. 10. The proposed model.

In addition to these analyses, the importance of the weekly cycles is investigated in modeling. As it will be mentioned in the methods' application details, justification of the use of 168 time lags for input set construction by examining Autoregression (AR) process for the dataset is performed. For this reason, Partial Autocorrelation Function (PACF) plot is depicted in Fig. 14.

Partial Autocorrelation Function in Fig. 14 justifies the plausibility of 168 time lag use in the input set beside the fact that most of the short term load forecasting studies use this time lag for their forecasting tasks.

After these preprocessing stages, the time series dataset is converted into a supervised learning dataset by shifting each input feature one time-step-ahead for the successive outputs, in turn, the time series dataset is structured as a supervised learning problem. An input set and the corresponding output for this input set are given in Eq. (17).

$$Input = \begin{bmatrix} D_{t-168,1} & \cdots & D_{t-168,7} & T_{t-168} & L_{t-168} \\ D_{t-167,1} & & D_{t-167,7} & T_{t-167} & L_{t-167} \\ \vdots & \ddots & & \vdots & \\ D_{t-2,1} & \cdots & D_{t-2,7} & T_{t-2} & L_{t-2} \\ D_{t-1,1} & & D_{t-1,7} & T_{t-1} & L_{t-1} \end{bmatrix}, Output = L_t \tag{17}$$

where D denotes dummy variable for a day, T denotes temperature, and L denotes load. $D_{i,j}$ corresponds to dummy variable for jth day of ith time step. T_i and L_i correspond to temperature and load values of ith time step respectively where L_t denotes the load value to be predicted by the input set. At the end, an input set for one training example constructs 168×9 dimensional matrix. In the proposed 1-D CNN method, there is a tiny difference for the input construction that is unique to the method's structure as seen in Fig. 10. In the method, temperature and daily cycles with 168 length are concatenated, and then this input along with the input for the load with 168 length were fed into the model separately in an exogenous variable manner to get more abstraction of the load and temperature by the method in advance. Then, the outputs of the temperatures and loads are concatenated with the dummy variables for days, thus an input matrix was constructed as the same with the remaining deep learning methods after this part of the method. This same setting in the input matrix was tried for simple GRU, LSTM, AutoEncoder GRU and LSTM methods to get more abstraction from the dataset, however, they did not work well with this setting so that we utilized somehow different input settings for the RNN based methods and the proposed 1-D CNN.

Different prediction strategies have been proposed for a multi-step-ahead time series forecasting task (Cai et al., 2019; Fan et al., 2019). The most common ones are recursive, direct, and Multi-Input Multi-Output (MIMO) strategies.

- ***Recursive strategy***: A single forecasting is performed by a given input sequence. Then, subsequent forecasts are performed by iteratively feeding the resultant output into successive inputs until multi-step forecasting is completed. For example, forecasted values vector o_t until time step t is used to forecast value at time step $t+1$ by being added to the last t sequence for input vector for $t+1$.
- ***Direct strategy***: In this strategy, n multi-step ahead forecasts are performed with n predictors which they have the same input. Different predictors are generated by the same input in this strategy, and they directly make forecasting for a specific time step value.
- ***Multi-Input Multi-Output (MIMO) Strategy***: In this strategy, a given multi-input sequence predicts a whole output sequence in one-shot with a single prediction function.

In the applications, a multi-step-ahead prediction setting beside one-step-ahead prediction has to be deployed when focusing on making 24-hour-ahead forecasting with the data. The recursive strategy for all the methods is used since the MIMO strategy does not fit our data, and direct strategy requires 24 distinct model applications for every configuration, thereby increasing computation times significantly. In addition to the restriction of direct strategy use for computational concerns, an individual value at time step t is affected by the values of 168-time steps back, which is inherently included in the use of recursive strategy contrary to the direct strategy.

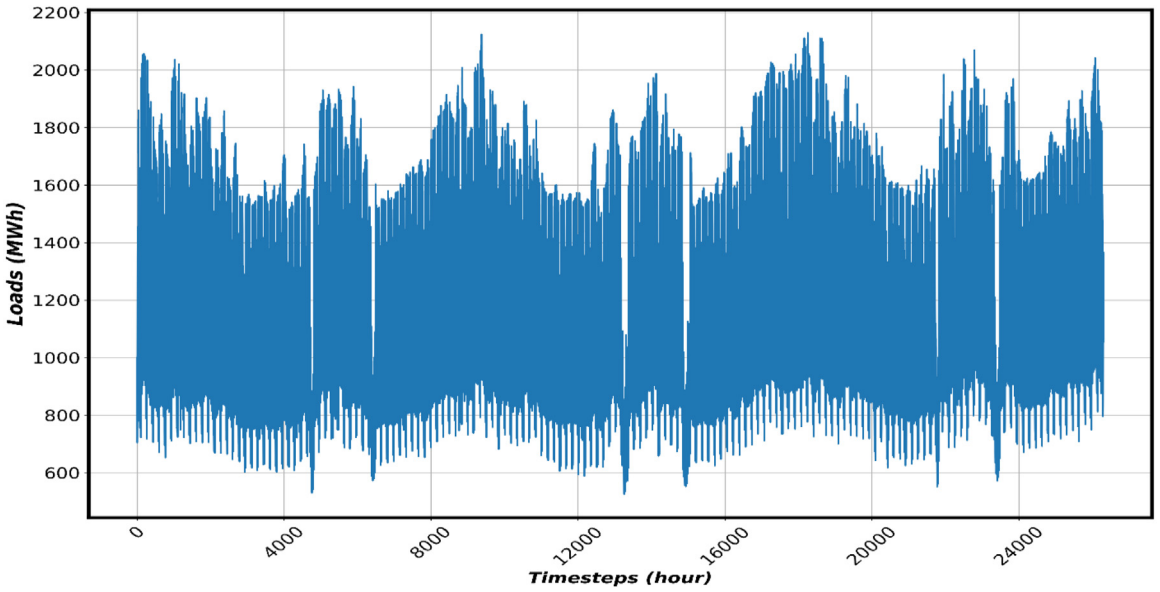


Fig. 11. The whole time series used.

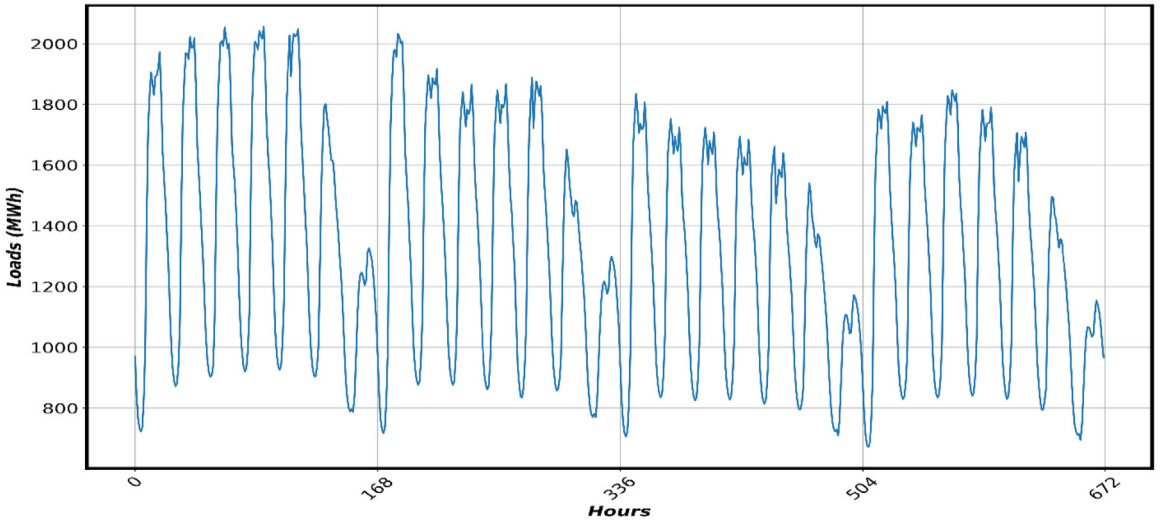


Fig. 12. Weekly cycles.

As mentioned before, stacked LSTM and GRU, encoder–decoder LSTM and GRU, and the proposed 1-D CNN is utilized and compared to each other in this study. The application settings of these deep learning methods will be presented in detail in the following subsections.

In stacked LSTM and GRU applications, several possible stack levels were systematically experimented. In the end, LSTM and GRU with three stacks, which can be seen in Fig. 3, performed the best according to the performance metrics used in our experiments. Hence, LSTM and GRU were implemented with three stacks in the comparative analysis of the methods. Similarly, both encoder–decoder LSTM and GRU methods performed the best with two-level stacks in each of the parts; therefore, these specifications (two stacks in the analysis) were utilized in the comparative analysis.

The stacked LSTM and GRU layer settings which are the resultant of extensive grid search study are given in Table 2. In the grid search experiment, the filter sizes for LSTM and GRU layers are [16, 32, 64, 128, 256, 512, 1024]. The filter size range for the dense layer is [16, 32, 64, 128]. The same experimental filter setting range for simple LSTM and GRU methods is used for Autoencoder LSTM and GRU methods. Dense layer neuron size range for Autoencoder LSTM and GRU methods are experimented in between 10 and 100 by an increment rate of 10. Learning rates for simple LSTM and GRU, and Autoencoder LSTM and GRU methods are experimented in between 0.0025 and 0.005 by an increment rate of 0.00025. Batch size range is in between 12 and 120 by an increment rate of 12.

Table 2
LSTM and GRU layer settings.

	LSTM		*GRU*	
	Filter size	Activation type	Filter size	Activation type
1st *LSTM layer*	512	tanh and sigmoid	256	tanh and sigmoid
2nd *LSTM layer*	256	tanh and sigmoid	128	tanh and sigmoid
3rd *LSTM layer*	128	tanh and sigmoid	64	tanh and sigmoid
1st *Dense layer*	64	ReLU	20	ReLU
2nd *Dense layer*	1	Linear	1	Linear

As seen in Fig. 3, there are subsequently connected 168 LSTM cells on the horizontal line, which forms the time-steps for the input (i.e., 168-time steps). On the other hand, three horizontal lines stacked vertically to form the stacked model. Each LSTM at the bottom has

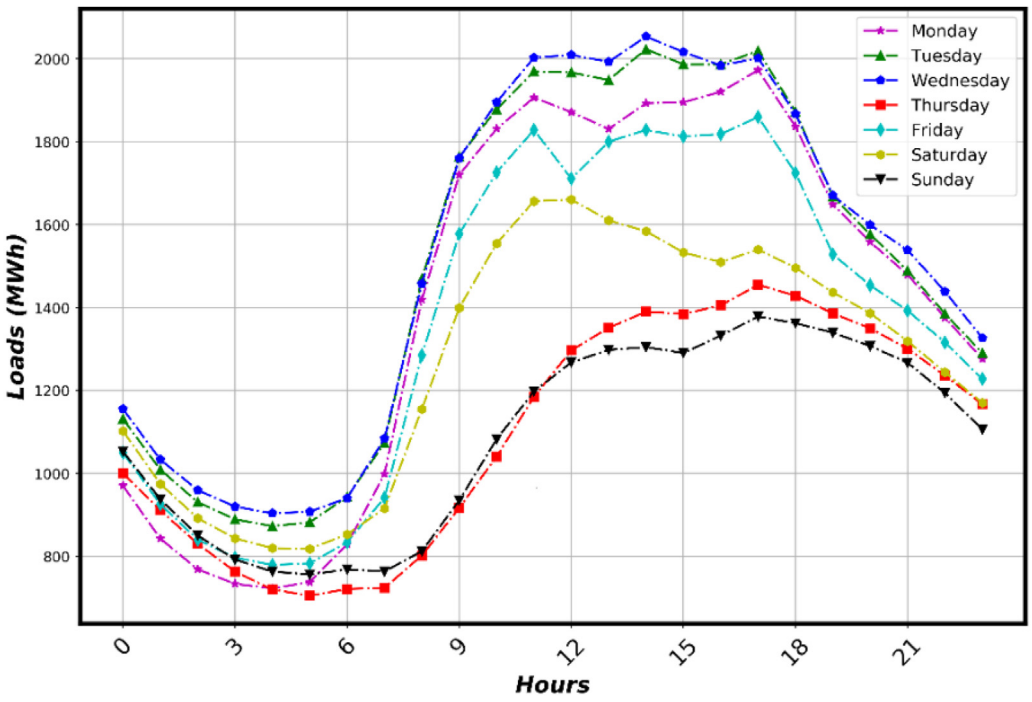


Fig. 13. Daily patterns for the load consumption.

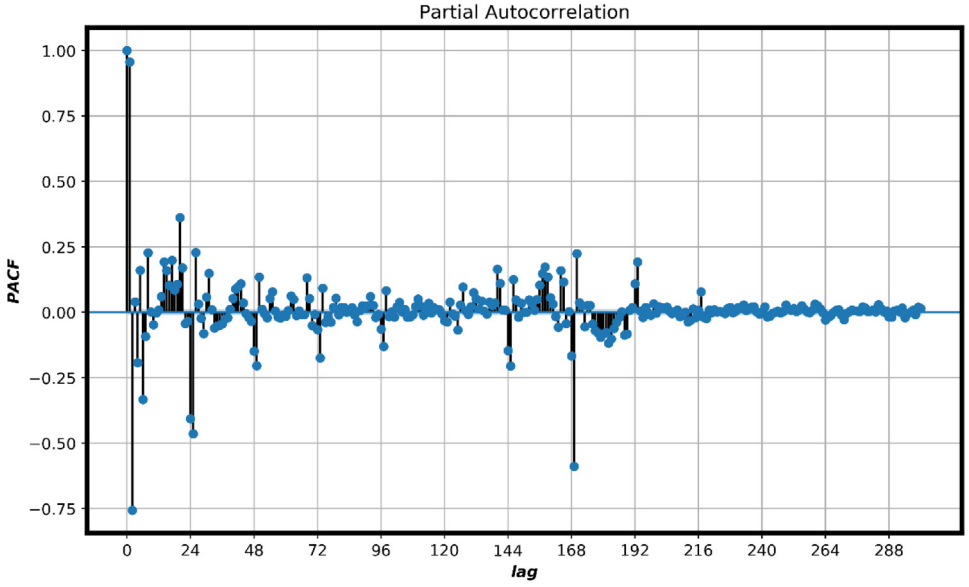


Fig. 14. PACF for the time series.

filters with a size of 512. In the mid-level of the stacks, each LSTM has filters with a size of 256, while LSTMs at the top level have a filter with the size of 128. First and second LSTM layers return their sequences to the cells above them. However, the third LSTM layer returns one output at the last LSTM cell, which is the output that gets connected to the dense layer with a layer size of 64. Finally, the first dense layer is connected to a second dense layer with linear activation to perform the regression task. In all LSTM layers, activation is set to tanh, and recurrent activation is set to sigmoid by default where tanh activation is used for hidden layer's computation, and sigmoid activation is used for the computations of input, forget and output gates. In GRU layers, the update and reset gates use sigmoid, and hidden layer uses tanh activation function by default. The stacked GRU structure is the same as the stacked LSTM structure, so we do not present its structural specification here again.

In each of the encoder–decoder models, two stacks are used both in encoder and decoder parts, as depicted in Fig. 4. Layer settings for these parts are presented in Table 3. In the encoder part, the first stack returns sequences, and then the outputs feed the second stack. This second stack stores conveyed information through this stack at LSTM/GRU cell at the end of it, and this information is embedded in the encoder state. This encoder state vector is repeated by RepeatVector as the encoder–decoder model suggests to feed the first stack of the decoder part. The output of the second stack of the decoder part is connected to a dense layer, and prediction is made by fully connected layers as in stacked LSTM/GRU.

Table 3
Encoder–Decoder LSTM and GRU layers' settings.

		Encoder–Decoder LSTM		*Encoder–Decoder GRU*	
		Filter size	Activation type	Filter size	Activation type
Encoder	1st *LSTM layer*	256	tanh and hard sigmoid	256	tanh and hard sigmoid
	2nd *LSTM layer*	128	tanh and hard sigmoid	128	tanh and hard sigmoid
Decoder	1st *LSTM layer*	128	tanh and hard sigmoid	128	tanh and hard sigmoid
	2nd *LSTM layer*	64	tanh and hard sigmoid	64	tanh and hard sigmoid
Dense	1st *Dense layer*	50	ReLU	50	ReLU
	2nd *Dense layer*	1	Linear	1	Linear

Table 4
Inputs for the first four methods.

Input	*Size*	*Description*
L	1	A numerical value for the load
T	1	A numerical value for the temperature
D	7	One-hot encoding for days (Categorical)

Table 5
Parameter and hyperparameter settings for the first four methods.

	Stacked LSTM	*Stacked GRU*	*Enc.-Dec. LSTM*	*Enc.-Dec. GRU*
Batch size	24	24	24	24
Epoch number	100	100	90	90
Learning rate	0.00375	0.003	0.0045	0.0045
Decay rate	0.0055	0.0055	0.0025	0.0025
rho	0.95	0.95	0.95	0.95
Epsilon	None	None	None	None

According to the experiments, only day dummies' inclusion to input features is considered for stacked and encoder–decoder LSTM/GRU methods. Hence, hourly load, temperature, and dummies for seven days are used for these methods. However, a different setting is used for the proposed 1-D CNN method as it will be explained later.

Inputs for stacked and encoder–decoder LSTM and GRU (the first four methods used in this study) methods are presented in Table 4. All inputs have length 168, and the load and temperature are continuous variables whereas day dummy variables created by examining the daily load patterns are one-hot encoded variables.

After conducting the experiments, the best performing parameters were included for comparisons. The best performing parameter and hyperparameter settings of stacked and encoder–decoder LSTM/GRU methods are given in Table 5. In all experiments, the Root Mean Square Prop (RMSProp) optimization algorithm was the best performing algorithm for the dataset, and hence, we utilized the RMSProp optimization algorithm throughout the comparisons.

In the proposed 1-D CNN method, one additional feature is used to represent the hours of days in the input features space. The inclusion of this feature did not produce promising results when used by the other methods utilized in this study. This feature is usually represented by a polar coordinate system—the hours of days are considered as forming a periodical cycle, which constructs a unit circle in a polar coordinate system, and the coordinates of this system correspond to the hours of the days. The periodical cycle for the hours of days is computed by Eq. (18).

$$h = \sin(\frac{2\pi t}{c}) \tag{18}$$

where h corresponds to the new feature vector for the hours, t and c denote a specific hour of a day and number of hours in a day *($c = 24$, cycle length)*, respectively.

As seen in Fig. 10, inputs are processed by using the exogenous variable and their inclusion to the model, which is quite contrary to the other models used in this study and their way of implementation.

Table 6
Inputs for the 1-D CNN.

Input	*Size*	*Description*
L	1	A numerical value for loads
T	1	A numerical value for temperatures
H	1	A numerical value for hours of a day
D	7	One-hot encoding for days (Categorical)

The inputs for load and temperature-hours are processed distinctly for preliminary information extraction to globally capture the features by using two CNNs. After these extraction operations for load and temperature, extracted features, and the inputs for days are concatenated, and then, these concatenated inputs are processed by the first RMB. The output of the first RMB is conveyed to the second RMB to be processed, and the resultant output of the second RMB is fed to the MU, which immediately follows the second RMB. Max-pooling operation is applied to the output of this MU, then flattening operation for the output is made to connect it to the succeeding fully connected layer. Finally, two fully connected layers are used for predictions. Dropout with the rate of 0.3 is used between the flattened output of the max-pooling layer and the first fully connected layer. A descriptive picture of the proposed 1-D CNN is presented in Fig. 10. The 1-D CNN method's inputs are presented in Table 6, and layer settings for the 1-D CNN method are shown in Table 7.

An extensive grid search for the filter sizes and learning rate for the proposed 1-D CNN method was conducted. The kernel size range for the first CNN in Table 6 is [25, 50, 100, 150, 200, 300]. This range is [450, 550, 650, 750, 850] for the second CNN in Table 7. The range for the first and second *RMB*s in Table 7 is [100, 200, 300, 400, 500]. The range for the last MU in Table 7 is [500, 600, 700, 800, 900, 1000] while the range for the first dense layer is in between 50 and 400 by an increment rate of 50.

In the proposed 1-D CNN method, linear projection is applied to the concatenated features by 1×1 convolution after the concatenation operation to reduce the filter size, which will then be fed into the first *RMB*. Then, *MU*s in the first *RMB* halved the filter size of this linearly projected layer into 200. The specific parameter and hyperparameter used as settings for the 1-D CNN method are given in Table 8.

In the experiments, the mean absolute percentage error (MAPE), which is one of the most commonly used performance measures for the time series forecasting and regression tasks, is employed as the key comparison metric to measure the accuracy of predictions. Since MAPE is a scale independent measure, and works well except for zero actual value, in addition to its frequent use in literature for measuring forecasting accuracy (Hyndman, 2006), this metric is used as the accuracy measurement metric for predictions in this study. On the other hand, Mean Squared Error (MSE), which is an intuitive and prevalently used for loss function for regression tasks by quadratically penalizing the errors, is used in the model training stage as the loss function metric for all the methods. MAPE is used to compare the performance of the methods used since MAPE metric is the commonly used load prediction error measurement for the providers. MAPE and MSE computations are performed as per Eqs. (19) and (20), respectively.

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{\hat{y}_t - y_t}{y_t}\right| \times 100 \tag{19}$$

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2 \tag{20}$$

where N corresponds to the size (number) of predictions, y_t and $\hat{y}_t$ correspond to actual value at time step t and predicted value for time step t, respectively.

In all experiments, the first two years of data are used for training, and the last year data is used for testing. 10% of training dataset was

Table 7
Layer settings for the 1-D CNN method.

			Filter size	Kernel size	Dilation rate	Activation(s)
Before concatenation		First CNN	200	4	2	ReLU
		Second CNN	750	3	4	ReLU
	First RMB	1st MU	400	3	1	Sigmoid/ReLU
		2nd MU	400	3	2	Sigmoid/ReLU
After concatenation	Second RMB	1st MU	400	3	4	Sigmoid/ReLU
		2nd MU	400	3	8	Sigmoid/ReLU
	Last MU	1st MU	700	4	16	Sigmoid/ReLU
		Max Pooling	–	6	–	–
		1st Dense Lay.	250	–	–	ReLU
		2nd Dense Lay.	1	–	–	Linear

Table 8
Parameter and hyperparameter settings for the 1-D CNN method.

Batch size	24
Epoch number	60
Learning rate	0.000225
Decay rate	0.005525
rho	0.95
Epsilon	None

Table 9
Mann–Whitney U test results of one-hour-ahead predictions.

Methods	p-value
1-D CNN – Stacked GRU	*0.368*
1-D CNN – Stacked LSTM	*0.942*
1-D CNN – AE-GRU	*0.942*
1-D CNN – AE-LSTM	*0.190*

Table 10
MAPE results of one-hour-ahead predictions for all the methods.

	Stacked LSTM	Enc.-Dec. LSTM	Stacked GRU	Enc.-Dec. GRU	The proposed 1-D CNN
January	0.99997	0.99997	0.99997	0.99997	0.99997
February	1.00002	1.00002	1.00002	1.00002	1.00002
March	1.00012	1.00012	1.00012	1.00012	1.00012
April	1.00033	1.00033	1.00033	1.00033	1.00033
May	1.00054	1.00054	1.00054	1.00054	1.00053
June	1.00055	1.00055	1.00055	1.00054	1.00054
July	1.00022	1.00022	1.00022	1.00022	1.00022
August	1.00017	1.00017	1.00017	1.00017	1.00017
September	1.00044	1.00043	1.00044	1.00043	1.00043
October	1.00037	1.00037	1.00037	1.00037	1.00037
November	1.00013	1.00013	1.00012	1.00012	1.00013
December	1.00003	1.00003	1.00003	1.00003	1.00003
Mean	1.000241	1.000240	1.000240	1.000238	1.000238
Std. Dev.	2.4e−6	2.1e−6	1.1e−6	1.8e−6	1.8e−6

used for validation to optimize hyperparameters of the deep learning method. The prediction problem in this paper was treated and formulated as a multivariate regression type forecasting problem through framing the data into a supervised learning type data structure. Although it was not framed as a classic time-series forecasting problem, to maintain the time dependent specificity the week-of-the-day designations are retained, and then, the training samples are randomly shuffled to achieve a more robust and generalizable multivariate prediction model. Shuffling of data is helped prevent sequence overfitting issue, which is a common problem in time series forecasting problems as well. The neural network parameters are initialized randomly at the beginning of each run, and the network, in turn, produces somehow different results for each run. Hence, to mitigate the effect of randomness and to produce more robust results, we repeated our experiments for ten times using the same training and test sets, and made ensembles of these ten results for each of the method types. In the experiments, the training time for the proposed method was about an hour, and the model building time for the other (comparative methods) was approximately 45 min.

4. Results and discussion

In this section, the monthly MAPE results of each method type for both multi-step-ahead and one-step-ahead predictions which they are important for intraday mechanism and day-ahead mechanism in the market mentioned earlier are presented (see Tables 10 and 12). In addition, preliminary examinations with naïve forecasting method and some machine learning methods for this dataset are performed. In the preliminary examinations, simple naïve forecasting method yielded 6.1% and 9.6% MAPE values for one-hour-ahead and 24-hour-ahead prediction tasks, respectively. Other examinations with shallow neural networks and deep feed-forward neural networks are performed, where their parameters of these networks are determined via a grid search method. With a shallow neural network that has one hidden layer with 50 neurons, 1.24% and 3.14% MAPEs were found for one-hour ahead and 24-hour ahead predictions, respectively. With a deep feed-forward neural network that has two hidden layers with 32 and 64 neurons in the two layers, 2.7% and 3.1% MAPEs were obtained for one-hour ahead and 24-hour ahead predictions, respectively. On the other hand, SVR, Random Forest and Gradient Boosting methods for both one-hour-ahead and 24-hour-ahead predictions are utilized. The best performing method for both prediction settings was SVR which produced 3.2% MAPE for 24-hour-ahead predictions, and 1.66% MAPE for one-hour-ahead predictions. Since the performances of the deep learning methods used are very high compared with the best performing machine learning method used, these machine learning methods to our analyses were not included. Thus, this study is dedicated only to the comparisons of deep learning methods.

Mann–Whitney U test, which is a non-parametric statistical test and has no strict assumptions apriori for datasets, was conducted for both prediction tasks to test whether ten ensemble results for the best performing method significantly better than the rest of the methods as well (see Tables 9 and 11).

When comparing the results for the ten ensembles of one-step-ahead predictions, null hypothesis states that there is no difference between the means of the compared methods. We used 0.01 alpha level for all comparisons to keep the confidence interval of the test very tight.

According to the results in Table 9, there is not enough evidence to reject the null hypothesis so that the ensemble results of the proposed 1-D CNN method do not significantly differ from the rest of the methods for one-step-ahead predictions. This fact is salient in the MAPE results presented in Table 10 as well which the methods nearly produce the same results obtained by the analyses.

From Table 10, differences between predictions of the methods are minuscule so that it can be inferred that the performances of all the methods are nearly the same for one-hour-ahead predictions. This fact

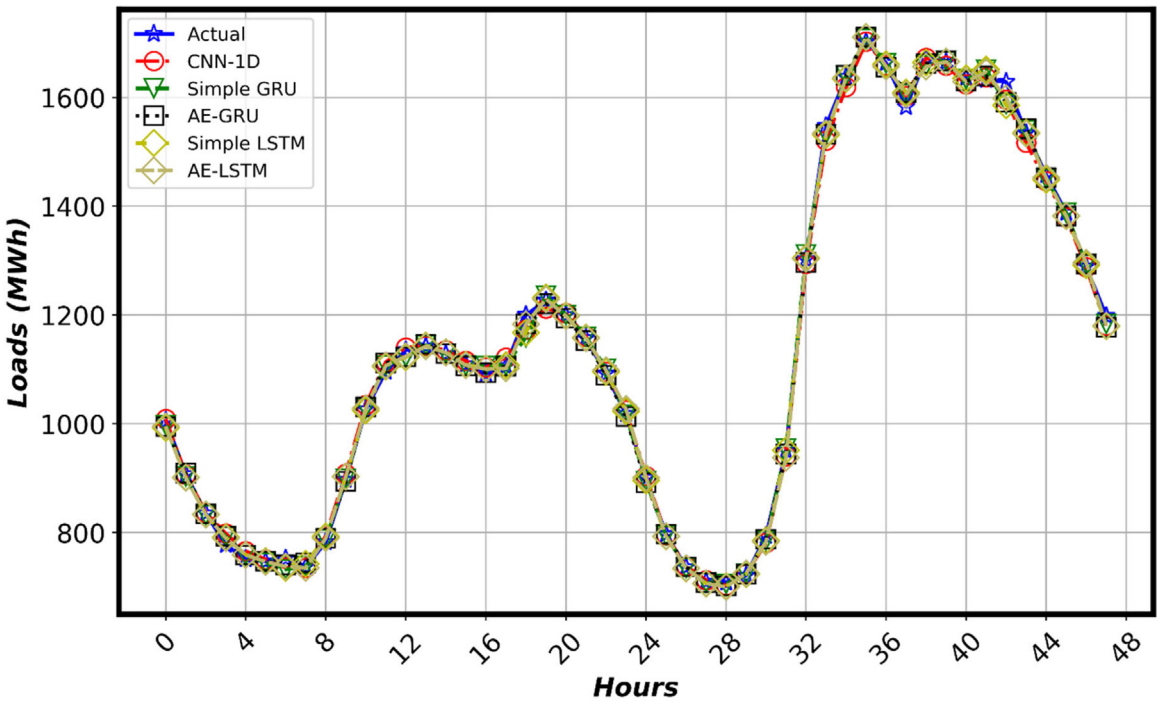


Fig. 15. One-step-ahead predictions for all the methods.

Table 11
Mann Whitney U test results of 24-hour-ahead predictions.

Methods	p-value
1-D CNN – Stacked GRU	*0.00018*
1-D CNN – Stacked LSTM	*0.00018*
1-D CNN – AE-GRU	*0.00025*
1-D CNN – AE-LSTM	*0.00033*

verifies the results of Table 9 which suggest that there is not significant difference between the ensemble averages of the methods. Even the performance of the proposed 1-D CNN method is superior to the rest of the methods, which cannot be seen due to rounding the numbers, it is hard to compare the superiority of the methods among them since all the results are around 1%. When the proposed 1-D CNN is applied without using temperature feature, it produces a 3.03% MAPE error in total. This result shows the marginal contribution of the temperature feature to the prediction outcome.

One day's predictions with respect to actual loads for exemplary visualization are presented in Fig. 15.

The same confidence level that is equal to 0.01 is used for 24-hour-ahead predictions comparisons as well. According to the results in Table 11, there is enough evidence to reject the null hypothesis so that the ensemble results of the proposed 1-D CNN method are significantly better than the rest of the methods for 24-hour-ahead predictions. Thus, the results for 24-hour-ahead predictions show clear performance distinction between the proposed 1-D CNN method and the rest of the methods.

To show that the proposed method cannot suffer from overfitting, loss function vs number of epochs for training and validation sets is given in Fig. 16.

In addition to Fig. 16, MAPE vs number of epochs is given in Fig. 17.

Figs. 16–17 shows that the model used dropout avoided overfitting in the training phase. Consecutively, three days' prediction results of all methods with respect to actual loads for exemplary visualizations are presented in Fig. 18.

According to the results presented in Table 12, the proposed 1-D CNN method dominantly outperformed the remaining deep learning methods. In Table 12, the result of the best performing method in each month is compared with the second-best performing method, and the performance gain (the results of these comparisons) are given in the last column (labeled as Performance Gain). Computations for the performance gain of the proposed 1-D CNN with respect to its successor method is given in Eq. (21).

Table 12
MAPE results for all the methods for 24-hour-ahead predictions.

	Stacked LSTM	Enc.-Dec. LSTM	Stacked GRU	Enc.-Dec. GRU	The proposed 1-D CNN	Performance gain (%)
January	3.06	3.07	*2.65*	3.39	***2.43***	***8.3***
February	2.28	2.16	*2.06*	2.35	***1.74***	***15.5***
March	2.14	2.06	2.07	*2.05*	***1.56***	***23.9***
April	2.01	*1.87*	1.94	1.96	***1.54***	***17.6***
May	2.74	2.68	*2.63*	3.02	***2.55***	***3***
June	4.33	*4.12*	5.36	4.68	***3.95***	***4.1***
July	2.94	*2.70*	2.92	3.56	***2.47***	***8.5***
August	3.02	*2.62*	2.82	3.09	***2.25***	***14.1***
September	4.62	4.66	4.80	*4.50*	***3.88***	***13.8***
October	1.30	*1.13*	1.18	1.18	***1.10***	***2.7***
November	1.72	1.63	*1.36*	1.49	***1.24***	***8.8***
December	2.01	1.89	***1.66***	1.88	*1.79*	*−7.3*
Mean	2.68	*2.55*	2.62	2.76	***2.21***	-
Std. Dev.	0.0016	0.0019	0.0018	0.0028	***0.0005***	

$$PG = \frac{\left(\hat{y}_c - \hat{y}_s\right)}{\hat{y}_s} \times 100 \tag{21}$$

In Eq. (21), PG corresponds to performance gain in percent difference. The $\hat{y}_c$ and $\hat{y}_s$ correspond to the prediction error of the proposed method, and prediction error for the next best performed method for the corresponding month, respectively. Negative sign for the performance gain of a given month indicates how much the proposed method falls behind the next best performing method in that month. On the other hand, positive sign means how much the proposed method goes beyond its next best successor method in the relevant month.

The result of the second best resultant method for each month also was italicized for the convenience of the performance comparisons in

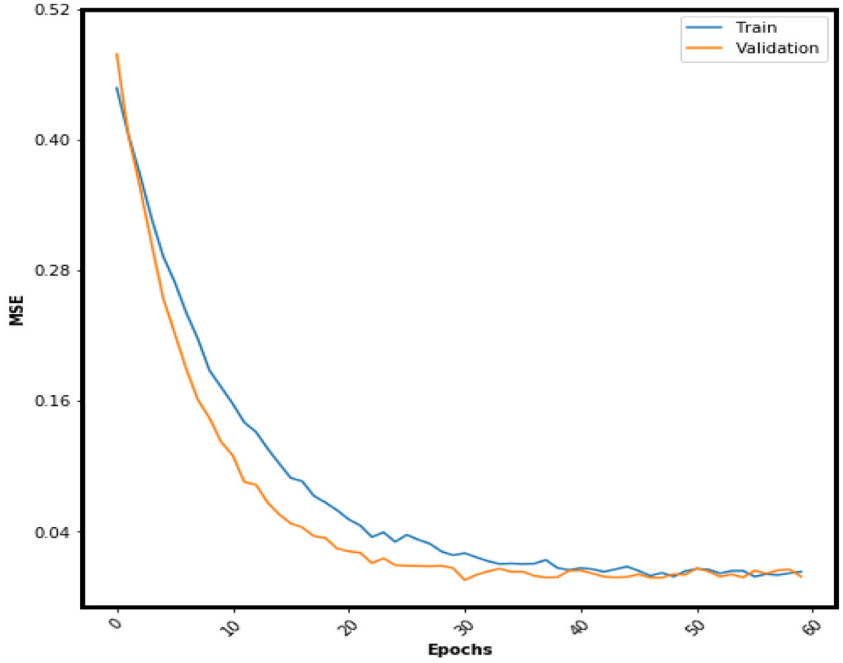


Fig. 16. Loss function vs number of epochs.

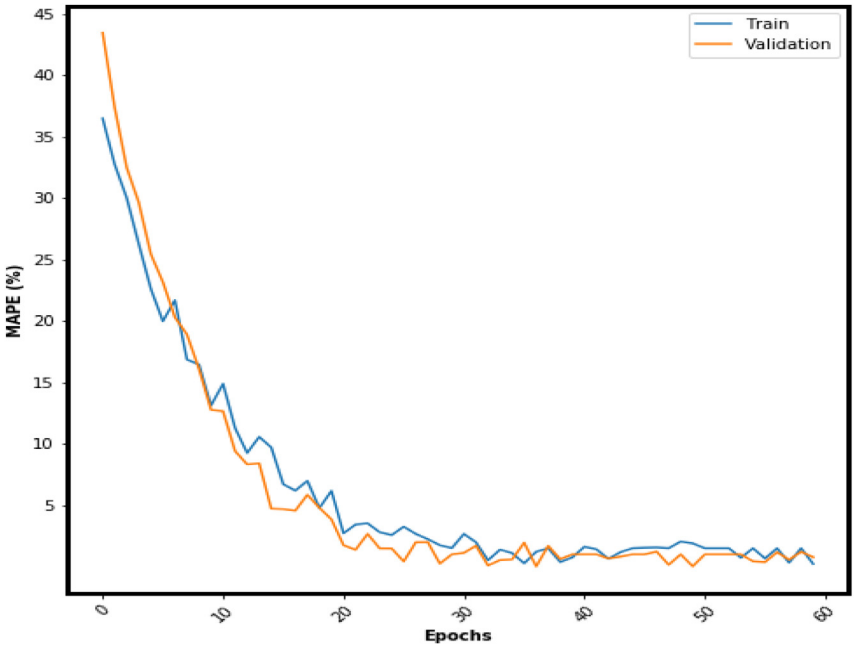


Fig. 17. MAPE vs number of epochs.

Table 12. According to the obtained results in Table 12, the 1-D CNN method, based on VPNs, yields significantly better MAPE results for a one-year prediction, with an average of *2.21*% better than the other methods It is followed by the encoder–decoder LSTM method with an average MAPE *2.55*%. The performance difference between the CNN method and its successor in total is *13.3*%. Hence, the proposed CNN method shows its superiority in predicting the short-term time-series predictions for 24-hour-ahead over the rest of the methods utilized in this study. Another noteworthy importance is that our proposed method outperforms the rest of the methods in each month except for December, as seen in the performance gain column in Table 12. In the month of December, stacked GRU yields the best result among the used methods. According to performance comparisons of the methods for months, the performances for May, June, July, and October appeared to have slight differences within the best performing method (the proposed CNN) and its follower. On the other hand, in the remaining months, with the exclusion of December, the proposed 1-D CNN method shows remarkably high-performance differences over the other methods.

In the analysis, encoder–decoder architecture is included for their efficient representation and learning of time series data. Although encoder–decoder LSTM performs better than stacked LSTM, this fact is not valid for GRU. On the other hand, GRU outperforms the LSTM method with stacked architecture. It can be said that encoder–decoder architecture is the best option within stacked and encoder–decoder architecture; however, the usefulness and superiority of LSTM or GRU

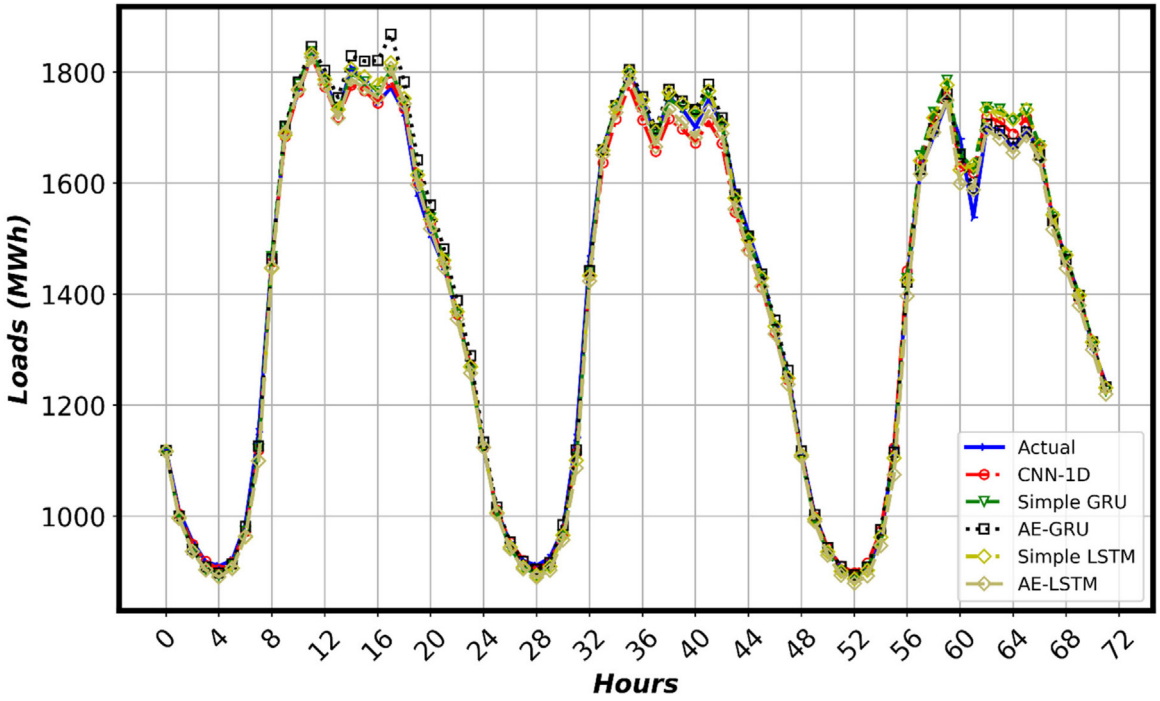


Fig. 18. Multi-step-ahead predictions for all methods for a week.

need to be tested as may change based on the characteristics of the data and the specifics of the architecture employed for a given time-series prediction problem.

According to the results, the proposed 1-D CNN method performed the best for the months of February, March, April, October, November, December, where the MAPE measures are all below 1.8%. On the other hand, the prediction results of the method for January, May, July, and August came out as moderately good when compared to the well-performed months. The results of the method for June and September were the worst of all twelve months used as the test dataset. These facts are in line with and validated by the other methods, as well. The months, where all of the methods performed the worst, may have some common characteristics. For instance, Muslims perform fasting in Ramadan, which is a month followed by a celebration in the Islamic calendar. Coincided with long days and short nights of the summer season, the load consumption in this month shows different characteristics than regular days of the year. The load consumption that is low at night diverges from the consumption of regular days of the year in this month. At the end of this month, a festival is being held which the load consumption also diverges from the consumption of regular days. Some parts of this month fell on the dates in May, and the remaining part fell on dates in June 2017.

Similarly, another religious festival held in September followed just after a 3-day national holiday so that different consumption patterns and characteristics in these rare time-periods may have affected the performance of the methods. These rare time issues would have been captured by the method. However, the Ramadan month's coincidence with the summer season, causing very different consumption characteristics during the month, and the second religious holiday following a national holiday, created a problem for capturing the relationship in these months. This is because of the fact that the dates of these months and rare time issues change every year since Turkey officially uses the Gregorian calendar. Hence, these Islamic calendar months and days fall on different dates in the Gregorian calendar, and each year they show different characteristics by the effect of seasons. Some of the previous studies had faced the same special days problem, and worked to mitigate the issue. Trull et al. (2019) studied Spanish electricity demand during Easter, where there was a rare-time issue. The authors proposed a new point of view to handle multiple seasonality within the Holt-Winters' method by introducing a new seasonality method. The proposed model incorporated Holt-Winters models with discrete-interval moving seasonality since this issue is rare, but the pattern of it repeatedly appears in certain moments of the time series. Therefore, the pattern of this rare time issue is considered discrete, and a model is proposed from this point of view (Trull et al., 2019). Bermúdez (2013) studied Spanish electricity demand forecasting as well. The author handled special days by using covariates in the exponential smoothing methods. In this study, a rare time issue that is Easter is considered as in the previous study for Spanish electricity demand forecasting (Trull et al., 2019). Darbellay and Slama (2000) studied the electricity demand of Wales by considering special days and related calendar effects. Erişen et al. (2017) inserted special days into the prediction model, and obtain promising results with Nonlinear AutoRegressive with Exogenous inputs networks (NARXNet) method. They used hourly electricity load data from the Dutch electricity grid, and proposed that their model can be further extended to achieve more promising results. Similar to the other countries that have different special days such as Turkey, since some characteristics of the special days, including religious and national holidays, they all tend to show common characteristics. However, when their model's suitability to our data is considered, in their dataset, there are no dual-calendar effects.

Another method is proposed by De Livera et al. (2011) that is capable of both modeling linear and nonlinear models having single seasonality, multiple seasonality, and dual calendar effects is the Trigonometric, Box–Cox transform, ARMA errors, Trend and Seasonal components (TBATS) method for this type of problem. TBATS method has superiority over traditional forecasting methods in modeling dual-calendar effects (De Livera et al., 2011). The dual calendar effects problem is the problem that caused a higher prediction error in our study. In the preliminary examinations of the study, we tested these methods, whether they are superior to the used deep learning model or not. First of all, the NARXNet method did not produce promising results compared to the used deep learning models. In addition, it may have a problem with our modeling scheme. We use an autoregressive type of modeling without sequence ordering in our deep learning methods; however, NARXNet is not free of sequence ordering. Hence,

the modeling of NARXNet may suffer sequence overfitting problem in modeling the forecasting problem. When its setting is tuned to shuffle the data in the training phase, it will probably produce the worse results than its non-shuffled counterpart. Shuffling the data, which each predicted time step's input is independent of previous and next step's inputs, unlike time-step based order, enhances the robustness of the performance of the method by breaking the sequence ties in the data; hence, the used deep learning method performed better than its non-shuffled data setting. Secondly, TBATS, which is able to decompose overall seasonal components into several individual components for modeling, was used in the preliminary examinations, and it fell short in modeling dual calendar effects – Hijri (Islamic) and Gregorian calendar issue – with our data when its performance was compared to the deep learning methods in this study. Hence it produced inferior results compared to the used deep learning models.

In addition to these preliminary examinations, the performance of the used deep learning methods by introducing special days into our models is investigated, however, presented settings in section application and settings are found the best input settings scheme for modeling this decision making problem. In future studies, other approaches can be proposed for these rare time issues for better modeling; thus, their side effects on predictions can be reduced. Humidity factor may also have been a negative effect on the high prediction errors along with rare time issues for May, June, July, August, and September because humidity effects combined with high temperatures lead to increased use of air conditioners. Thus, the load consumption patterns in summer months may diverge from the rest of the months, and this fact must be considered as an adjustment to obtain lower prediction errors. Because we did not have the humidity information for the data we could not include it in the analyses. The proposed 1-D CNN method was applied without including the temperature feature, as a result, 3.4% MAPE error was obtained for 24-h ahead predictions. This fact shows the marginal contribution of the temperature feature to the prediction outcome, as was in 1-h ahead prediction task.

5. Summary and conclusions

In this paper, a novel methodology based on VPN method is proposed, and then comparative analyses of the most prevalent deep learning architectures and their variants for the tasks of one-step-ahead and multi-step ahead time series predictions using a real-world application case are performed. The data utilized in this study originally captured as an hourly electricity load and corresponding temperature values for the city of İstanbul between 2015 and 2017. In the employed methodology, first, data consolidation and preprocessing tasks are performed. As a result of these data transformation tasks, an optimal time-lag, which will be used as input time steps for the modeling tasks, is identified. Calendar effects are also considered along with the information obtained from the provider firm, and the inclusion of dummy variables for days is proposed after examining day effect characteristics. The objectives of this study were to present deep learning utilization in the forecasting process for the managerial decision-makers and to provide insights into practitioners for the managerial decision-makers and researchers with a real-world application of the efficacy of deep learning methods in short term load forecasting problem. The newly proposed method performed better than the current deep learning models applied to the data in an inappreciable extent for one-hour-ahead forecasting, and outperformed the current deep learning models with the utilized data for 24-hour-ahead forecasting; hence it may be applicable to other datasets in forecasting the different regions loads in Turkey. Hence, the study contributes to STLF by providing a new decision making assistant.

One dimensional CNNs are rarely used for time series forecasting compared to LSTM and GRU. However, two-dimensional CNNs have proven their efficiency in many tasks such as computer vision, person re-identification, and image classification. In this paper, we proposed a new method with 1-D CNN based on VPNs which there is no existent study in the literature that uses VPNs in time series forecasting domain. Accordingly, the proposed method was compared against the presumed best time-series predicting architectures, namely stacked LSTM and GRU and encoder–decoder LSTM and GRU. According to the obtained results of a time-based holdout dataset, the proposed 1-D CNN method outperformed the rest of the methods for 24-hour-ahead predictions. The proposed method had negligible performance gain for one-hour-ahead predictions compared to the rest of the methods used for the same dataset. The performance of the proposed 1-D CNN method with the rest of the methods used are compared for the tasks of one-hour-ahead and 24-hour-ahead predictions. For this purpose, Man–Whitney U test was conducted to spot whether the performance differences of the ten ensembles for the best performing method and the rest of the deep learning methods are statistically significant for both of these tasks. As the result of the tests, the performance difference of the proposed 1-D CNN with respect to the rest of the deep learning methods was spotted statistically significant at 0.01 confidence level for 24-hour-ahead predictions. On the other hand, there was no enough evidence for performance difference of the proposed 1-D CNN when compared with the rest of the methods for one-hour-ahead predictions which this fact is clearly seen from the results presented in Table 10.

This study offers some contributions to the field of time series forecasting. The study presents a comparative analysis of the off-the-shelf deep learning methods, including stacked, encoder–decoder architecture, and CNN methods, for the underlying prediction task. The methods employed in this study provide an easy understanding and straightforward implementation for both researchers and managerial decision-makers in the forecasting area since they presented as end-to-end learning methods that do not exploit the combined/hybridized network architectures, and they do not get into the fully automated feature extraction procedures outside of the deep neural networks scope. The main takeaways in this study can be summarized as follows:

- Calendar effects must be considered as an integral part for time series modeling since they may contribute significantly to prediction performance,
- Rare time event issues affect prediction performances and must be handled separately and methodologically (perhaps with additional business rules and adjustments),
- Due to the big data phenomenon, newly developed deep learning architectures, and more powerful computational hardware (e.g., GPUs), increased attention is being paid to the deep learning methods and their use in time series forecasting to obtain higher performance when compared to the traditional statistical and machine learning methods,
- Rarely used 1-D CNNs and their variants, like the one proposed in the current study, can be used for time series and can produce promising (and perhaps superior) results,
- CNN's spatio-temporal feature extraction ability and auto-capturing of the correlations in long sequences make it a standing rival to the well known time series prediction network architectures like LSTM and GRU,
- The 1-D CNN based on VPNs method proposed and employed in this study provides the means for exogenous feature inclusion as inputs and thereby extends the preliminary convolution operation with a larger time–frequency resolution to capture high-level/global features in the sequence. Hence, this may boost the method's performance,

In summary, this study aims to handle point estimation rather than confidence interval estimation for the underlying time series forecasting. For further studies, the method's ability can be extended to include interval estimations. Since the proposed method (1-D CNN based on VPNs) performed very well for time series prediction problem using the CNN's feature extraction ability and capturing the correlations in long sequences, the method can be extended to be used for other time

series problems such as time series classification and human activity recognition tasks. In addition, applying the developed methodology to electricity load forecasting data of other regions or countries can demonstrate its utility as a general decision-making tool for forecasting.

Declaration of competing interest

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

Appendix

Acronyms

CNN	Convolutional Neural Network
LSTM	Long-Short Term Memory
GRU	Gated Recurrent Unit
VPN	Video Pixel Network
VSTLF	Very Short-Term Load Forecasting
MTLF	Medium-Term Load Forecasting
LTLF	Long-Term Load Forecasting
STLF	Short-Term Load Forecasting
EPİAŞ	Elektrik Piyasasıİşletmeleri Anonim Şirketi
DG	Distributed Grid
ANN	Artificial Neural Network
WT	Wavelet Transform
SVR	Support Vector Regression
PSO	Particle Swarm Optimization
ARIMA	Autoregressive Integrated Moving Average
ELM	Extreme Learning Machine
MLP	Multiple Layer Perceptron
MLR	Multiple Linear Regression
EMD	Empirical Mode Decomposition
EKF	Extended Kalman Filter
AE	Auto Encoder
RNN	Recurrent Neural Network
RBM	Restricted Boltzman Machine
DBN	Deep Belief Network
DBM	Deep Boltzman Machine
GRNN	Gated Recurrent Neural Network
GCNN	Gated Convolutional Neural Network
FCRBM	Factorized Conditional Restricted Boltzman Machine
SARIMAX	Seasonal Autoregressive Integrated Moving Average with External Variable
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DNN	Deep Neural Network
TD-LSTM	Time Dependent Long-Short Term Memory
C-LSTM	Cycle Long-Short Term Memory
AEMO	Australian Electricity Market Operator
LP	Linear Predictor
DRNN-GRU	Deep Recurrent Neural Network Gated Recurrent Unit
MAPE	Mean Absolute Percentage Error
RICNN	Recurrent Inceptional Convolutional Neural Network
FCBM	Factored Conditional Boltzman Machine
PACF	Partial Autocorrelation Function
TCN	Temporal Convolutional Network
MU	Multiplicative Unit
RMB	Residual Multiplicative Block
ReLU	Rectified Linear Unit
MIMO	Multi-Input Multi-Output
MSE	Mean Square Error
PG	Performance Gain
NARX	Nonlinear Autoregression with External Variable
TBATS	Trigonometric seasonality Box–Cox transformation ARMA errors Trend and Seasonal components
GPU	Graphical Process Unit

Notations

x_i'	Normalized value of an example (ith example) in a dataset
x_i	Unnormalized value of an example (ith example) in a dataset
μ	Mean value of a dataset
σ_d	Standard Deviation of a dataset
h_t	Hidden state of a GRU/LSTM
w_i	Input weights
x_t	Input value for time step t
w_h	Hidden state weights
h_{t-1}	Previous hidden state
f	Arbitrary activation function
f_t	Forget gate
σ	Sigmoid activation function
W_f	Weights for forget gate
b_f, b_i, b_c	Bias term for relevant gates
c_{t-1}	Cell state at time step t-1
i_t	Input gate for GRU/LSTM
$\check{C}_t$	Candidate values for next states in GRU/LSTM
T_h	Hyperbolic tangent activation function
o_t	Output gate
W_o	Weights for output gate
b_o	Bias term for output gate
z_t	Update gate for GRU
r_t	Reset gate for GRU
W_z, W_r	Weights for update gates of GRU
$\check{h}_t$	Current memory content for GRU
$g_{(i)}$	Convolution operation output
$h\,(i)$	Causal convolution operation output
g_1, g_2, g_3	Relevant gates for a Multiplicative Unit
u	Update gate for a Multiplicative Unit
$D_{i,j}$	Dummy variables for j^{th} day of i^{th} time step
T_i	Temperature value of i^{th} time step
L_i	Load value of i^{th} time step
hp	Hourly periodical cycle
c_d	Daily hour cycle
$\hat{y}_t$	Predicted value at time step t
y_t	Actual value at time step t
N	Number of time steps in a dataset
$\hat{y}_c$	Prediction error of the proposed method in a month
$\hat{y}_s$	Prediction of the first/second method in a month

References

Almalaq, A., Edwards, G., 2017. A review of deep learning methods applied on load

forecasting. In: 2017 16th IEEE International Conference on Machine Learning and Applications. ICMLA, IEEE, pp. 511–516.
Amato, U., Antoniadis, A., De Feis, I., Goude, Y., Lagache, A., 2020. Forecasting high resolution electricity demand data with additive models including smooth and jagged components. Int. J. Forecast..
Amjady, N., Keynia, F., 2009. Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm. Energy 34, 46–57.
Amjady, N., Keynia, F., 2011. A new neural network approach to short term load forecasting of electrical power systems. Energies 4, 488–503.
Amral, N., Ozveren, C.S., King, D., 2007. Short term load forecasting using multiple linear regression. In: 2007 42nd International Universities Power Engineering Conference. IEEE, pp. 1192–1198.
Bahrami, S., Hooshmand, R.-A., Parastegari, M., 2014. Short term electric load forecasting by wavelet transform and grey model improved by PSO (particle swarm optimization) algorithm. Energy 72, 434–442.
Bai, S., Kolter, J.Z., Koltun, V., 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv Prepr. arXiv:1803.01271.
Bedi, J., Toshniwal, D., 2019. Deep learning framework to forecast electricity demand. Appl. Energy 238, 1312–1326.
Berk, K., Hoffmann, A., Müller, A., 2018. Probabilistic forecasting of industrial electricity load with regime switching behavior. Int. J. Forecast. 34, 147–162.
Bermúdez, J.D., 2013. Exponential smoothing with covariates applied to electricity demand forecast. Eur. J. Ind. Eng. 7, 333–349.
Binkowski, M., Marti, G., Donnat, P., 2018. Autoregressive convolutional neural networks for asynchronous time series. In: International Conference on Machine Learning. PMLR, pp. 580–589.
Cai, M., Pipattanasomporn, M., Rahman, S., 2019. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. Appl. Energy 236, 1078–1088.
Cancelo, J.R., Espasa, A., Grafe, R., 2008. Forecasting the electricity load from one day to one week ahead for the Spanish system operator. Int. J. Forecast. 24, 588–602.
Ceperic, E., Ceperic, V., Baric, A., 2013. A strategy for short-term load forecasting by support vector regression machines. IEEE Trans. Power Syst. 28, 4356–4364.
Chen, Kunjin, Chen, Kunlong, Wang, Q., He, Z., Hu, J., He, J., 2018. Short-term load forecasting with deep residual networks. IEEE Trans. Smart Grid.
Chen, Y., Luh, P.B., Guan, C., Zhao, Y., Michel, L.D., Coolbeth, M.A., Friedland, P.B., Rourke, S.J., 2009. Short-term load forecasting: Similar day-based wavelet neural networks. IEEE Trans. Power Syst. 25, 322–330.
Cheng, L., Yu, T., 2019. A new generation of AI: A review and perspective on machine learning technologies applied to smart energy and electric power systems. Int. J. Energy Res. 43, 1928–1973.
Chitalia, G., Pipattanasomporn, M., Garg, V., Rahman, S., 2020. Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks. Appl. Energy 278, 115410.
Cho, K., Merriënboer, B.Van., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. arXiv Prepr. arXiv:1406.1078.
Choi, H., Ryu, S., Kim, H., 2018. Short-term load forecasting based on ResNet and LSTM. In: 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids. SmartGridComm, IEEE, pp. 1–6.
Darbellay, G.A., Slama, M., 2000. Forecasting the short-term demand for electricity: Do neural networks stand a better chance? Int. J. Forecast. 16, 71–83.
De Livera, A.M., Hyndman, R.J., Snyder, R.D., 2011. Forecasting time series with complex seasonal patterns using exponential smoothing. J. Amer. Statist. Assoc. 106, 1513–1527.
Din, G.M.U., Marnerides, A.K., 2017. Short term power load forecasting using deep neural networks. In: 2017 International Conference on Computing, Networking and Communications. ICNC, IEEE, pp. 594–598.
Djukanovic, M., Ruzic, S., Babic, B., Sobajic, D.J., Pao, Y.H., 1995. A neural-net based short term load forecasting using moving window procedure. Int. J. Electr. Power Energy Syst. 17, 391–397.
Dong, Y., 2019. Implementing deep learning for comprehensive aircraft icing and actuator/sensor fault detection/identification. Eng. Appl. Artif. Intell. 83, 28–44.
Dong, X., Qian, L., Huang, L., 2017a. Short-term load forecasting in smart grid: A combined CNN and K-means clustering approach. In: 2017 IEEE International Conference on Big Data and Smart Computing. BigComp, IEEE, pp. 119–125.
Dong, X., Qian, L., Huang, L., 2017b. A CNN based bagging learning approach to short-term load forecasting in smart grid. In: 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation. IEEE, pp. 1–6.
Dordonnat, V., Pichavant, A., Pierrot, A., 2016. GEFCom2014 probabilistic electric load forecasting using time series and semi-parametric regression models. Int. J. Forecast. 32, 1005–1011.
Du, S., Li, T., Yang, Y., Horng, S.-J., 2020. Multivariate time series forecasting via attention-based encoder-decoder framework. Neurocomputing.
Duan, Q., Liu, J., Zhao, D., 2017. Short term electric load forecasting using an automated system of model choice. Int. J. Electr. Power Energy Syst. 91, 92–100.
Dudek, G., 2015. Short-term load forecasting using random forests. In: Intelligent Systems' 2014. Springer, pp. 821–828.
Erişen, E., Iyigun, C., Tanrısever, F., 2017. Short-term electricity load forecasting with special days: an analysis on parametric and non-parametric methods. Ann. Oper. Res. 1–34.
Fan, C., Ding, C., Zheng, J., Xiao, L., Ai, Z., 2020. Empirical mode decomposition based multi-objective deep belief network for short-term power load forecasting. Neurocomputing 388, 110–123.
Fan, S., Hyndman, R.J., 2012. Short-term load forecasting based on a semi-parametric additive model. IEEE Trans. Power Syst. 27, 134–141.
Fan, C., Wang, J., Gang, W., Li, S., 2019. Assessment of deep recurrent neural network-based strategies for short-term building energy predictions. Appl. Energy 236, 700–710.
Fang, X., Yuan, Z., 2019. Performance enhancing techniques for deep learning models in time series forecasting. Eng. Appl. Artif. Intell. 85, 533–542.
Gasparin, A., Lukovic, S., Alippi, C., 2019. Deep learning for time series forecasting: The electric load case. arXiv Prepr. arXiv:1907.09207.
Gil-Martín, M., San-Segundo, R., Fernández-Martínez, F., Ferreiros-López, J., 2020. Improving physical activity recognition using a new deep learning architecture and post-processing techniques. Eng. Appl. Artif. Intell. 92, 103679.
Guan, C., Luh, P.B., Michel, L.D., Wang, Y., Friedland, P.B., 2013. Very short-term load forecasting: wavelet neural networks with data pre-filtering. IEEE Trans. Power Syst. 28, 30–41.
Guo, Z., Zhou, K., Zhang, X., Yang, S., 2018. A deep learning model for short-term power load and probability density forecasting. Energy 160, 1186–1200.
Guyot, D., Giraud, F., Simon, F., Corgier, D., Marvillet, C., Tremeac, B., 2019. Overview of the use of artificial neural networks for energy-related applications in the building sector. Int. J. Energy Res. 43, 6680–6720.
Hafeez, G., Alimgeer, K.S., Khan, I., 2020. Electric load forecasting based on deep learning and optimized by heuristic algorithm in smart grid. Appl. Energy 269, 114915.
Hajirahimi, Z., Khashei, M., 2019. Hybrid structures in time series modeling and forecasting: A review. Eng. Appl. Artif. Intell. 86, 83–106.
Han, L., Peng, Y., Li, Y., Yong, B., Zhou, Q., Shu, L., 2019. Enhanced deep networks for short-term and medium-term load forecasting. IEEE Access 7, 4045–4055.
He, W., 2017. Load forecasting via deep neural networks. Procedia Comput. Sci. 122, 308–314.
He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.
Hernández, L., Baladrón, C., Aguiar, J.M., Carro, B., Sánchez-Esguevillas, A., Lloret, J., 2014. Artificial neural networks for short-term load forecasting in microgrids environment. Energy 75, 252–264.
Hewamalage, H., Bergmeir, C., Bandara, K., 2020. Recurrent neural networks for time series forecasting: Current status and future directions. Int. J. Forecast..
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780.
Hong, T., Wang, P., 2014. Fuzzy interaction regression for short term load forecasting. Fuzzy Optim. Decis. Mak. 13, 91–103.
Hooshmand, R.-A., Amooshahi, H., Parastegari, M., 2013. A hybrid intelligent algorithm based short-term load forecasting approach. Int. J. Electr. Power Energy Syst. 45, 313–324.
Hu, R., Wen, S., Zeng, Z., Huang, T., 2017. A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm. Neurocomputing 221, 24–31.
Hui, X., Qun, W., Yao, L., Yingbin, Z., Lei, S., Zhisheng, Z., 2017. Short-term load forecasting model based on deep neural network. In: 2017 2nd International Conference on Power and Renewable Energy. ICPRE, IEEE, pp. 589–591.
Hyndman, R.J., 2006. Another look at forecast-accuracy metrics for intermittent demand. Foresight Int. J. Appl. Forecast. 4, 43–46.
Javidi, M., Jampour, M., 2020. A deep learning framework for text-independent writer identification. Eng. Appl. Artif. Intell. 95, 103912.
Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., Kavukcuoglu, K., 2017. Video pixel networks. In: Proceedings of the 34th International Conference on Machine Learning. vol. 70, JMLR. org, pp. 1771–1779.
Kara, E., Traquair, M., Simsek, M., Kantarci, B., Khan, S., 2020. Holistic design for deep learning-based discovery of tabular structures in datasheet images. Eng. Appl. Artif. Intell. 90, 103551.
Kavousi-Fard, A., Kavousi-Fard, F., 2013. A new hybrid correction method for short-term load forecasting based on ARIMA, SVR and CSA. J. Exp. Theor. Artif. Intell. 25, 559–574.
Kavousi-Fard, A., Samet, H., Marzbani, F., 2014. A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting. Expert Syst. Appl. 41, 6047–6056.
Kim, J., Moon, J., Hwang, E., Kang, P., 2019. Recurrent inception convolution neural network for multi short-term load forecasting. Energy Build. 194, 328–341.
Kong, W., Dong, Z.Y., Jia, Y., Hill, D.J., Xu, Y., Zhang, Y., 2017. Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans. Smart Grid.

Kumar, S., Hussain, L., Banarjee, S., Reza, M., 2018. Energy load forecasting using deep learning approach-LSTM and GRU in spark cluster. In: 2018 Fifth International Conference on Emerging Applications of Information Technology. EAIT, IEEE, pp. 1–4.

Lee, D., Huang, H., Lee, W., Liu, Y., 2020. Artificial intelligence implementation framework development for building energy saving. Int. J. Energy Res..

Lee, C.-M., Ko, C.-N., 2011. Short-term load forecasting using lifting scheme and ARIMA models. Expert Syst. Appl. 38, 5902–5911.

Li, N., Wang, L., Li, X., Zhu, Q., 2020. An effective deep learning neural network model for short-term load forecasting. Concurr. Comput. Pract. Exp. 32, e5595.

Liu, N., Tang, Q., Zhang, J., Fan, W., Liu, J., 2014. A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids. Appl. Energy 129, 336–345.

Liu, X., Zhang, Z., Song, Z., 2020. A comparative study of the data-driven day-ahead hourly provincial load forecasting methods: From classical data mining to deep learning. Renew. Sustain. Energy Rev. 119, 109632.

Memarzadeh, G., Keynia, F., 2021. Short-term electricity load and price forecasting by a new optimal LSTM-NN based prediction algorithm. Electr. Power Syst. Res. 192, 106995. http://dx.doi.org/10.1016/j.epsr.2020.106995.

Mishra, M., Nayak, J., Naik, B., Abraham, A., 2020. Deep learning in electrical utility industry: A comprehensive review of a decade of research. Eng. Appl. Artif. Intell. 96, 104000.

Mocanu, E., Nguyen, P.H., Gibescu, M., Kling, W.L., 2016. Deep learning for estimating building energy consumption. Sustain. Energy Grids Netw. 6, 91–99.

Muzaffar, S., Afshari, A., 2019. Short-term load forecasts using LSTM networks. Energy Procedia 158, 2922–2927.

Narayan, A., Hipel, K.W., 2017. Long short term memory networks for short-term electric load forecasting. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics. SMC, IEEE, pp. 2573–2578.

Osman, Z.H., Awad, M.L., Mahmoud, T.K., 2009. Neural network based approach for short-term load forecasting. In: 2009 IEEE/PES Power Systems Conference and Exposition. IEEE, pp. 1–8.

Ozcanli, A.K., Yaprakdal, F., Baysal, M., 2020. Deep learning methods and applications for electrical power systems: A comprehensive review. Int. J. Energy Res.

Pang, S., del Coz, J.J., Yu, Z., Luaces, O., Díez, J., 2017. Deep learning to frame objects for visual target tracking. Eng. Appl. Artif. Intell. 65, 406–420.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318.

Qiu, X., Ren, Y., Suganthan, P.N., Amaratunga, G.A.J., 2017. Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. Appl. Soft Comput. 54, 246–255.

Sadaei, H.J., e Silva, P.C. de L., Guimarães, F.G., Lee, M.H., 2019. Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series. Energy 175, 365–377.

Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T., 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. Int. J. Forecast. 36, 1181–1191.

Santos, P.J., Martins, A.G., Pires, A.J., 2007. Designing the input vector to ANN-based models for short-term load forecast in electricity distribution systems. Int. J. Electr. Power Energy Syst. 29, 338–347.

Selakov, A., Cvijetinović, D., Milović, L., Mellon, S., Bekut, D., 2014. Hybrid PSO–SVM method for short-term load forecasting during periods with significant temperature variations in city of burbank. Appl. Soft Comput. 16, 80–88.

Shi, H., Xu, M., Ma, Q., Zhang, C., Li, R., Li, F., 2017. A whole system assessment of novel deep learning approach on short-term load forecasting. Energy Procedia 142, 2791–2796.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv Prepr. arXiv:1409.1556.

Skomski, E., Lee, J.-Y., Kim, W., Chandan, V., Katipamula, S., Hutchinson, B., 2020. Sequence-to-sequence neural networks for short-term electrical load forecasting in commercial office buildings. Energy Build. 226, 110350.

Soares, L.J., Medeiros, M.C., 2008. Modeling and forecasting short-term electricity load: A comparison of methods with an application to Brazilian data. Int. J. Forecast. 24, 630–644.

Sudheer, G., Suseelatha, A., 2015. Short term load forecasting using wavelet transform combined with Holt–winters and weighted nearest neighbor models. Int. J. Electr. Power Energy Syst. 64, 340–346.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9.

Tong, C., Li, J., Lang, C., Kong, F., Niu, J., Rodrigues, J.J.P.C., 2018. An efficient deep model for day-ahead electricity load forecasting with stacked denoising auto-encoders. J. Parallel Distrib. Comput. 117, 267–273.

Torres, J.F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., Troncoso, A., 2021. Deep learning for time series forecasting: A survey. Big Data 9, 3–21.

Trull, Ó., García-Díaz, J.C., Troncoso, A., 2019. Application of discrete-interval moving seasonalities to Spanish electricity demand forecasting during Easter. Energies 12 (1083).

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A.W., Kavukcuoglu, K., 2016. WaveNet: A generative model for raw audio. SSW 125.

Wang, Shouxiang, Wang, X., Wang, Shaomin, Wang, D., 2019. Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting. Int. J. Electr. Power Energy Syst. 109, 470–479.

Wen, L., Zhou, K., Yang, S., 2020. Load demand forecasting of residential buildings using a deep learning model. Electr. Power Syst. Res. 179, 106073.

Xiuyun, G., Ying, W., Yang, G., Chengzhi, S., Wen, X., Yimiao, Y., 2018. Short-term load forecasting model of GRU network based on deep learning framework. In: 2018 2nd IEEE Conference on Energy Internet and Energy System Integration. EI2, IEEE, pp. 1–4.

Xu, Q., Yang, X., Huang, X., 2020. Ensemble residual networks for short-term load forecasting. IEEE Access 8, 64750–64759.

Yang, W., Dong, Y., Du, Q., Qiang, Y., Wu, K., Zhao, J., Yang, X., Zia, M.B., 2021. Integrate domain knowledge in training multi-task cascade deep learning model for benign–malignant thyroid nodule classification on ultrasound images. Eng. Appl. Artif. Intell. 98, 104064. http://dx.doi.org/10.1016/j.engappai.2020.104064.

Yang, Y., Hong, W., Li, S., 2019. Deep ensemble learning based probabilistic load forecasting in smart grids. Energy 189, 116324.

Yu, Y., Ji, T.Y., Li, M.S., Wu, Q.H., 2018. Short-term load forecasting using deep belief network with empirical mode decomposition and local predictor. In: 2018 IEEE Power & Energy Society General Meeting. PESGM, IEEE, pp. 1–5.

Zeng, N., Zhang, H., Liu, W., Liang, J., Alsaadi, F.E., 2017. A switching delayed PSO optimized extreme learning machine for short-term load forecasting. Neurocomputing 240, 175–182.

Zhang, R., Dong, Z.Y., Xu, Y., Meng, K., Wong, K.P., 2013. Short-term load forecasting of Australian National Electricity Market by an ensemble model of extreme learning machine. IET Gener. Transm. Distrib. 7, 391–397.

Zheng, J., Chen, X., Yu, K., Gan, L., Wang, Y., Wang, K., 2018. Short-term power load forecasting of residential community based on GRU neural network. In: 2018 International Conference on Power System Technology. POWERCON, IEEE, pp. 4862–4868.

Zheng, J., Xu, C., Zhang, Z., Li, X., 2017. Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network. In: 2017 51st Annual Conference on Information Sciences and Systems. CISS, IEEE, pp. 1–6.