Electric Power Systems Research 178 (2020) 106025

Contents lists available at ScienceDirect

Electric Power Systems Research

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

A novel fuzzy-based ensemble model for load forecasting using hybrid deep neural networks

George Sideratos[a,b,*], Andreas Ikonomopoulos[a], Nikos D. Hatziargyriou[b]

[a] Institute of Nuclear & Radiological Sciences & Technology, Energy & Safety, National Center for Scientific Research "DEMOKRITOS", P.O. Box 60037, 15310, Agia Paraskevi, Greece

[b] National Technical University of Athens, 15773, Zografou, Attiki, Greece

ARTICLE INFO

Keywords:
Load forecasting
Fuzzy clustering
Ensemble prediction
Radial basis function neural networks
Convolutional neural networks

ABSTRACT

A novel, hybrid structure for week-ahead load forecasting is presented. It is the energy market evolution that compels its participants to require load predictions whose accuracy cannot be provided by traditional means. The proposed implementation combines attributes from ensemble forecasting, artificial neural networks and deep learning architectures. The proposed model initially clusters the input data using a novel fuzzy clustering method for creating an ensemble prediction. For each cluster created, a new regression approach is applied to model locally the load forecasting problem. Following a two-stage approach, initially, a radial basis function neural network (RBFNN) is trained using three-fold cross-validation and the hidden layers of the best three RBFNNs are used to transform the input data to a four dimensional dataset. Then, a convolutional neural network (CNN) is deployed receiving as input the latter dataset. Thus, a neural network is formed consisting of a radial basis function (RBF), a convolutional, a pooling and two fully-connected layers. Both RBFNNs and CNNs are trained with the Adam optimization algorithm within the Tensorflow deep learning framework. The proposed model is designed to predict the hourly load for the next seven days and its effectiveness is evaluated in two different case studies; namely the Hellenic interconnected power system and the isolated power system of Crete. Both case-studies exhibit the superior performance of the proposed model when compared to state-of-the-art and traditional load forecasting schemes.

1. Introduction

Load forecasting is an essential tool in various aspects of the energy sector. The load time series are influenced by many factors that include weather conditions, socioeconomic aspects and random effects [1]. The non-linear and non-stationary characteristics of the load timeseries as well as the high performance required by the industry render short-term load forecasting a very difficult task.

Weekly load forecasts are important for a reliable and economic operation of a power system [2]. An over-estimated load prediction obligates Transmission System Operators (TSOs) to commit more expensive generation units leading to higher prices in the energy market. On the other hand, under-estimated load demand can put the safe operation in danger and impose the startup of expensive units with short synch time in real-time. Furthermore, the reliability of power systems is ensured with network analysis based on load predictions. In an open-market environment, the establishment of the marginal spot price and the energy trading are strongly influenced by the load demand. Accurate load forecasts are crucial to guarantee energy transactions and market shares to players in a competitive electricity market. For this reason, TSOs usually pose penalties to energy suppliers with high load prediction errors [2].

Several researches have focused on increasing the performance of load forecasting models [3,4]. Advanced methods based on machine learning techniques along with fuzzy approaches and hybrid implementations [3,5–29] have been applied in short-term load forecasting. Support vector machines (SVMs) [10–14] and RBFNNs [20–24] have been primarily used for load forecasting during the last years. Better performance has been achieved by combining different methods of computational intelligence such as data clustering with unsupervised and supervised neural network training [4].

Some indicative examples of such methods are briefly discussed next. In Ref. [5], the load recorded by several residential and industrial smart meters is predicted using a low-rank output kernel learning method which is a multi-task regression approach. In Ref. [6], additive models are utilized where Group-LASSO is employed for feature selection and P-splines for estimating the additive models. In Refs. [8,9], the performance of generalized linear models, multi-layer neural

* Corresponding author at: National Technical University of Athens, 15773, Zografou, Attiki, Greece.
E-mail address: joesider@power.ece.ntua.gr (G. Sideratos).

https://doi.org/10.1016/j.epsr.2019.106025
Received 17 May 2019; Received in revised form 31 July 2019; Accepted 4 September 2019
Available online 13 September 2019

networks, Gaussian processes and random forests are compared when applied to a combination of four expert designs. In Ref. [11], empirical mode decomposition (EMD) is combined with recursive support vector regressors (rSVRs), where the signals produced by EMD are first classified as principal and behavioral and then rSVRs are applied to the most informative signals producing an ensemble prediction. In Ref. [15], an interval type-2 fuzzy logic system optimized by a neural network-based type reduction algorithm is proposed. In Ref. [20], RBFNN and neuro-fuzzy networks (ANFIS) are employed. RBFNN parameters are tuned by a genetic algorithm using a non-symmetric penalty function as a metric, while an ANFIS model uses the weekly load and temperature variations to improve the prediction accuracy. In Ref. [21], a self-organizing fuzzy neural network is proposed to solve the load forecasting problem using the bi-level optimization algorithm to optimize its hyper-parameters.

It appears [1,3–4] that RBFNNs have exhibited remarkable performance when applied to load forecasting. However, their predictability is mainly dependent on the widths of the radial basis functions at the hidden layer [30–31]. Moreover, they suffer from the 'curse of dimensionality' [32], especially in high-dimensional problems such as load forecasting. The curse of dimensionality, common in all kernel methods, is due to the aggregation of the kernel function units. In order to alleviate the RBFNN drawbacks, a hybrid radial basis function neural network combined with deep learning is proposed. Deep learning (DL) [33] has been studied in many disciplines that include automation processes such as automated labor, picture and audio detection, decision making in critical fields, etc. The DL architecture is more complex than other neural networks as it has more layers and computations [33] and CNN is one of the DL developments. Load forecasting is one of the applications that has benefited from DL algorithms [34]. In Refs. [35–37], several applications of long-short memory recurrent neural networks have shown promising results. In Ref. [38], a 3-filter CNN is used to receive inputs in the form of a 2-dimensional image, while different models are kept during training creating a predictor ensemble. In Ref. [39], the input data is transformed to a graphical representation and a CNN is used to extract the main features. In Ref. [40], the load data is grouped in several clusters using the k-means algorithm and, for each cluster, a CNN is used for prediction making.

The idea underlying this work is to create an ensemble prediction using multiple local regressors. The regressors are activated to produce forecasts by a data clustering method that assigns an input to multiple clusters. The proposed implementation is characterized by novelty that stems from the introduction of (i) a new fuzzy clustering algorithm that classifies to more than one clusters an input vector in order to create ensemble predictions and (ii) an innovative neural network architecture composed by an RBF, a convolutional, a pooling and two fully connected layers (identified as Fuzzy-RBF-CNN). The proposed model uses fuzzy clustering to divide the input space to overlapping areas, such as an input vector to belong to multiple groups. The created input subsets are utilized to develop an ensemble of RBF-CNN regressors. Each RBF-CNN regressor is trained in two phases using the Adam (Additive moment) optimization algorithm [41] employed in a deep learning framework [42]. The RBF centers and widths of a RBF-CNN regressor are first optimized applying a RBFNN training procedure. The corresponding CNN receives the above RBFNN hidden layer outputs and performs an individual load prediction. This way, an ensemble of load predictions is created and the final prediction is computed as an ensemble average.

2. Description of the proposed model structure

The Fuzzy-RBF-CNN model consists of three layers, namely: a fuzzy clustering, an RBF-CNN regressor and an aggregation layers. At the first layer, a novel clustering method groups the input data into multiple clusters. The fuzzy clustering method forms clusters that have to share a portion of their space with their neighboring clusters in order every input to be able to activate more than one clusters and an ensemble prediction to be created at the second layer. Each cluster created corresponds to a fuzzy rule of the fuzzy clustering layer. The proposed clustering method requires each variable to be represented by a set of membership functions. The number of membership functions depends on the variable importance to the problem [43] and the desired number of clusters that will split the input dataset.

Following the fuzzy clustering method, the inputs are distributed to the second layer where an RBF-CNN regressor is applied to each cluster. An RBF-CNN regressor receives the data subset constructed by the corresponding cluster (fuzzy rule) at the first layer. Training of an RBF-CNN regressor requires that the optimal parameters of the RBF kernels, namely the kernel number, centers and widths, are estimated through a procedure typical to RBFNN training. The optimized RBF kernels transform the input data to a higher dimensional space with their activations becoming new data representations. The CNN is trained using the transformed input data at a second stage that analyzes in more detail than an RBFNN the relations of a kernel element with its neighbors; namely with the elements that correspond to a different input variable or kernel. Following the above two-stage training procedure, an RBF-CNN regressor operates as a compact neural network that consists of an RBF, a convolutional, an averaging pooling and two fully-connected layers.

The final prediction of the proposed model will be provided at the third layer by averaging the ensemble predictions of the RBF-CNN regressors that correspond to the clusters activated in the fuzzy clustering layer. Fig. 1 shows the proposed model structure, where three clusters (darker shade) are activated and contribute to the final load prediction. In this example, input x_i activates fuzzy rules 2, 3 and 5 and the RBF-CNNs which are connected to these rules provide an independent prediction. The final prediction is obtained by averaging these predictions.

3. Model training

3.1. The fuzzy clustering method

The first Fuzzy-RBF-CNN layer is designed to cluster the input dataset according to the variables containing the most useful information, so that each input vector will belong to more than two groups. Thus, the most highly-correlated to the output variables define the shape and number of the first layer clusters. In load forecasting, the most valuable input variables to a model are the latest load observation, maximum daily temperature and calendar data – hour, month and special day index – of the prediction time. The training process of the Fuzzy-RBF-CNN first layer requires those variables to be represented with fuzzy membership functions. The value range of each variable is divided into fuzzy sets having common characteristics. Each fuzzy set has a linguistic representation and is described by a membership function. For example, regarding the 'hour' variable multiple fuzzy sets are used during daylight hours, while night hours – when the load timeseries has low variance – belong to one fuzzy set.

Gaussian membership functions are used for the most important continuous input variables, while trapezoid membership functions are employed for representing categorical variables (hour, month, etc.). The variables that do not participate in the proposed fuzzy clustering procedure are modeled with a membership function that takes the value of 1. The membership functions corresponding to all variables of the Fuzzy-RBF-CNN first layer are designed once and applied to all presented case studies without fine tuning.

Fuzzification of the input variables leads to the creation of fuzzy rules using the linguistic representations of the corresponding membership functions. Each fuzzy rule defines a data cluster of the Fuzzy-RBF-CNN first layer on which an RBF-CNN regressor will be connected to. The fuzzy rules are constructed using an iterative training procedure. In the first training cycle, an input vector x_i is randomly selected from the training set and used to create an initial fuzzy rule. For each variable *i*, its membership function outputs $g_m(x_i)$ are first calculated using its value from the selected input vector and the linguistic representation of the membership function with the highest output is used for the fuzzy rule creation. Then, the initial fuzzy rule activations

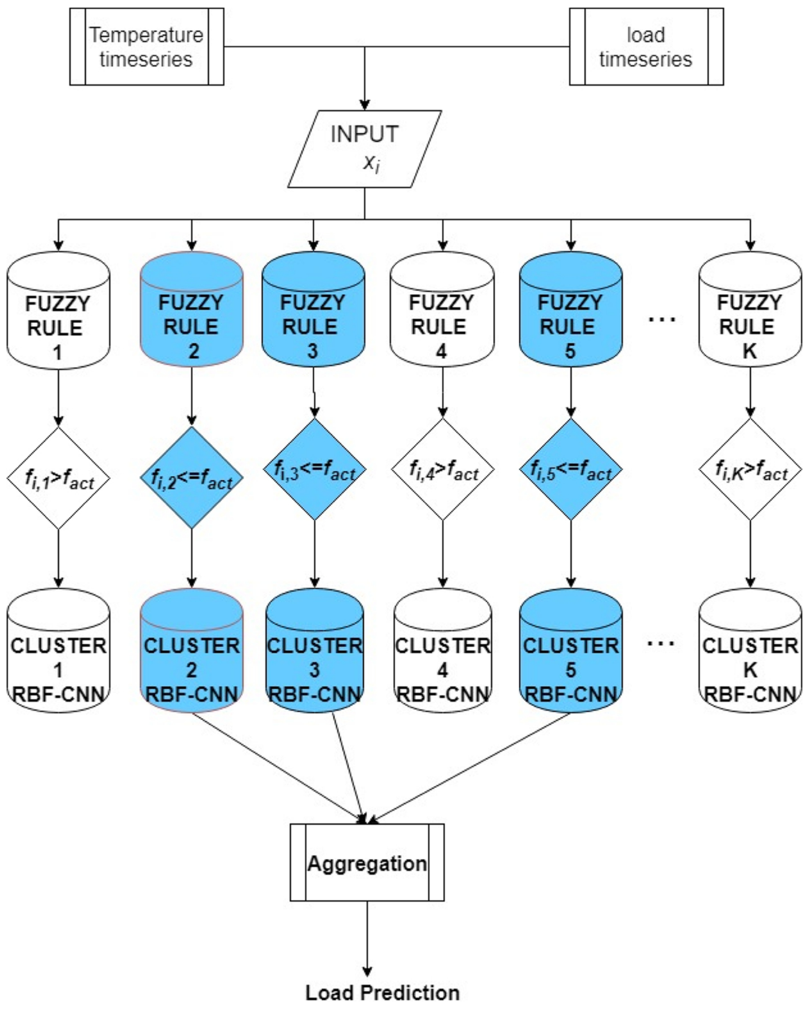

Fig. 1. The proposed model architecture (colored branches signify the activated clusters).

$f_{i,k}$ are computed applying the following equation to the entire training dataset:

$$f_{i,k} = \sqrt[p]{\prod_{\forall m} g_m(x_i)} \tag{1}$$

where $g_m(x_i)$ is the output of the variable m membership function contained in fuzzy rule k and p is the number of the most important input variables based on which the input data is clustered.

The next fuzzy rule is created using the input vector that activates less the existing rules. In specific, if there are k rules in the first layer, then the maximum rule activation is calculated for each input of the training set. The next rule is constructed using the input vector x_{i_s} that activates less both of the existing rules. Namely,

$$i_s = argmin_{\forall i}\{f_{i,argmax_{\forall k}\{f_{i,k}\}}\} \tag{2}$$

The training process of the Fuzzy-RBF-CNN first layer continues until the minimum activation $f_{i_s,argmax_{\forall k}\{f_{i,k}\}}$ stops increasing after a new fuzzy rule creation. The purpose of the f_{split} threshold is to limit the number of clusters at the first layer in order to minimize the overall model complexity. The value of 0.75 was concluded as being appropriate for the f_{split} threshold following an extensive trial-and-error procedure. Subsequently, the value of 0.75 was adopted in all forecasting case studies presented in this work.

Each fuzzy rule forms a data subset that belongs to the training set. The data subset of cluster k consists of the inputs that activate the fuzzy rule k more than a threshold $f_{act} = 0$. An algorithmic description of the fuzzy clustering approach is listed below:

```
Algorithm 1 Fuzzy clustering
1: Load the predefined membership functions $g_m(x_i)$
   for each variable $m$
2: Load the input dataset
3: Select randomly an input vector $x_i$
4: Compute the outputs $g_m(x_i)$
5: For each variable, select the membership function
   with the maximum output
6: Create the first rule
7: Compute the rule activation $f_{i,k}$ of each input
   using the eq. 1
8: do
9:     Compute the maximum rule activation $f_{max,i}$ of each input $x_i$
10:    Find the input vector $x_i^{'}$ with the minimum of $f_{max,i}$
11:    Compute the outputs $g_m(x_i)$
12:    For each variable, select the membership function
   with the maximum output
13:    Create the next rule
14:    Compute the rule activations $f_{i,k}$ of each input
   using the eq. 1
15: while the minimum of $f_{max,i}$ is less than the $f_{split}$ threshold
```

3.2. The RBFNN training procedure

For each data cluster formulated by the first layer fuzzy rules, an RBF-CNN training procedure is applied. In an RBF-CNN, the RBF kernels of an RBFNN are connected with a CNN and the RBF-CNN training procedure is performed in two steps. At the first step, a stand-alone RBFNN is trained using a cross-validation technique that provides three different sets of RBFs, while at the second step a CNN is trained using

Table 1
MAPE comparison between the proposed and the benchmark models.

Number of days ahead	Persistence	24h-MLR	24h-SVM	ML-SVM	Fuzzy-RBFNN	Fuzzy-RBF-CNN
1	4.92	2.46	2.03	1.98	1.86	**1.68**
2	6.74	2.80	2.27	2.21	1.97	**1.75**
3	7.35	2.84	2.33	2.31	1.97	**1.77**
4	7.73	2.98	2.52	2.46	1.99	**1.77**
5	7.73	3.10	2.63	2.54	2.03	**1.80**
6	7.06	3.15	2.71	2.59	2.05	**1.81**
7	5.88	3.16	2.72	2.64	2.08	**1.84**

Table 2
RMSE comparison between the proposed and the benchmark models in MW

Number of days ahead	Persistence	24h-MLR	24h-SVM	ML-SVM	Fuzzy-RBFNN	Fuzzy-RBF-CNN
1	1410	712	596	588	494	**463**
2	1876	784	664	628	514	**473**
3	2008	804	692	664	516	**475**
4	2082	854	755	733	518	**478**
5	2102	888	786	784	525	**481**
6	1943	906	800	804	529	**484**
7	1674	890	810	802	538	**492**

Table 3
Performance comparison between the proposed models and the ML-SVM During the normal and special days.

Number of days ahead	ML-SVM		Fuzzy-RBF-CNN	
	Normal days	Special days	Normal days	Special days
1	1.95	2.32	1.62	2.08
2	2.18	2.45	1.71	2.09
3	2.27	2.55	1.72	2.13
4	2.51	2.61	1.73	2.13
5	2.61	2.62	1.76	2.14
6	2.65	2.69	1.77	2.16
7	2.70	2.72	1.80	2.18

the input data transformed to three-dimensional arrays using the above three RBF sets.

Before the RBF-CNN training procedure begins, the input variables that do not contain useful information need to be removed from the data subset of cluster k by applying the permutation importance technique [44] using a random forest [8]. Initially, a random forest is trained and its performance defines the baseline. Then, a randomly chosen variable is permuted from the dataset and the random forest performance is calculated again. The variable importance is calculated as the difference between the baseline performance and the performance on the permuted dataset. The input variables with positive importance are kept for the RBF-CNN training.

Following the feature extraction on the cluster k dataset, the RBFNN training procedure is initiated. Generally, an RBFNN is a two-layer neural network with a hidden layer consisting of a set of non-linear RBFs and a linear output layer. The RBFNN hidden layer parameters that influence its performance are the number, centers and radii of RBFs which are correlated. The RBF centers map the problem features, the RBF radii determine the RBF activations and mine the information from the data, while the RBF number controls the model over- or under-fitting. The following equation describes the RBF type used for RBFNN training.

$$f_j^{2,k} = e^{-\sqrt{\sum\left(\frac{\|x_i - c_{i,j}^{2,k}\|}{(b_{i,j}^{2,k})^2}\right)}} \quad \forall\ j \in [1.\ J_m] \tag{3}$$

where k, i, j are the cluster, input variable and RBF indices, respectively. $c_{i,j}^{2,k}$, $b_{i,j}^{2,k}$ and $f_j^{2,k}$ are the centers, radii and activations of RBFs that belong to cluster k.

Using the dataset formed by the k cluster and the permutation importance technique, an RBFNN with a hidden layer output described in Eq. (3) is trained by a procedure that includes the K-means algorithm used for the RBF center allocation and the Adam optimization algorithm applied for the RBF radius estimation. The latter training procedure is repeated by a three-fold cross-validation technique for the optimal estimation of the RBF number.

In specific, the RBF centers are first located applying the K-means algorithm and then the Adam iterative stochastic optimization algorithm is employed to estimate the optimal RBF radii. The Adam algorithm implemented by the TensorFlow deep learning framework [42] is an effective, first-order derivative optimization algorithm which calculates the first and second order moments in each iteration by applying an adaptive learning rate and it has been selected due to its low computational and memory requirements. Additionally, it performs well with optimization problems having many parameters and very noisy gradients [33].

In the RBFNN training process, the Adam algorithm uses the sum squared error as an objective function calculated with a validation and testing sets. At each iteration, the RBF widths are updated and the RBFNN output layer weights are computed by the least square algorithm. Then, the sum squared errors on the validation and testing sets are calculated and when an error minimum is found, the radii are considered optimal and adopted.

The above procedure that consists of the RBF centers allocation and the RBF radius optimization is performed for a different number of RBFs using a coarse-to-fine approach [45]. In addition, for each candidate RBF number tested, a three-fold cross-validation is employed. Each fold of cross-validation consists of a validation and testing sets. The testing set corresponds to a continuous – most recent – portion of the training set, while the validation set is randomly chosen. The optimal number J_k of the hidden layer RBFs is selected by averaging the cross-validation results and the three corresponding RBFNNs are obtained. An algorithmic description of RBFNN training is shown below:

```
Algorithm 2 RBFNN training
1:  for Each cluster $k$ do
2:      Create the testing set, as the last part of the dataset
3:      Create the 3-fold datasets for the cross-validation
4:      for Each $j_k$ inside [12, 16, 20, 24, 28, 32, 36, 40, 44, 48] do
5:          Find $j_m$ RBF centers $c_{i,j}^{2,k}$ using Kmeans algorithm
6:          Initialize the RBF radius $b_{i,j}^{2,k}$
7:          for Each fold of cross-validation do
8:              for Each iteration of Adam algorithm do
9:                  Compute $f_{j,n}^{2,k}$ using eq.3
10:                 Estimate the RBFNN output layer weights
    using the least square method
11:                 Compute the gradients and update the radius $b_{i,j}^{2,k}$
12:                 Compute $SSE_m$ on the validation and testing set
13:             end for
14:         end for
15:     end for
16:     Return the RBF number $j_{best}$
    with the best cross-validation score
17:     for Each $j_k$ inside [$j_{best}$-2, $j_{best}$-1, $j_{best}$+1, $j_{best}$+2] do
18:         Perform the above RBFNN training procedure
19:     end for
20:     Save the RBFNNs with $j_{globalbest}$ RBFs
21: end for
22:
```

3.3. The CNN training procedure

Due to the 'curse of dimensionality' RBFNNs, similarly to all methods using kernel functions, have reduced performance. This is due to the aggregation of the individual activations of every element contained in an RBF that is a pair of ($c_{i,j}^{2,k}$, $b_{i,j}^{2,k}$), in order to extract a single

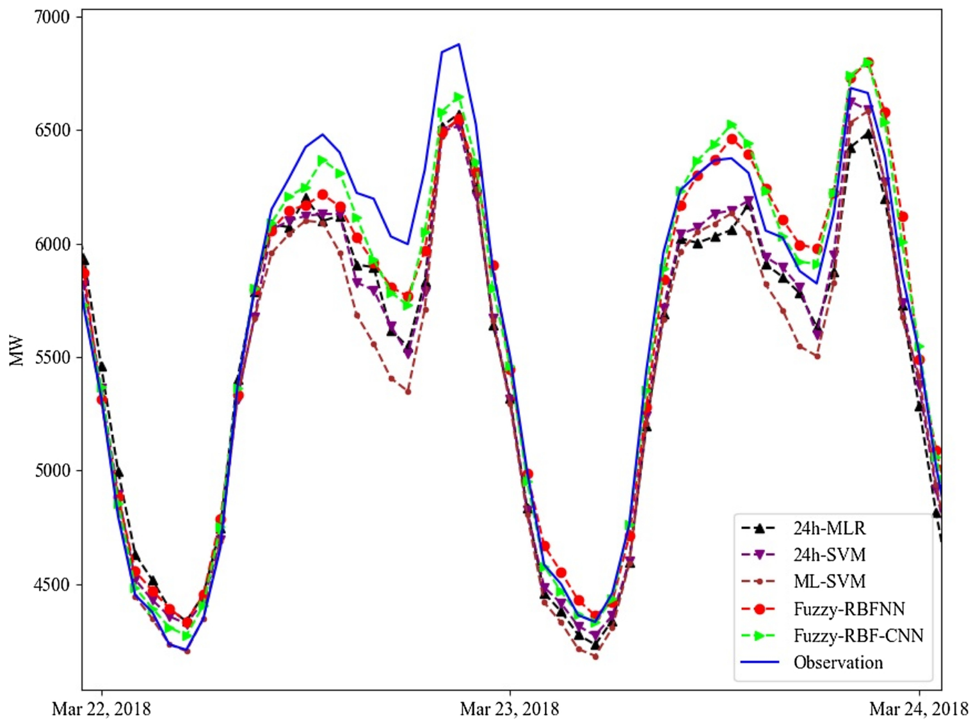

Fig. 2. Comparison of the proposed model against the benchmark models during a two day period in March 2018.

Table 4
MAPE comparison between the proposed and the benchmark models.

Number of days ahead	Persistence	24h-MLR	24h-SVM	ML-SVM	Fuzzy-RBFNN	Fuzzy-RBF-CNN
1	5.17	4.98	3.61	**2.87**	3.30	2.94
2	6.54	5.29	4,00	3.41	3.72	**3.39**
3	6.89	5.97	4.90	3.69	3.93	**3.59**
4	7.54	6,10	6.06	3.86	4.12	**3.77**
5	8.10	6.58	8.28	3.98	4.26	**3.91**
6	7.71	7.31	11.20	4.08	4.31	**3.95**
7	6.81	8.49	14.73	4.22	4.41	**4.07**

Table 5
RMSE comparison between the proposed and the benchmark models in MW.

Number of days ahead	Persistence	24h-MLR	24h-SVM	ML-SVM	Fuzzy-RBFNN	Fuzzy-RBF-CNN
1	103.71	94.57	70.39	**54.39**	64.16	56.41
2	127.15	101.52	75.53	60.51	67.98	**60.39**
3	129.33	108.92	88.79	64.15	70.96	**64.02**
4	137.31	111.77	108.59	66.87	73.08	**66.48**
5	148.60	118.55	142.52	68.43	74.39	**67.64**
6	141.21	130.21	187.49	69.16	75.04	**68.12**
7	121.06	147.24	243.60	70.36	75.99	**69.14**

Table 6
MAPE comparison between the proposed models and the ML-SVM during normal and special days.

Number of days ahead	ML-SVM		Fuzzy-RBF-CNN	
	Normal days	Special days	Normal days	Special days
1	**2.85**	3.19	2.98	**2.67**
2	**3.41**	3.77	3.42	**3.16**
3	3.67	3.96	**3.64**	**3.44**
4	3.82	3.97	**3.82**	**3.42**
5	3.95	4.25	**3.94**	**3.65**
6	4.03	4.40	**3.96**	**3.72**
7	4.17	4.53	**4.07**	**3.90**

matching value. To reduce the 'curse of dimensionality' effect from an RBFNN, the detailed activations of each RBF element computed by the following equation are considered as an image.

$$f_{i,j}^{2,k} = e^{-\sqrt{\frac{\|x_i - c_{i,j}^{2,k}\|}{(b_{i,j}^{2,k})^2}}} \quad \forall\ j \in [1.\ J_m] \tag{4}$$

Using the RBFs of the three RBFNNs obtained from the training procedure described in the previous subsection, each input vector is transformed to a three dimensional array. To that end, the training, validation and testing sets are applied to Eq. (4) using the RBFs extracted by the three RBFNNs and consequently three, four-dimensional arrays are formed with dimensions $3 \times D \times J_k \times N_{tr,k}$, $3 \times D \times J_k \times N_{val,k}$ and $3 \times D \times J_k \times N_{test,k}$, where D is the number of input variables, J_k is the optimal RBF number and $N_{tr,k}$, $N_{val,k}$ and $N_{test,k}$ are the sizes of the training, validation and testing sets, respectively. These datasets are used to deploy a CNN that represents a deep neural network commonly employed in image processing.

A CNN consists of convolutional layers followed by pooling layers, as well as fully-connected layers. A convolutional layer extracts different features from an image using a sliding frame (kernel) which has weights and strides along the image. Convolving the kernel outputs, multiple maps with the extracting features are created which are called filters [33]. In a convolutional layer, different weights are applied on each filter resulting in different representations of an image. A convolutional layer is always followed by a pooling layer that is used to reduce the filter spatial resolution and the pooling layer outputs are concluded by fully-connected layers. The Adam optimization algorithm is implemented for CNN training using the training, validation and testing sets transformed by the estimated RBFs during the previous training stage to four-dimensional arrays as described above. In each iteration of the Adam algorithm a small mini-batch from the training set is applied to update the CNN parameters, while the sum squared errors on the validation and testing sets are calculated to investigate the

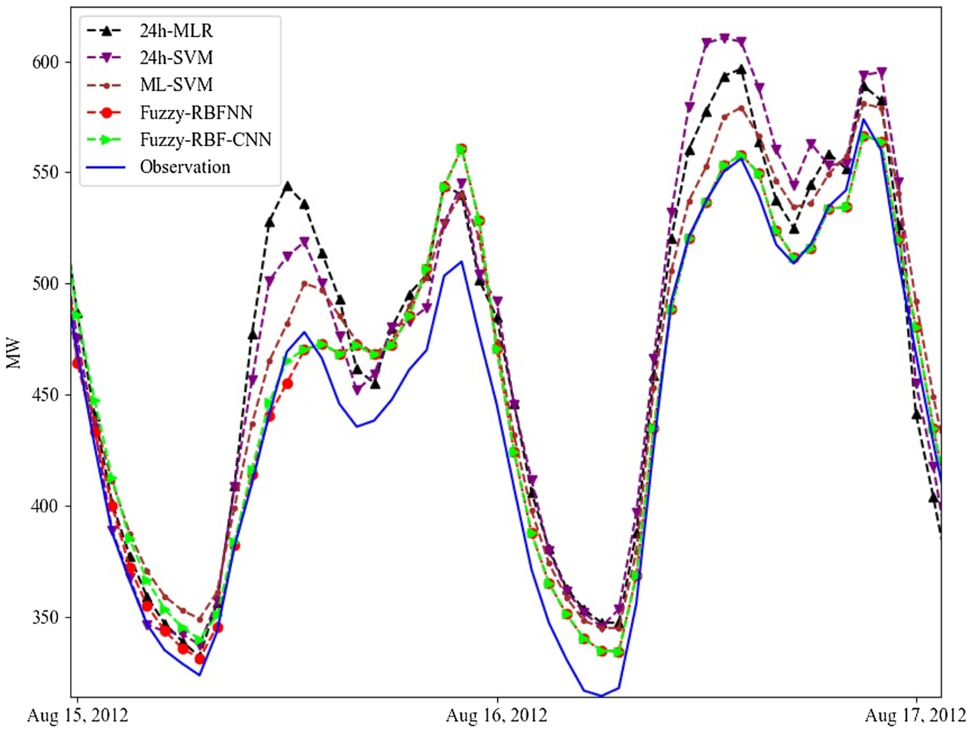

Fig. 3. Comparison of the proposed model against the benchmark models during a two day period in August 2012.

performance. Each CNN developed for each cluster of the proposed model consists of a convolutional, a pooling, and two fully-connected layers. In specific, the convolutional layer contains 32 filters analyzing the input with a kernel of size 2 × 4 pixels. The following pooling layer aggregates the filters with a stride of 2 pixels, while the fully-connected layers have 2048 and 512 neurons, respectively. An algorithmic description of RBFNN training is listed below:

```
Algorithm 3 CNN training
1:  for Each cluster $k$ do
2:      Create a testing set, as the last part of the dataset
3:      Create a validation set randomly
4:      Retrieve the RBFs of the three RBFNNs
    produced by the cross-validation of Algorithm 2
5:      Transform all inputs to 4-dimentional arrays $3 \times D \times j_{globalbest} \times N_k$
6:      Initialize the CNN
7:      Create mini-batches for each iteration of Adam algorithm
8:      for Each iteration of Adam algorithm do
9:          Retrieve a mini-batch
10:         Compute the CNN outputs
11:         Compute the gradients and update the CNN parameters
12:         Retrieve the validation and testing set and compute $SSE_k$
13:         Save the network if $SSE_k$ is the minimum
14:     end for
15: end for
16:
```

4. Model evaluation

4.1. Case studies

In order to evaluate the proposed model performance, two case studies with different characteristics have been examined. The first case study corresponds to the Hellenic interconnected power system, where the load timeseries used (available at Ref. [46]) covers the period from 01/01/2014 to 30/06/2018, while timeseries of temperature forecasts have been obtained from the SKIRON meteorological model (available at Ref. [47]. The temperature forecasts correspond to five Greek cities, namely Athens, Thessaloniki, Patra, Ioannina and Alexandroupoli. The proposed model was trained with the data recorded during the first four years and it was tested with the data recorded during 2018. This case study is distinct since the annual demand is not increasing as it would be expected but on the contrary, the annual demand for 2014 and 2015 was 50,04 TWh and 51,22 TWh, respectively, while for 2016 and 2017 it was approximately 50,09 TWh.

The proposed model was also applied to the power system of the Greek island of Crete. The Cretan power system is an isolated system that supplies high loads during the summer since it is a popular tourist destination. The available load data provided by the Hellenic distribution system operator, covers a three year period from 01/01/2010 to 31/12/2012. During that period, the peak load demand in the summer of 2012 was 660 MW while at the same year the minimum load was 135 MW. Furthermore, the annual peak load in 2010 and 2011 was 655 MW and 608 MW, respectively.

The proposed model has been evaluated in accordance to customary requirements posed by a typical energy market participant (supplier). It runs once per day using the data recorded until the previous day and provides hourly load predictions for the next seven days, iteratively. Initially, the hourly loads of the current day are estimated and used as input to forecast the day-ahead loads necessary for energy market participation. Furthermore, the proposed model hyper-parameters and fuzzy membership functions of the first layer are applied to both case studies without any modifications. To illustrate the model performance, the mean absolute percentage error (MAPE), the mean absolute error (MAE) and the root mean squared error (RMSE) are used.

4.2. Input data and fuzzy modeling

Following the data selection strategy discussed in Ref. [48], the input variables applied to the proposed model consist of historical load measurements [49], temperature forecasts [50] and calendar data [49] that includes numerical categories indicating the hour, month and type of the day at the time the load prediction is made. In addition, an index that indicates the special days is used. The past load values used as model inputs correspond to several lags, mainly during last week, but they can also reach until the respective day of the previous year. Due to limited data availability for the Cretan power system, the lags used are from the last three week period only.

The most important variables in the fuzzy clustering method in the first layer are the mean value of the previous day loads, the maximum daily temperature of the prediction day and the 'hour', 'month' and special day index of the prediction time. For the variables 'hour' and 'month' three trapezoidal membership functions have been employed, while two trapezoidal membership functions have been used for the special day index. The mean value of previous day loads has been modeled using three Gaussian membership functions and the maximum daily forecasted temperature has been split into five fuzzy sets represented by Gaussian membership functions. The number of clusters produced at the first layer is 76 in case of the Hellenic power system and 68 in case of the Cretan power system.

Thus, the variables 'hour', 'month', ‘special day index’ and ‘previous day load’ are modelled by fuzzy membership functions, while the remaining input variables described above are modeled with a constant membership function that takes the value of 1. The fuzzy rules create data clusters forming subsets of the initial dataset. A subset consists of the input samples that activated the corresponding fuzzy rule. The two-stage training procedure of the RBF-CNNs is repeated for each fuzzy rule. When the Fuzzy-RBF-CNN is used for prediction making, the activations of the fuzzy rules are computed using the input vector and the RBF-CNNs that correspond to the fuzzy rules with activations greater than the threshold f_{act} (in this cases $f_{act} = 0$) are engaged to produce the ensemble predictions.

4.3. Benchmarks

Two state-of-the-art and a traditional forecasting models have been developed in order to validate the performance of the proposed model. The first two models have similar structure, but they use a different regression method. Their structures consist of 24 regressors, each one trained to provide forecasts for one hour during a day. The regressors applied to the first benchmark model are developed using the multiple linear regression method [51], while the second model has been built using support vector machines [10]. These benchmarks have been identified as 24h-MLR and 24-SVM, respectively.

In order to validate the proposed RBF-CNN performance, SVMs have been used in the proposed model structure replacing the RBF-CNN regressors. This benchmark model is identified as ML-SVM and has also three layers. The input data is clustered in the first layer using the same method proposed in this work. For each cluster, a different SVM has been trained for producing a different prediction and the final prediction is obtained as the average of the produced ensemble predictions.

The forecasts obtained from the RBFNNs applied in the proposed model have been evaluated. These forecasts are produced averaging first the three RBFNN outputs on each activated cluster and second all the activated cluster predictions. This model is identified as Fuzzy-RBFNN. In addition, the performance of the naïve ‘persistence’ method has been computed and compared with that of the proposed model and the two aforementioned models [52].

4.4. Results from the Hellenic interconnected power system

The MAPE and RMSE of the proposed model are shown in Tables 1 and 2. Initially, the non-linearity and complexity characterizing the load timeseries are shown comparing the accuracy achieved by the linear 24h-MLR and the non-linear 24h-SVM models. The MAPEs of the Fuzzy-RBF-CNN and Fuzzy-RBFNN have similar values during the forecasting horizon, in contrary to other benchmarks that seem less robust. Regarding the 24h-SVM and ML-SVM performances, the proposed architecture does not seem to provide a remarkable improvement in this case study. However, the proposed RBFNNs outperform the robust and popular SVMs achieving improvements in the range of 6% to 21% [25]. Furthermore, significant performance improvement presents the applied CNNs w.r.t. the stand-alone RBFNNs. The finer analysis on RBF activations performed by a CNN provides an approximate 10% improvement in comparison with the least square method that is commonly used in an RBFNN second layer.

Comparing the ML-SVM performance with that of the 24h-SVM in this case-study, the model structure obtained by the fuzzy clustering method provides significant improvement in short horizons, while in horizons longer than 4 days ahead the ML-SVM and 24h-SVM perform similarly. In this case study, the high prediction accuracy is achieved by the RBF-CNN regressors.

Table 3 illustrates the MAPE of the Fuzzy-RBF-CNN model and the ML-SVM model calculated using their forecasts obtained during the normal and special days (Sundays and holidays), separately. The proposed model outperforms the benchmark model during the entire forecasting horizon for both normal and special days. In case of special days, the Fuzzy-RBF-CNN's MAPE is 2.13% for three and four day-ahead horizons. It is important that a number of utilities uses these forecasts for following non-working days in order to participate in the day-ahead energy market.

The high accuracy of the proposed model is shown in Fig. 2, where the forecasts of both discussed models during two consecutive days in March 2018, are presented. It appears that the best results are obtained from the Fuzzy-RBFNN and Fuzzy-RBF-CNN models, while the benchmark model provides similar performance.

4.5. Results from the Cretan power system

The proposed model is also evaluated in the Cretan power system. In Table 4 and 5 the MAPE and RMSE of the presented models (Fuzzy-RBF-CNN and benchmarks) are listed for every day of a week-ahead horizon. The proposed model performance in that case is comparable to that of ML-SVM. In specific, the MAPE as well as the RMSE of the ML-SVM is slightly lower than that of the proposed model for one day-ahead forecasting which corresponds to the current prediction day, while for horizons longer than two day-ahead, the Fuzzy-RBF-CNN outperforms all benchmark models. Comparing the 24h-SVM MAPE with the ML-SVM and the Fuzzy-RBFNN MAPEs, the proposed fuzzy clustering method appears effective, especially for longer horizons, since the ML-SVM and the Fuzzy-RBFNN achieve 40% and 35% on the average improvement w.r.t. the 24h-SVM, respectively. The CNN application to a RBFNN hidden layer output provides significant robustness to the RBFNNs, which outperform the SVM. As shown, the poor performance of the 24h-MLR proves the high complexity of that case study. Furthermore, the naïve ‘persistence’ method outperforms the 24h-MLR and the 24h-SVM in long-term horizons.

In a similar evaluation approach to the case study of the Hellenic power system, the MAPE values of the proposed model and the ML-SVM are listed in Table 6. The superior performance of the Fuzzy-RBF-CNN is shown during special days where the proposed model outperforms the ML-SVM, providing a 15% improvement on the average. However, during normal days, the ML-SVM outperforms the Fuzzy-RBF-CNN in current day predictions, while they have similar performances beyond the day-ahead forecasting.

Fig. 3 shows the predictions of both presented models during a special day (15/08/2012) and a normal day (16/12/2012). The improved performance of the proposed model with respect to the benchmark models appears in most hours during both days. Especially in the morning hours, the benchmark model errors outreach 10%, while at the same hours, the proposed model performs with errors lower than 1%.

5. Conclusions

A novel, fuzzy-based ensemble model that uses hybrid deep learning neural networks for load forecasting has been presented. The proposed model initially clusters the input data to multiple groups in a manner that an input vector belongs to more than one, but not all, groups. Using this fuzzy clustering method, the inputs are distributed to different clusters shaping multiple data subsets. Each of these subsets is used to train a hybrid deep neural network that consists of an RBF, a convolutional and two fully-connected layers. The RBF layer parameters are estimated separately by initially training a stand-alone RBFNN using a three-fold cross-validation technique. Following the RBFNN training, the RBF hidden layers from the three networks obtained by the cross-validation are used to transform the one dimensional input data to a three dimensional space. The resulting four dimensional training set is used to train a CNN that provides a load prediction. The RBFNN and the CNN are designed using the Tensorflow deep learning framework and trained by the Adam optimization algorithm. The above training procedure is performed for each cluster created by the fuzzy clustering method. Future improvements of the proposed model include the training procedure to be performed in an one-step approach instead of the two-stage method presented here. The proposed model has been evaluated in two different case studies that correspond to the Hellenic interconnected power system and to the isolated Cretan power system. The results show the superior performance of the proposed model as compared to other state-of-the-art and traditional models and exhibit the effectiveness of both the novel fuzzy clustering method and the replacement of the RBFNN output layer with a CNN.

Conflict of Interest

None.

Acknowledgements

We acknowledge support of this work by the project "NCSRD-INRASTES research activities in the framework of the national RIS3" (MIS 5002559) which is implemented under the "Action for the Strategic Development on the Research and Technological Sector", funded by the Operational Programme "Competiveness, Entrepreneurship and Innovation" (NSRF 2014-2020) and co-financed by Greece and the European Union (European Regional Development Fund).

References

[1] H. Hahn, S. Meyer-Nieberg, S. Pickl, Electric load forecasting methods: tools for decision making, E. J. Oper. Res. 199 (3) (2009) 902–907.
[2] E. Delarue, W. D'haeseleer, Adaptive mixed-integer programming unit commitment strategy for determining the value of forecasting, Appl. Energy 85 (4) (2008) 171–181.
[3] S.N. Fallah, R.C. Deo, M. Shojafar, M. Conti, S. Shamshirband, Computational intelligence approaches for energy load forecasting in smart energy management grids: state of the art, future challenges, and research directions, Energies 11 (3) (2018).
[4] L. Hernandez, et al., A survey on electric power demand forecasting: future trends in smart grids, microgrids and smart buildings, IEEE Commun. Surv. Tutorials 16 (3) (2014) 1460–1495.
[5] J. Fiot, F. Dinuzzo, Electricity demand forecasting by multi-task learning, IEEE Trans. Smart Grid 9 (March (2)) (2018) 544–551.
[6] V. Thouvenot, A. Pichavant, Y. Goude, A. Antoniadis, J. Poggi, Electricity forecasting using multi-stage estimators of nonlinear additive models, IEEE Trans. Power Syst. 31 (September (5)) (2016) 3665–3673.
[7] G.-F. Fan, L.-L. Peng, W.-C. Hong, Short term load forecasting based on phase space reconstruction algorithm and bi-square kernel regression model, Appl. Energy 224 (2018) 13–33.
[8] G. Sideratos, A. Ikonomopoulos, N. Hatziargyriou, A committee of machine learning techniques for load forecasting in a smart grid environment, Int. J. Energy Power 4 (2015) 98–105.
[9] T. Boutsika, G. Sideratos, A. Ikonomopoulos, An expert committee evaluation for load forecasting in a smart grid environment, ENEFM2015 (2015) 135–141.
[10] E. Ceperic, V. Ceperic, A. Baric, A strategy for short-term load forecasting by support vector regression machines, IEEE Trans. Power Syst. 28 (November (4)) (2013) 4356–4364.
[11] L. Ghelardoni, A. Ghio, D. Anguita, Energy load forecasting using empirical mode decomposition and support vector regression, IEEE Trans. Smart Grid 4 (March (1)) (2013) 549–556.
[12] X. Zhang, J. Wang, K. Zhang, Short-term electric load forecasting based on singular spectrum analysis and support vector machine optimized by Cuckoo search algorithm, Electr. Power Syst. Res. 146 (2017) 270–285.
[13] Y. Yang, J. Che, C. Deng, L. Li, Sequential grid approach based support vector regression for short-term electric load forecasting, Appl. Energy (2019) 1010–1021.
[14] S. Fan, L. Chen, Short-term load forecasting based on an adaptive hybrid method, IEEE Trans. Power Syst. 21 (February (1)) (2006) 392–401.
[15] A. Khosravi, S. Nahavandi, Load forecasting using interval type-2 fuzzy logic systems: optimal type reduction, IEEE Trans. Ind. Inf. 10 (May (2)) (2014) 1055–1063.
[16] A. Laouafi, M. Mordjaoui, S. Haddad, T.E. Boukelia, A. Ganouche, Online electricity demand forecasting based on an effective forecast combination methodology, Electr. Power Syst. Res. 148 (2017) 35–47.
[17] M. Hanmandlu, B.K. Chauhan, Load Forecasting Using Hybrid Models, IEEE Trans. Power Syst. 26 (February (1)) (2011) 20–29.
[18] M.S. Nazar, A.E. Fard, A. Heidari, M. Shafie-khah, J.P.S. Catalão, Hybrid model using three-stage algorithm for simultaneous load and price forecasting, Electr. Power Syst. Res. 165 (2018) 214–228.
[19] S. Li, P. Wang, L. Goel, Short-term load forecasting by wavelet transform and evolutionary extreme learning machine, Electr. Power Syst. Res. 122 (2015) 96–103.
[20] H. Kebriaei, B.N. Araabi, A. Rahimi-Kian, Short-term load forecasting with a new nonsymmetric penalty function, IEEE Trans. Power Syst. 26 (Novemebr (4)) (2011) 1817–1825.
[21] H. Mao, X. Zeng, G. Leng, Y. Zhai, J.A. Keane, Short-term and midterm load forecasting using a bilevel optimization model, IEEE Trans. Power Syst. 24 (May (2)) (2009) 1080–1090.
[22] C. Cecati, J. Kolbusz, P. Różycki, P. Siano, B.M. Wilamowski, A novel rbf training algorithm for short-term electric load forecasting and comparative studies, IEEE Trans. Ind. Electron. 62 (October (10)) (2015) 6519–6529.
[23] Z. Yun, Z. Quan, S. Caixin, L. Shaolan, L. Yuming, S. Yang, RBF neural network and ANFIS-based short-term load forecasting approach in real-time price environment, IEEE Trans. Power Syst. 23 (August (3)) (2008) 853–858.
[24] G. Sideratos, I. Vitellas, N. Hatziargyriou, A load forecasting hybrid method for an isolated power system, ISAP'11 conference, Crete, 2011, pp. 1–5.
[25] A.S. Khwaja, X. Zhang, A. Anpalagan, B. Venkatesh, Boosted neural networks for improved short-term electric load forecasting, Electr. Power Syst. Res. 143 (2017) 431–437.
[26] M. Ghayekhloo, M.B. Menhaj, M. Ghofrani, A hybrid short-term load forecasting with a new data preprocessing framework, Electr. Power Syst. Res. 119 (2015) 138–148.
[27] A.S. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, B. Venkatesh, Improved short-term load forecasting using bagged neural networks, Electr. Power Syst. Res. 125 (2015) 109–115.
[28] S. Li, P. Wang, L. Goel, A Novel wavelet-based ensemble method for short-term load forecasting with hybrid neural networks and feature selection, IEEE Trans. Power Syst. 31 (May (3)) (2016) 1788–1798.
[29] P. Singh, P. Dwivedi, Integration of new evolutionary approach with artificial neural network for solving short term load forecast problem, Appl. Energy 217 (2018) 537–549.
[30] S.A. Billings, H.L. Wei, M.A. Balikhin, Generalized multiscale radial basis function networks, Neural Networks 20 (10) (2007) 1081–1094.
[31] W. Yao, X. Chen, Y. Zhao, M. van Tooren, Concurrent subspace width optimization method for rbf neural network modeling, IEEE Trans. Neural Networks Learn. Syst. 23 (2) (2012) 247–259.
[32] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Beijing, 1995.
[33] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[34] K. Amarasinghe, D.L. Marino, M. Manic, Deep neural networks for energy load forecasting, IEEE 26th International Symposium on Industrial Electronics (ISIE), Edinburgh, 2017, pp. 1483–1488.
[35] J. Bedi, D. Toshniwal, Deep learning framework to forecast electricity demand, Appl. Energy (2019) 1312–1326.
[36] F. He, J. Zhou, Z.-K. Feng, G. Liu, Y. Yang, A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with Bayesian optimization algorithm, Appl. Energy (2019) 103–116.
[37] S. Wang, X. Wang, S. Wang, D. Wang, Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting, Int. J. Electr. Power Energy Syst. 109 (2019) 470–479.
[38] X. Dong, L. Qian, L. Huang, A CNN based bagging learning approach to short-term load forecasting in smart grid, 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, San Francisco, CA, 2017, pp. 1–6.
[39] L. Li, K. Ota, M. Dong, Everything is Image: CNN-based Short-Term Electrical Load Forecasting for Smart Grid, 2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC), Exeter, 2017, pp. 344–351.
[40] Xishuang Dong, Lijun Qian, Lei Huang, Short-term load forecasting in smart grid: a combined CNN and K-means clustering approach, 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju, 2017, pp. 119–125.
[41] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, International Conference on Learning Representations, San Diego, 2015.
[42] M. Abadi, et al., TensorFlow: a system for large-scale machine learning, Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, Savannah, USA, 2016, pp. 265–283 Software available from www.tensorflow.org.
[43] G. Sideratos, N. Hatziargyriou, An advanced statistical method for wind power forecasting, IEEE Trans. Power Syst. 22 (February (1)) (2007) 258–265.
[44] C. Strobl, A.L. Boulesteix, T. Kneib, T. Augustin, A. Zeileis, Conditional variable

importance for random forests, BMC Bioinf. 9 (1) (2008) 307.
[45] H.X. Tang, H. Wei, A coarse-to-fine method for shape recognition, J. Comput. Sci. Technol. 22 (2) (2007) 330–334.
[46] www.admie.gr.
[47] www.openskiron.org.
[48] P. Jiang, F. Liu, Y. Song, A hybrid forecasting model based on date-framework strategy and improved feature selection technology for short-term load forecasting, Energy 119 (2017) 694–709.
[49] L. Xiao, W. Shao, T. Liang, C. Wang, A combined model based on multiple seasonal patterns and modified firefly algorithm for electrical load forecasting, Appl. Energy 167 (2016) 135–153.
[50] Z. Hu, Y. Bao, T. Xiong, R. Chiong, Hybrid filter–wrapper feature selection for short-term load forecasting, Eng. Appl. Artif. Intell. 40 (2015) 17–27.
[51] T. Hong, P. Wang, H. Willis, A naïve multiple linear regression benchmark for short term load forecasting, Proceedings of the IEEE Power Energy Society General Meeting, San Diego, CA, USA, 2011, pp. 1–6.
[52] S. Dutta, et al., Load and renewable energy forecasting for a microgrid using persistence technique, Energy Procedia 143 (2017) 617–622.