Energy 175 (2019) 365–377

Contents lists available at ScienceDirect

Energy

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

# Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series

Hossein Javedani Sadaei [a, b], Petrônio Cândido de Lima e Silva [a, c], Frederico Gadelha Guimarães [a, d, *], Muhammad Hisyam Lee [e]

[a] Machine Intelligence and Data Science (MINDS) Laboratory, Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, Brazil
[b] Machine Learning Practice Lead in Avenue Code, Brazil
[c] Graduate Program at Department of Electrical Engineering, Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, Brazil
[d] Department of Electrical Engineering, Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, Brazil
[e] Department of Mathematical Sciences, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia

ARTICLE INFO

*Article history:*
Received 26 July 2018
Received in revised form
12 March 2019
Accepted 13 March 2019
Available online 19 March 2019

*Keywords:*
Multivariate time series
Convolutional neural networks
Short term load forecasting
Time series forecasting
Deep learning

ABSTRACT

We propose a combined method that is based on the fuzzy time series (FTS) and convolutional neural networks (CNN) for short-term load forecasting (STLF). Accordingly, in the proposed method, multivariate time series data which include hourly load data, hourly temperature time series and fuzzified version of load time series, was converted into multi-channel images to be fed to a proposed deep learning CNN model with proper architecture. By using images which have been created from the sequenced values of multivariate time series, the proposed CNN model could determine and extract related important parameters, in an implicit and automatic way, without any need for human interaction and expert knowledge, and all by itself. By following this strategy, it was shown how employing the proposed method is easier than some traditional STLF models. Therefore it could be seen as one of the big difference between the proposed method and some state-of-the-art methodologies of STLF. Moreover, using fuzzy logic had great contribution to control over-fitting by expressing one dimension of time series by a fuzzy space, in a spectrum, and a shadow instead of presenting it with exact numbers. Various experiments on test data-sets support the efficiency of the proposed method.



## 1. Introduction

### 1.1. Motivation

Load forecasting is essential to recent smart energy management systems. Every day, there are more and more uses for load forecasting. Usually, Short-Term Load Forecasting (STLF) with lead times in the scope of half an hour to one day is necessary to control programming and energy transfer scheduling and load dispatching. Thus, any improvement in the accuracy of STLF can lead to an improvement in the performance of power management and a decrease in the expenses of the power system. In a very important work, Gross et al. [1] provided a discussion about the importance of STLF and its role in the formulation of economic, reliable, and secure operating strategies for the power system. In their paper they counted the following impacts of STLF on power supply organization:

- STLF drives the scheduling functions that determine the most economic commitment of generation sources.
- STLF is required for the hydro scheduling function to determine the optimal releases from the reservoirs and the generation levels in the power plants.
- STLF is needed by the unit commitment function to determine the minimal cost hourly strategies for the start-up and shut-down of units to supply the forecast load, for purely thermal systems.
- For mixed hydro and thermal systems, STLF are required by the hydro-thermal coordination function to schedule the hourly operation of the various resources so as to minimize production costs.

* Corresponding author. Machine Intelligence and Data Science (MINDS) Laboratory, Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, Brazil.
*E-mail addresses:* hsadaei@avenuecode.com (H.J. Sadaei), petronio.candido@ifnmg.edu.br (P.C. de Lima e Silva), fredericoguimaraes@ufmg.br (F.G. Guimarães).

https://doi.org/10.1016/j.energy.2019.03.081
0360-5442/

- STLF is helpful for predictive assessment of the power system security.
- STLF is an essential data requirement of the off-line network analysis function for the detection of future conditions under which the power system may be vulnerable.
- STLF provides system dispatchers with timely information, i.e., the most recent load forecast, with the latest weather prediction and random behavior taken into account.

As it can be seen an accurate STLF is a key to success for electricity supply companies and it motivated the researchers of our study to develop accurate and practical methods for this end. To explain better the contribution of our study, the next section provides the literature review of related works and also some problems and gaps are discussed.

### 1.2. Literature review

STLF has become more interesting in recent years among researchers because of its increasing importance in smart grids and micro-grids, the integration of renewable energy sources closer to the consumers, and the degree of difficulty that is involved in this line of research of forecasting. Different classical and novel methods have been suggested for STLF, such as Kalman filtering, Regression, Auto-regressive Integrated Moving Average models (ARIMA), Seasonal Auto-regressive Integrated Moving Average models (SARIMA), Seasonal Auto-regressive Fractionally Integrated Moving Average (SARFIMA), etc. In 2004, Al-Hamadi et al. [2] used time-varying state space model to model the load demand on hourly basis. Kalman filter was used recursively to estimate the optimal load forecast parameters for each hour of the day [2]. Later, Song et al. in Ref. [3] proposed a fuzzy linear regression model that was made from the load data. In that method, by using previous three years, the coefficients of the model were found, by solving the mixed linear programming problem. In a study conducted by Vähäkylaand et al. [4], a Box-Jenkins time series analysis using ARIMA model for STLF was employed, and a forecasting system developed at the Imatra Power Company was described. The forecasting algorithm was simple, fast and accurate and transfer function model was used to introduce temperature effects, thus further improving accuracy.

In the recent years, other new methods have been applied for load forecasting, such as Artificial Intelligence (AI) and fuzzy logic. Intelligent solutions, based on AI technologies, to solve STLF are becoming more and more widespread nowadays. AI-based systems are being developed and deployed worldwide in many applications, mainly because of their symbolic reasoning, flexibility and explanation capabilities. As an example, Ranaweera et al. proposed a methodology that used fuzzy rules to incorporate historical weather and load data. These fuzzy rules were obtained from the historical data using a learning-type algorithm [5]. Yaoyao et al. [6] presented a model to quantify the uncertainty connected with power load and getting more information of future load, and then a neural network was employed to reconstruct the quantile regression model for constructing probabilistic forecasting method. In another work, a hybrid forecast method of the wavelet transform, neural network and the evolutionary algorithm was proposed in Ref. [7] and employed for STLF. In a similar approach, Chofrani et al. in Ref. [8] introduced a combination of Bayesian neural network (BNN) and a wavelet transform to produce the detailed load characteristics for BNN training. In this model, a weighted sum of the BNN outputs was applied to predict the load for a specific day. Yet in another hybrid model, an STLF model proposed by Fan et al. [9] combines Phase Space Reconstruction (PSR) algorithm with BSK regression model. In that model load data could be reproduced by PSR algorithm to derive the evolutionary trends of the historical load data and the embedded important features information to increase the reliability of the prediction. The BSK model, on the other hand, represented the spatial structures between regression points and their neighbor points to get the rules of rotation rules and disturbance in each dimension. Finally, the proposed model including multi-dimensional regression was successfully established and used for STLF [9]. Mamlook et al. in Ref. [10] employed fuzzy logic controller on an hourly base to forecast the impact of various conditional parameters, e.g., climate, time, load historical data, and random disturbances, on load forecasting in terms of fuzzy sets through the generation process. In this study, wavelet transform decomposed the time series into its components and then each component was forecast by a combination of neural network and an evolutionary algorithm.

In an important study, a list of main researches about using AI for STLF was provided by Metaxiotis et al. [11]. Compared against those models of AI, CNNs recently show very good performance in forecasting. CNNs were introduced by Fukushima in its very simple form in Ref. [12]. Later, LeCun et al. in Ref. [13] presented the current form of CNNs with more advanced concepts. Since CNNs proposed by Fukushima, there have been many improvements and extensions, such as max pooling layers & batch normalization by LeCun [14]. In Section 2, the architecture and components of CNNs is explained with more details. Although CNNs are very successful for the prediction [15,16], one of the main problems of using them which is common in many other deep neural networks methods is over-fitting [17]. As Burnham defined in Ref. [18] over-fitting is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably. There are some techniques to avoid or reduce over-fitting in deep learning models, such as data augmentation techniques, batch normalization, using regularization terms in the objective function, see Ref. [19] for more details. Some techniques are specifically designed for CNN models, such as including a pooling layer in the architecture [20]. In summary, CNN is a good candidate for STLF applications, as long as over-fitting can be controlled. There will be discussion about this issue in next section.

On the other hand, Fuzzy Time Series (FTS) has been used as a pattern learning based method in many time series applications including STLF. FTS method has been introduced in 1993 by Song [21] and since then they have been adopted in many forecasting applications. For example Yu in 2005, see Ref. [22], developed a new version of weighted FTS for stock market forecasting. In 2009, in another study, conducted by Wang et al., a new model of FTS for temperature and stock index forecasting has been introduced [23]. Related to the focus of this paper, there are also some studies which used FTS model by combination with another logic. For instance, Sadaei et al. combined an evolutionary algorithm, i.e., Improved Harmony Search (IHS) with Weighed FTS and used load data from France and United Kingdom to validate the model [24]. That model showed better performance in comparison with other FTS methods and some other state-of-the-art methods. Yet in another study, Sadaei et al. combined Seasonal Auto-regressive Fractional Moving Average model with FTS for STLF. The proposed hybrid model suppressed all its counterparts in terms of accuracy and performance [25]. In addition to these studies, FTS has been adopted for STLF in many other researches [26–28].

### 1.3. The challenges while building STLF model

For providing a high performance STLF model, some considerations need to be taken into account. It is usually believed that a short-term load is a variable that is influenced by many factors, e.g.,

historical load data, weather data, such as wind speed, precipitation, atmospheric pressure, temperature, and humidity. It is almost impossible to make an accurate prediction using a single model. Another problem related to building robust STLF model is the complexity that exists inside its data. It is shown in many researches that confirm various non-linear features and complicated intra-seasonality features in the data [29,30] (see for example Fig. 1).

This complexity is an obstacle first to find the right model for forecasting and second for setting up required parameters of that model in an optimal way. As an example, using classical linear models for STLF, e.g., SARIMA, ARIMA, and SARFIMA involve massive load data pre-processing and Auto Correlation and Partial Auto Correlation Functions analyzing to figure out the proper sets of hyper-parameters of the model. Even by having information from ACF and PACF, it is hard to select the best combinations of parameters [31,32] (see for instance Fig. 2).

Table 1 gathers the literature review discussed so far, summarizing advantages and disadvantages of each model.

As it can be seen, hybrid FTS models have shown good performance for STLF, in addition, AI based models, especially CNNs, produced good results, if and only if over-fitting can be avoided. Another good aspect of deep learning is that feature selection and extraction can be automated, embedded in the network. This will resolve the complexity of model building. From this discussion, it can be concluded that the combination of FTS with CNNs by controlling over-fitting can be a good alternative for STLF. This is the main motivation of this paper. In the next section the authors discuss about contributions of this work.

### 1.4. Contribution

As explained earlier, the main concern of this study is to propose a combined model with high accuracy and automated feature extraction. As it will be shown later in Section 2, the proposed method, in addition to its effectiveness in performance, is easy to be implemented with minimum effort for all required setting. However, as we discussed, one of the problems while using deep learning model, including CNNs, is over-fitting. Therefore, one of the main contributions in our proposed method is to deal with this problem. There are some proposed approaches in some other works to resolve over-fitting in CNNs, e.g., using dropout techniques [33]. Besides using dropout technique, we will use fuzzy logic which is the core of FTS to apply regularization in the input layer. The idea is to use FTS together with original time series and convert them to the format of images in the input layer. This approach will be discussed in Section 4. Another motivation to use FTS in our proposed method is to harness the power of fuzzy logic to successfully deal with uncertainties that reside on many non-linear time series applications, including STLF. There was a discussion about usefulness of using FTS as a hybrid method for STLF in the previous section. Moreover, as it is shown in the next Sections, the combination of FTS with CNNs improves the performance of the proposed method.

### 1.5. Paper organization

This paper is organized as follows: Next Section reviews the related works and preliminary concepts of FTS, CNN, and fuzzy logic; Section 3 describes the data that has been used in this study and later in Section 4 the methodology of the research is presented. Section 5 discusses the results, and finally, Section 6 concludes the paper.

## 2. Background of study

In the first part of this Section the fundamentals and definitions of FTS are reviewed and then CNN concept is explained. Finally, a brief introduction to fuzzy logic is discussed.

### 2.1. Fuzzy time series

Song and Chissom first presented the concepts of FTS [21,34], where the values in a time series data are represented by fuzzy sets [35]. Hereafter, we base our presentation on the conventional literature on FTS, with few changes.

Let $U$ be the universe of discourse, usually a continuous domain. A fuzzy set of $U$ is defined as follows:

$$A_i = \{\mu_{A_i}(x)/x \mid x \in U\} = \int_U \mu_{A_i}(x)/x \tag{1}$$

where $\mu_{A_i} : U \rightarrow [0, 1]$ is the membership function of the fuzzy set $A_i$, $x \in U$ is a generic element of the fuzzy set $A_i$, $\mu_{A_i}(x)$ is the degree of membership of $x$ to $A_i$. The integral should not be algebraically interpreted, this notation simply indicates that $A_i$ is a collection of ordered pairs.

Definition: (Song and Chissom) [21]. Let a time series $y(t) : t = \ldots, 0, 1, 2, \ldots$ and let the range of values of $y(t)$, a subset of real numbers $\mathbb{R}$, form the universe of discourse by which fuzzy sets $\mu_{A_i}, i = 1, \ldots, N$, are defined. If $F(t)$ is a collection of fuzzy sets in time then $F(t)$ is called a Fuzzy Time Series defined on $y(t)$.

### 2.2. Chen uni-variate fuzzy time series algorithm (1996)

In 1996, Chen proposed a model with simplified arithmetic operators, see Ref. [36], replacing the complicated max-min composition operations presented by Song and Chissom [21]. The basic algorithm as proposed by Chen is reviewed here.

**Step 1:** Define the universe of discourse as an interval, whose limits are defined from the range of values in the historical data (in-

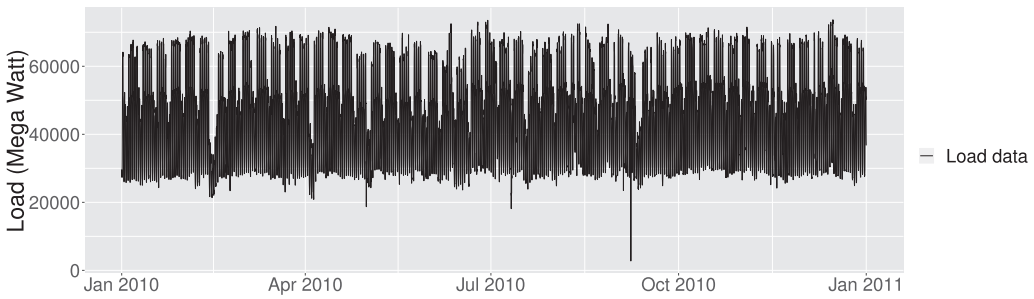


**Fig. 1.** Hourly load data of Malaysia of year 2010.

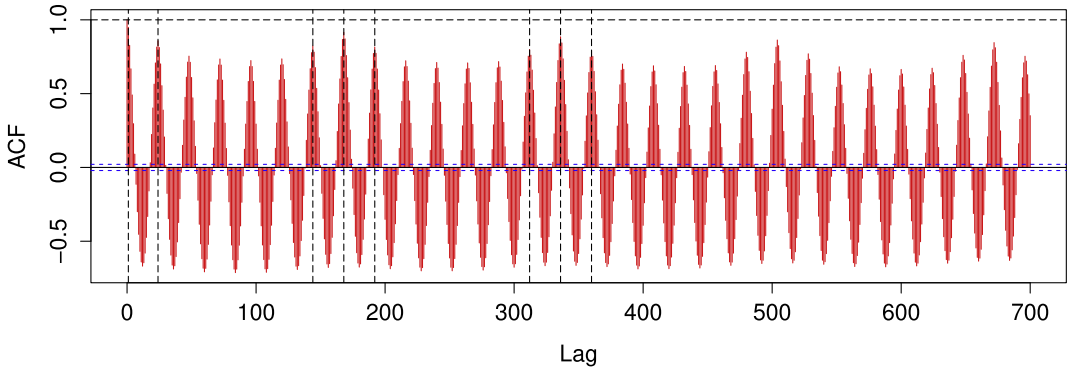


**Fig. 2.** Auto-correlation of the 2010 load data of Malaysia.

**Table 1**
STLF models.

| Model Category | Accuracy | Over-fitting | Easy of use |
|---|---|---|---|
| Classical method | low | No | No |
| AI based models | fair | Yes | No |
| Fuzzy Time Series | fair | No | No |
| Hybrid FTS | very good | No | No |

sample or training data).

**Step 2:** Partition $U$ into equal-length intervals as

$$U = \{u_1, u_2, \cdots, u_n\}$$

**Step 3:** Define fuzzy sets $A_i$ where $1 \le i \le n$ on $U$ and then fuzzify values in the in-sample data. If the value belongs to interval $u_i$, it is fuzzified to $A_i$.

**Step 4:** Establish first-order FLR, e.g., $F(t-1) \rightarrow F(t)$. By considering the historical fuzzified data, identify the patterns $A_i \rightarrow A_j$, where $1 \le i,j \le n$, in the data.

**Step 5:** Establish FLR Groups (FLRGs) using the recognized FLR. Those groups are established by grouping those FLR which have the same left hand side. For instance, given the following FLR,

$$\begin{aligned} &A_i \rightarrow A_j \\ &A_i \rightarrow A_k \\ &A_i \rightarrow A_m \\ &A_j \rightarrow A_n \\ &A_j \rightarrow A_m \\ &A_m \rightarrow A_i \end{aligned}$$

where $1 \le i,j,m \le n$, produce these FLRG:

$$\begin{aligned} &A_i \rightarrow A_j, A_k, A_m \\ &A_j \rightarrow A_n, A_m \\ &A_m \rightarrow A_i \end{aligned}$$

**Step 6:** Forecasting and defuzzification. Given the current value, calculate the forecast value based on the fuzzy sets in the right hand side of the FLRG. If at time $t$, $y(t)$ is fuzzified as $A_i$ and the RHS contains one fuzzy set in the sequence i.e. $A_i \rightarrow A_j$, where $1 \le i.j \le n$, then

$$f(t+1) = M_j$$

if at time $t$, the RHS contains more than one fuzzy set i.e. $A_i \rightarrow A_{j1}, A_{j2}, \ldots, A_{jm}$ then

$$f(t+1) = \frac{M_{j1} + M_{j2} + M_{j3} + \cdots + M_{jm}}{m}$$

if at time $t$, the RHS contains no fuzzy sets in the sequence i.e. $A_i \rightarrow \#$ then

$$f(t+1) = M_i$$

where $f(t)$ is the defuzzified value of the FTS $F(t)$ and $M_i$ corresponds to the defuzzified value of $A_i$, usually the midpoint of the interval associated to this fuzzy set.

In the proposed model, it just requires the fuzzification concept generating the FTS sequence and there will not be any need to establish FLRs, FLRGs, and defuzzification step. So by following Chen algorithm up to Step 3, the FTS can be generated, fulfilling our requirements for the proposed method. Later in Section 4, it will be discussed how the FTS can be used.

### 2.3. Convolutional neural networks

Convolutional Neural Networks (CNNs, or ConvNets) is a class of deep, feed-forward artificial neural networks that employ a variety of multilayer perceptrons designed to require minimal preprocessing. They were encouraged by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli just in a limited region of the visual field identified as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field [37].

#### 2.3.1. Architecture outline

Regular Neural Nets do not scale well to high dimensional data. For instance, an image of size, let us say $200 \times 200 \times 3$, would lead

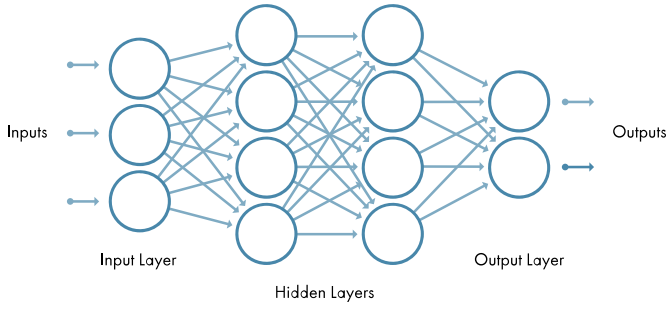


**Fig. 3.** An example of deep neural network.

to neurons that have $200 \times 200 \times 3 = 120,000$ weights. Moreover, we would almost certainly want to have some layers of such neurons, hence the parameters would add up very fast (see Fig. 3). Clearly, this full connectivity is inefficient and the huge number of parameters would quickly drive to over-fitting.

CNNs take benefit of the fact that the input consists of data and they constrain the architecture in a more practical way. In particular, CNNs include four different layers as follows:The Convolution layer: CNNs inherit their name from the "convolution" operator. The primary purpose of Convolution in case of a CNN is to extract features from the input data. Convolution preserves the spatial connection among data items by learning data features utilizing tiny squares of input data. Every data input can be counted as a matrix of some values. In CNN vocabulary, a 'filter' is the matrix that performed on input data, and the matrix formed by sliding the filter over the image and computing the dot product is named the 'Convolved Feature'. An additional operation called Rectified Linear Unit (ReLU) is applied after every Convolution operation. ReLU is a non-linear operation. Its output is defined by and shown in Fig. 4.

$$f(x) = \max(0, x) \tag{2}$$

ReLU is an element-wise process (implemented per pixel) and substitutes all negative values in the feature map by zero. The idea of ReLU is to introduce non-linearity in CNNs. Recently, ReLU has become very popular, because it was proved that it had a lot of improvement in convergence from the Hyperbolic Tangent function (see Experiment 1). This is mainly due to its simple form (see eq. (2)). Hence it avoids and rectifies the vanishing gradient problem. Almost all deep learning models use ReLU nowadays. In contrast, two other activation functions that were popular in the past, namely Sigmoid and Hyperbolic Tangent, are suffering from slow convergence and vanishing gradient problems, which cause lots of problems to train, degrade the accuracy and performance of a deep Neural Network Model. The other advantage of ReLU is sparsity. Sparsity occurs when $x \geq 0$. The more such units that exist in a layer the more sparse the resulting representation. Sigmoids and Hyperbolic Tangent, on the other hand, are constantly expected to produce some non-zero value resulting in dense representations. Sparse representations appear to be more advantageous than dense representations.

The Pooling layer: Spatial Pooling (also called sub-sampling or down-sampling) reduces the dimension of each feature map but in some way preserves the most valuable information. Spatial Pooling can be of various types: Max, Average, Sum etc. In the case of Max Pooling, a spatial area is determined and the largest element from the rectified feature map within that window is taken. The idea of Pooling is to frequently reduce the spatial size of the input representation [38].

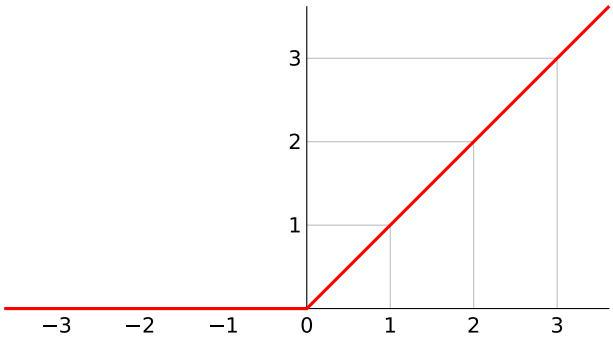


**Fig. 4.** Rectified linear unit.

- Pooling makes the input representations (feature dimension) smaller and more controllable
- Pooling reduces the number of parameters and calculations in the network, hence, controlling over-fitting
- Pooling makes the network invariant to small transformations, distortions, and translations in the input data (a small distortion in the input will not change the output of Pooling because we take the maximum/average value in a local neighborhood).

Fully Connected Layer: The Fully Connected layer is a traditional Multi-Layer Perceptron that employs a softmax activation function in the output layer. The term "Fully Connected" means that every neuron in the previous layer is connected to every neuron in the following layer. The output from the convolutional and pooling layers render high-level features of the input data. The idea of the Fully Connected layer is to use these features for classifying the input data into various classes in classification problems. For the regression problem, however, the last Fully Connected layer is just linked to a linear output [39].

Dropout layer: Dropout is a method applied to control over-fitting on neural networks by restricting complex co-adaptations on training data. In this regularization technique, some units (both hidden and visible), randomly, by a specific know rate will be dropped out, during the training phase [33].

#### 2.3.2. *CNNs architecture for regression problem*

As it is known, in CNN topology, the Convolution together with Pooling layers act as Feature Extractors from the input data (for example images). Because we are using CNNs for solving a regression problem, rather than a classification one, in our proposed method, the fully connected layer will be used as a classifier, except the final fully connected layer which will be used for value prediction instead of label class prediction.

### 2.4. *Fuzzy logic*

Fuzzy logic is a sort of many-valued logic in which the truth values of variables may be any real number between 0 and 1. It is running to handle the theory of partial truth, where the truth value may range within completely true and false. In contrast, in Boolean logic, the truth values of variables might only be the integer values 0 or 1 [40]. Additionally, if linguistic variables are applied, these degrees may be regulated by specific (membership) functions. The term fuzzy logic was introduced with the 1965 proposal of fuzzy set theory by Lotfi Zadeh [35]. Fuzzy logic had still been studied since the 1920s, as infinite-valued logic notably by Lukasiewicz and Tarski. Classical logic only allows inferences which are either true or false [41]. In our proposed methodology, however, fuzzy logic has been already embedded in FTS. So the proposed method implicitly benefits from its features.

## 3. Data and configurations

Short term load data is one of the most complicated kind of time series for forecasting. These data is considered as nonlinear with many intra-seasonality which make solving this problem even harder (see Fig. 2). Because of the difficulty that exists in using short term load data, and to show how proposed method was capable to solve complex forecasting problem, the authors have selected them to validate the proposed model. Moreover as discussed in introduction section, having accurate STLF is vital for most of power supply companies, and it was the second reason for choosing this case study. Therefore, the hourly load data of the power supply company of the city of Johor in Malaysia generated in 2009 and 2010 (see Fig. 1) were deployed to accomplish the aims of this

study. Temperature time series was coupled with hourly load data, to improve the accuracy of the model. These temperature time series were plotted in Figs. 5 and 6, respectively.

In addition to these informative plots, a short-listed of load data together with temperature and fuzzified data is presented in Table 2. It should be mentioned that for producing FTS, in this study, we implement the method presented in Ref. [42] for partitioning the universe of discourse. The data is split into train and test sets as detailed in Table 3.

### 3.1. Comparison criteria

The Absolute Percentage Error (APE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE), introduced in Eqs. (3)–(5), are the most important factors to use when comparing the outcomes of the models in STLF problems. In this study, the comparison relies on these metrics, which are described as follows:

$$\mathrm{APE}_i = \left|\frac{\mathrm{actual}(i) - \mathrm{forecasted}(i)}{\mathrm{actual}(i)}\right| \times 100 \tag{3}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{APE}_i \tag{4}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\mathrm{actual}(i) - \mathrm{forecasted}(i))^2} \tag{5}$$

In this study measures based on relative errors have also been used. To calculate MdRAE first $r_t$ must be obtained as follows:

$$r_t = \frac{|e_t|}{\mathrm{actual}(i) - f_i^*}$$

where $f_i^*$ is the predicted value obtained by using a reference model prediction (benchmark model). The main practice is to use a naive model as a reference model, i.e., $f_i^* = \mathrm{actual}(i-1)$. Then MdRAE is calculated as according to

$$\mathrm{MdRAE} = \mathrm{median}\,|r_i| \tag{6}$$

where $i = 1, 2, \ldots, n$.

Here, the actual $(i)$ is the real load value at time $i$, and the forecasted $(i)$ is the final predicted value at time $i$.

### 3.2. System configuration

To verify and validate the proposed method, a Keras version 2.1.5 platform with the back-end of Tensorflow version 1.8.0 on a PC with an Intel Core i7, 2.3 GHz CPU, 8 GB memory and 1 TB hard disk with a Linux operating system is utilized.

## 4. Proposed methodology

The proposed methodology of this study includes three phases. The first phase is about processing the time series to make them ready for converting multi-variate time series into multi-channel images. The second phase explains the way the input images are built from outputs of phase I and finally, in phase III, the topology of the proposed CNN is explained.

### 4.1. Phase I: data processing

In this section, the term $t$ refers to the current state of the time for time series of load, temperature and indices of fuzzy sets. Thus $t$ can be any time index from the beginning of training set plus image width size up to the length of training set. For training the proposed model, as with training all other models, the data up to time $t$ can be used to forecast the value of load for time $t+1$, which is assumed to be unknown (in our methodology only temporal time series from $t-n$ to $t$ will be used to calculate the forecast for each $t+1$, where $n$ will be defined later in this part). Now by this explanation, assume the current state of time series is $t$ and the forecast target state is $t+1$. For each time series at hand, e.g., original load data, temperature and indices of fuzzy sets in generated fuzzy time series, assign $j = 1, 2$ and 3 respectively and then, perform the following:

- set $j = 1$ and until $j \leq$ number of variables in multi-variate time series (in our use case it is 3), do:
  - select value of original load time series at time $t+1$ as a prediction (target value) that corresponds to those set of images which will be created in next steps.
  - decide about the size of images, let us say $n \times n$ (in our use case we select $n = 32$).
  - set $i = 0$.
  - while $t - i - (n-1) > 0$ do
    - starting from $t - i$, select $n$ former values in the sequence of time series and create a list of sequence of them, namely, $list_i$. So $list_i$ contains items from $item_{t-i-(n-1)}$ to $item_{t-i}$.
    - increasingly sort $list_i$ to create another sorted-list, i.e., $slist_i$. One simplified example with $n = 4$ can be:

$$list_i = [\,30361 \quad 29155 \quad 29155 \quad 28031\,]$$

and its corresponding sorted-list can be:

$$slist_i = [\,28031 \quad 29155 \quad 29155 \quad 30361\,]$$

- use both $list_i$ and $slist_i$ to establish a $matrix_{i,j}$ for $jth$ channel of image corresponding to index $i$ as follows:

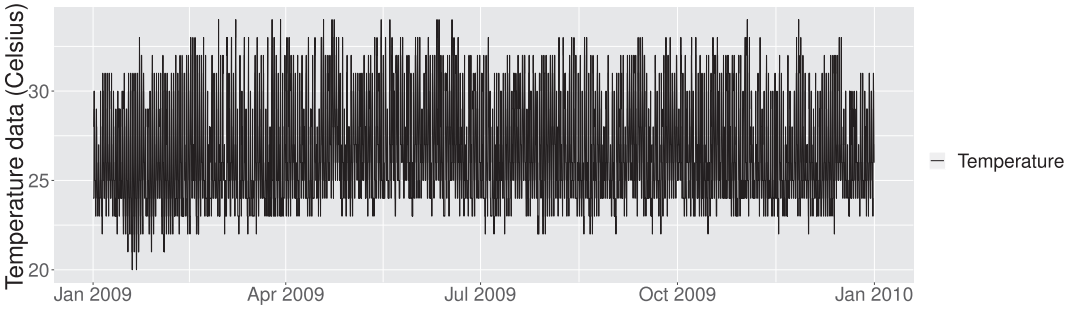


**Fig. 5.** Temperature data of the year 2009.

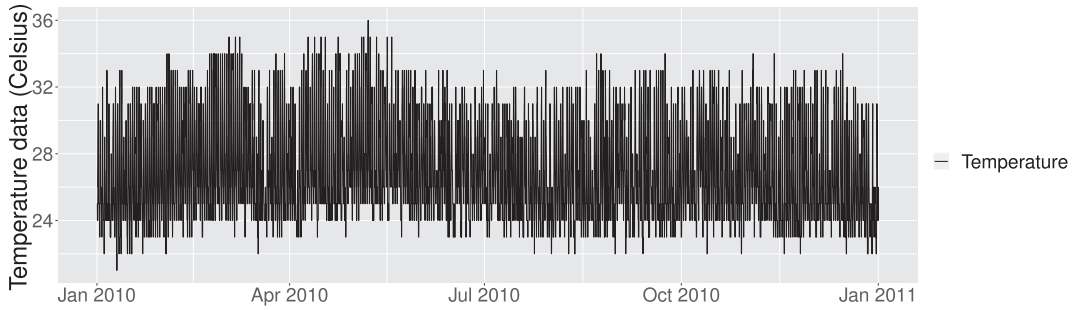


**Fig. 6.** Temperature data of the year 2010.

**Table 2**
Short-listed data of temperature, load, and fuzzified load.

| Date and Time index | Temperature (Celsius) | Load (MW) | Fuzzified |
|---|---|---|---|
| 01/01/09 01:00 a.m. | 24 | 30,360 | $A_2$ |
| 01/01/09 02:00 a.m. | 24 | 29,155 | $A_2$ |
| 01/01/09 03:00 a.m. | 24 | 28,086 | $A_2$ |
| 01/01/09 04:00 a.m. | 24 | 28,031 | $A_2$ |
| 01/01/09 05:00 a.m. | 24 | 27,730 | $A_1$ |
| 01/01/09 06:00 a.m. | 24 | 30,490 | $A_2$ |
| 01/01/09 07:00 a.m. | 24 | 37,081 | $A_4$ |
| 01/01/09 08:00 a.m. | 24 | 46,975 | $A_6$ |
| 01/01/09 09:00 a.m. | 25 | 57,261 | $A_8$ |
| 01/01/09 10:00 a.m. | 28 | 63,679 | $A_{10}$ |
| 01/01/09 11:00 a.m. | 30 | 65,123 | $A_{10}$ |
| 01/01/09 12:00 p.m. | 30 | 65,142 | $A_{10}$ |
| 01/01/09 01:00 p.m. | 30 | 64,745 | $A_{10}$ |
| 01/01/09 02:00 p.m. | 28 | 64,783 | $A_{10}$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 12/31/09 03:00 p.m. | 30 | 65,003 | $A_{10}$ |
| 12/31/09 04:00 p.m. | 30 | 60,655 | $A_{10}$ |
| 12/31/09 05:00 p.m. | 29 | 51,507 | $A_9$ |
| 12/31/09 06:00 p.m. | 28 | 50,743 | $A_7$ |
| 12/31/09 07:00 p.m. | 27 | 51,317 | $A_7$ |
| 12/31/09 08:00 p.m. | 27 | 48,414 | $A_7$ |
| 12/31/09 09:00 p.m. | 26 | 39,262 | $A_4$ |
| 12/31/09 10:00 p.m. | 26 | 35,809 | $A_3$ |
| 12/31/09 11:00 p.m. | 26 | 33,426 | $A_3$ |

- select *kth* element of $list_i : 1 \leq k \leq n$, i.e., $l_k$
- find the position of $l_k$ in $slist_i$ namely $p$
- assign a $vector_k$ with size $n$ in such a way that *pth* element of $vector_k$ will be 1 and other elements of $vector_k$ will be zero.
- vertically stack all obtained $vector_k : 1 \leq k \leq n$ to establish $matrix_{i,j}$.

Following our numeric examples of $list_i$ and $slist_i$, $vector_1$,$vector_2$, $vector_3$, and $vector_4$ will be:

$$vector_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$vector_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$vector_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$vector_4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

and after stacking these vectors the matrix is established as follows:

$$matrix_i = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

### 4.2. Phase II: creating image time series

As explained in former phase, for each step $i : t - n \leq i \leq t$, corresponding to each variable j, there is a $matrix_{i,j}$. After stacking these matrices (as channels of images) for each $i$, our image time series will be established. In fact, by stacking those matrices, in each $i$, we will have a multi-channel image. A real example of three matrices for $i = 8728$ and $n = 32$ of year 2010 can be seen in Figs. 7–9.

The following image is also generated from the fuzzy data presented in Fig. 9. As it can be seen in Fig. 10, the seasonality of time series is reflected and encapsulated inside the image.

### 4.3. Phase III: selecting the proper model for prediction

After preparing image series following last two phases, in this phase, a proper forecasting model is selected which can be used for

**Table 3**
Train and test sets for year 2009 and 2010.

| Year | | Malaysia 2009 | Malaysia 2010 | Number of items |
|---|---|---|---|---|
| Train set | start from | 01/01/09 01:00 a.m. | 01/01/10 01:00 a.m. | 7884 |
| | to | 11/25/09 11:00 a.m. | 11/25/10 11:00 a.m. | |
| Test set | start from | 11/25/09 12:00 p.m. | 11/25/10 12:00 p.m. | 876 |
| | to | 01/01/10 12:00 a.m. | 01/01/11 12:00 a.m. | |

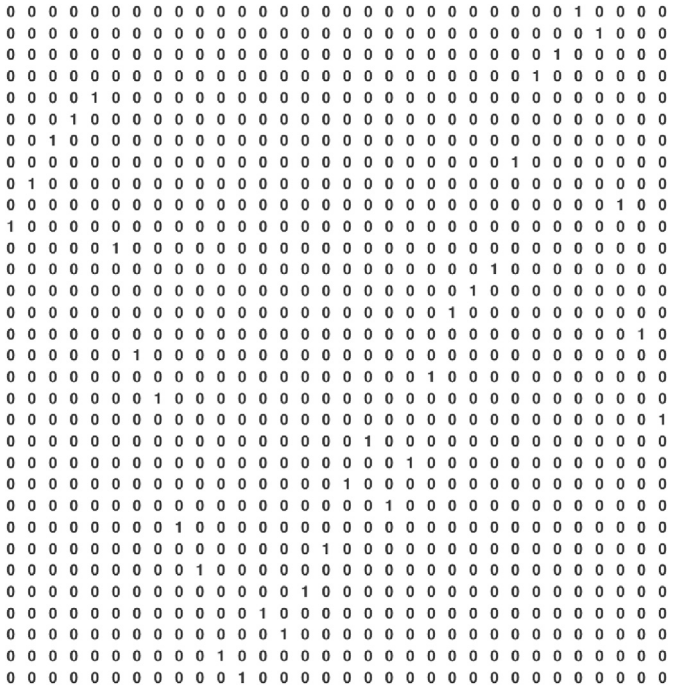

**Fig. 7.** Matrix of Hourly load data of year 2010 for $i = 8728$ and $n = 32$.

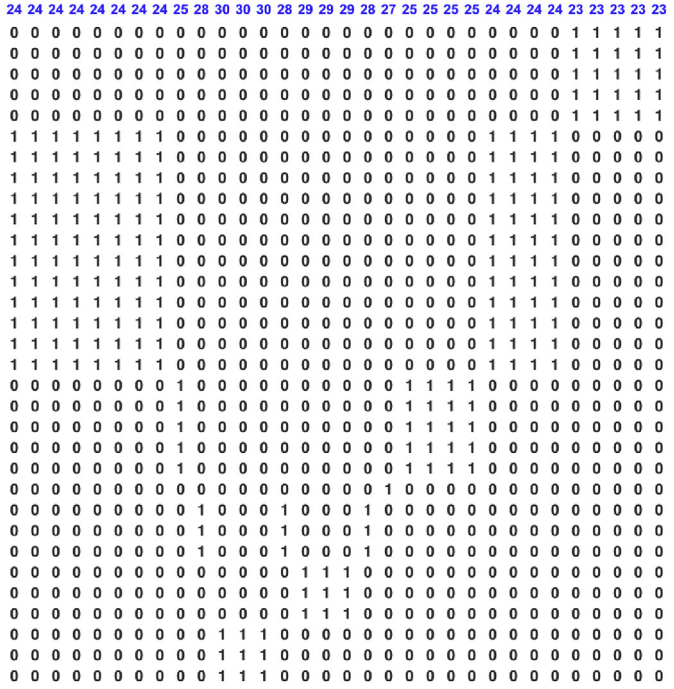

**Fig. 8.** Matrix of Hourly temperature data of year 2010 for $i = 8728$ and $n = 32$.

image series. For this aim we select a CNN model with an architecture as follows;

- image size = 32
- batch size = 100
- number of epochs = 20
- learning rate = 0.001

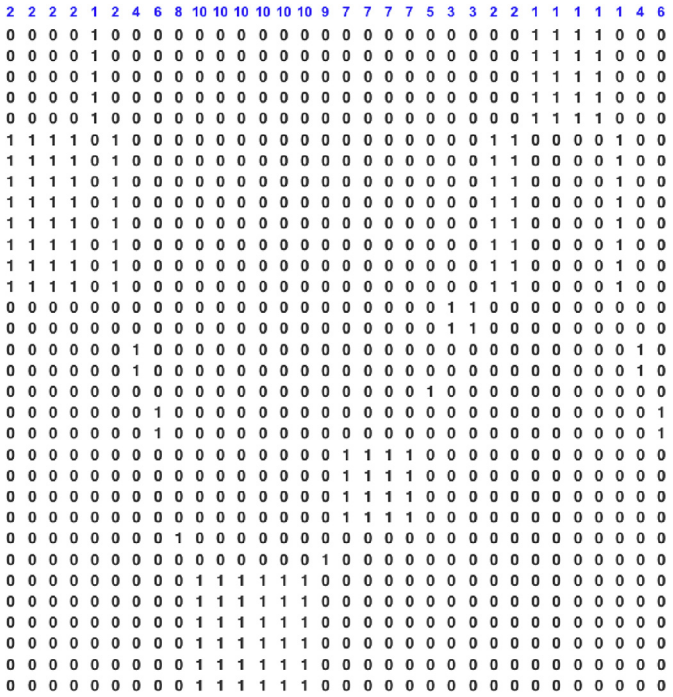

**Fig. 9.** Matrix of fuzzified hourly load data of year 2010 for $i = 8728$ and.$n = 32$.

- input shape will be [32, 32, number of channel], in our example number of channel = 3.

The network used has 2 convolutional layers and 5 fully connected ones. There is one dropout layer with the rate of 40%, after second fully connected, during the training, to avoid over-fitting. All convolutional layers are followed by Rectified Linear Units (ReLU), and the first two are followed by a 2 by 2 max pooling with a stride of 2. Finally, the training for Regression is performed with a 1-way summarizing function (see Fig. 11).

## 5. Results and discussions

To evaluate the effectiveness of the proposed method, in this Section we design four Experiments to test the model from the different point of views as follows:

### 5.1. Experiment 1: convergence time for various activation functions

As explained in Section 2, ReLU is one of the best activation function that is being used in many deep learning models. In this study, ReLU has been used, however we design this experiment to confirm other findings about the superiority of ReLU over other activation functions on our case studies.

As it is apparent from Table 4, the convergence time while using ReLU is much smaller than others (less than half the time compared with Hyperbolic Tangent). These results are in line with those of previous studies.

### 5.2. Experiment 2: determine the size of images

As it can be seen from the results of Table 5, the proposed method is not very sensitive to image sizes, if the image size is selected in the logical range. As an example, with an image size of 224, the MAPE in the testing set is 3.77 for the year 2009 which has

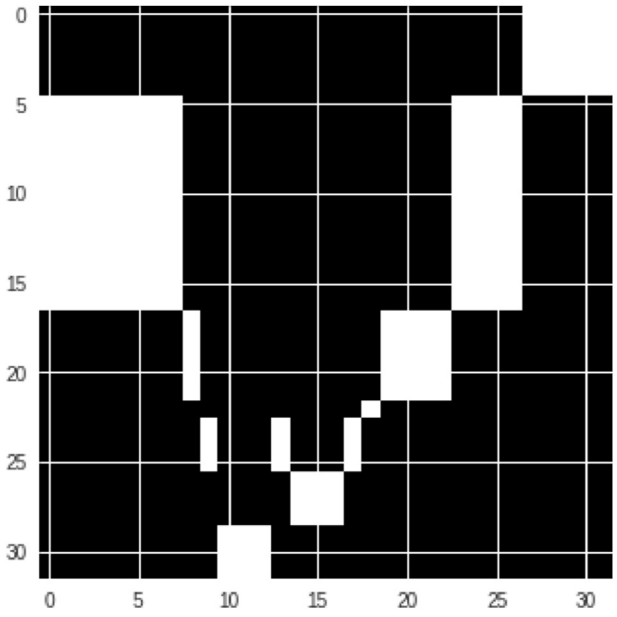


**Fig. 10.** Image of fuzzified hourly load data of year 2010 for $i = 8728$ and $n = 32$. Note: in Figs. 8 and 9, the first row is there just to be as a guideline, it does not belong to the matrix.

the small deviation (just 0.75) from the results when the image is 32. This shows the robustness of the way the data is explained by images and also the power of CNNs in mapping complex relationship between input images to output, with small errors.

### 5.3. Experiment 3: Significance of the input layers

In this experiment, it is shown that, for our use case, using all three input channels together yield better results. To facilitate presenting the results we are using the following abbreviations:

- Original Time Series = OTS
- Fuzzy Time Series = FTS
- Temperature Time Series = TTS

As it is evident from Table 6, the best possible scenario is to use all layers together. The second best results happened when OTS and FTS are used together. The method is so strong that even by using just images with the single channel, i.e., only OTS, it is able to produce considerable results.

### 5.4. Experiment 4: comparison of the accuracy of the state-of-the-art models on testing dataset

We repeated 30 times the experiments with the proposed methods for testing dataset and then calculated average MAPE, RMSE, and MdRAE. The results obtained with other STLF methods are gathered and presented in Tables 7–9. For developing Seasonal Autoregressive Integrated Moving Average (SARIMA) on data we

**Table 5**
MAPE of the proposed method based on various image size.

| Image size | 32 × 32 | 48 × 48 | 64 × 64 | 128 × 128 | 224 × 224 |
|---|---|---|---|---|---|
| Malaysia 2009 | 3.02 | 3.26 | 3.45 | 3.52 | 3.77 |
| Malaysia 2010 | 2.89 | 3.43 | 3.49 | 3.34 | 3.71 |

**Table 6**
MAPE of the proposed method based on using different image channel.

| Image type | OTS | OTS & FTS | OTS & FTS & TTS | FTS & TTS |
|---|---|---|---|---|
| Malaysia 2009 | 3.34 | 3.16 | 3.02 | 3.20 |
| Malaysia 2010 | 3.19 | 3.15 | 2.98 | 3.18 |

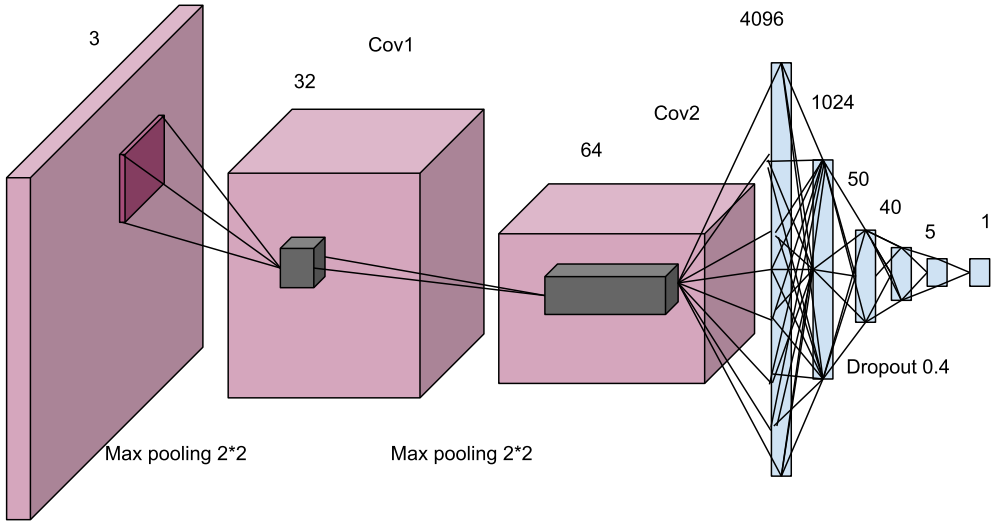


**Fig. 11.** CNN architecture used for STLF.

**Table 4**
Convergence time (in minutes) for various activation functions.

| Activation function | Sigmoid | Hyperbolic Tangent | ReLU |
|---|---|---|---|
| Convergence time for Malaysia 2009 train set | 52.12 | 45.32 | 17.23 |
| Convergence time for Malaysia 2010 train set | 54.51 | 47.43 | 18.01 |

employed one famous function in forecast package of R, namely, auto.arima, which conducts a search over best possible model within the same logical range of parameters [43]. We believe that the selected linear model, SARIMA(1,0,3)(2,1,2) [24] for Malaysia 2009 and SARIMA(1,0,1)(1,1,2) [24] for Malaysia 2010, by this approach could yield the best results among all linear models. Another selected model, in this experiment, is the Probabilistic Weighted Fuzzy Time Series (PWFTS) introduced in pyFTS [44]. pyFTS is a new FTS library that covers a wide range of models in this domain, which is recently equipped with hyperparameter optimization as well. Therefore, the reported results of PWFTS can be seen as the best possible outputs of FTS. In addition, other models of FTS were selected from pyFTS library in the experiment, i.e., Weighted Fuzzy Time Series (WFTS) and Integrated Weighted Fuzzy Time Series (IWFTS) and obtained results are collected in Tables 7–9 Besides the state-of-the-art linear and FTS models for STLF, we have tested a very important model of deep learning for sequence forecasting, namely Long Short-Term Memory (LSTM) networks. LSTM is a special kind of (Recurrent Neural Networks) RNNs, capable of learning long-term dependencies. They were introduced in Ref. [45] and were refined and popularized by many researchers and practitioners in a large variety of problems, especially, time series forecasting. In this study various settings were selected for LSTM and best results are produced when, the 168 (one week) former lags of time series for all original data, temperature, and fuzzy time series indices were selected. For LSTM model 1, 2 and 3, the 24, 48 and 72 former lags were selected as input. For all cases of LSTM, original data at time $t + 1$ was targeted to be predicted.

As the obtained results suggested, the best performance belonged to the proposed method, followed by LSTM and PWFTS. Even though LSTM was in the second rank in terms of accuracy in this experiment, the proposed method could achieve 22.84% better accuracy than LSTM, for the year 2009 as an example. It is about 27.81% better than PWFTS in terms of MAPE comparison. Figs. 12 and 13 provide a better vision for the top three STLF models used in this experiment. In addition, RMSE for proposed method is less than 460 units less than PWFTS for year 2010. For the year 2010, proposed method produced results with 20% better than LSTM using MdRAE metric.

### 5.5. Experiment 5: use residual and Visual Geometry Group networks

As it is shown in former section, the CNN with proposed topology already overcame some advanced time series models for STLF with better results. However, the next valid and true question is if the results can be improved by using more advanced complex and novel derivation of CNN models. This section is aimed to address this question. For doing so, we employed some of those models such as ResNet 50, VGG 16, etc., on the same data. In fact, ResNet 50 is the CNN that won ILSRVC 2015 competition and surpassed the human performance on ImageNet dataset [46]. ResNet 50 is one of the methods employed in experiments. In this experiment the following scenarios of using ResNet 50 have been tested on data:

- ResNet 50, fully train: In this scenario, we tuned all required parameters, during the training phase, and test the model on test data.
- ResNet 50, last layer train: In this case, we used transfer learning technique, whereby we only used all pre-tuned weights, except the last layer. In this case last fully connected layer has been trained, and the model then tested.
- ResNet 50, change top to use input images with the size of $32 \times 32 \times 3$: The architecture of ResNet 50 does not allow to use images with the height and weight less than 224. However, we developed another architecture of ResNet by changing the top layer to accept images with the smaller size. In this case, just 5 blocks of ResNet have been employed.

Another main and new methodology derived from CNNs is VGG 16, (also called OxfordNet) which is an architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014. To this day it is still considered to be an excellent vision model, although it has been somewhat outperformed by more advanced models such as Inception and ResNet. It has been employed in this experiment by following the same scenarios that have been used in ResNet in former part [47].

- VGG16, fully train
- VGG16, last layer train
- VGG16, change top to use input 3 layer
- VGG16, change top to use input 3 layers, with just 5 blocks

Although it has been expected that these advanced deep learning models could produce good results, both ResNet 50 and

**Table 7**
Evaluation of average MAPE of the proposed method and some other models in STLF.

| Methods | Malaysia 2009 | Malaysia 2010 |
|---|---|---|
| FTS-CNN (proposed method) | 3.02 | 2.89 |
| SARIMA | 4.68 | 4.23 |
| PWFTS | 3.86 | 4.00 |
| WFTS with differences | 5.33 | 9.09 |
| IWFTS with differences | 5.32 | 7.69 |
| PWFTS with differences | 4.59 | 5.83 |
| LSTM | 3.71 | 3.45 |
| LSTM model 1 | 4.55 | 4.21 |
| LSTM model 2 | 4.11 | 4.23 |
| LSTM model 3 | 3.93 | 3.88 |

**Table 8**
Evaluation of average RMSE of the proposed method and some other models in STLF.

| Methods | Malaysia 2009 | Malaysia 2010 |
|---|---|---|
| FTS-CNN (proposed method) | 1777.99 | 1702.70 |
| SARIMA | 2763.66 | 2501.25 |
| PWFTS | 2230.91 | 2162.57 |
| WFTS with differences | 2930.36 | 4419.11 |
| IWFTS with differences | 2961.17 | 4663.17 |
| PWFTS with differences | 2987.40 | 3797.35 |
| LSTM | 2194.19 | 2037.49 |
| LSTM model 1 | 2689.42 | 2044.68 |
| LSTM model 2 | 2413.60 | 2483.71 |
| LSTM model 3 | 2317.88 | 2279.23 |

**Table 9**
Evaluation of average MdRAE of the proposed method and some other models in STLF.

| Methods | Malaysia 2009 | Malaysia 2010 |
|---|---|---|
| FTS-CNN (proposed method) | 0.4857 | 0.4537 |
| SARIMA | 0.7504 | 0.6655 |
| PWFTS | 0.6107 | 0.6319 |
| WFTS with differences | 1.0838 | 2.1105 |
| IWFTS with differences | 1.0780 | 1.2185 |
| PWFTS with differences | 0.5284 | 0.7638 |
| LSTM | 0.5981 | 0.5435 |
| LSTM model 1 | 0.7313 | 0.6622 |
| LSTM model 2 | 0.6610 | 0.6673 |
| LSTM model 3 | 0.6273 | 0.5991 |

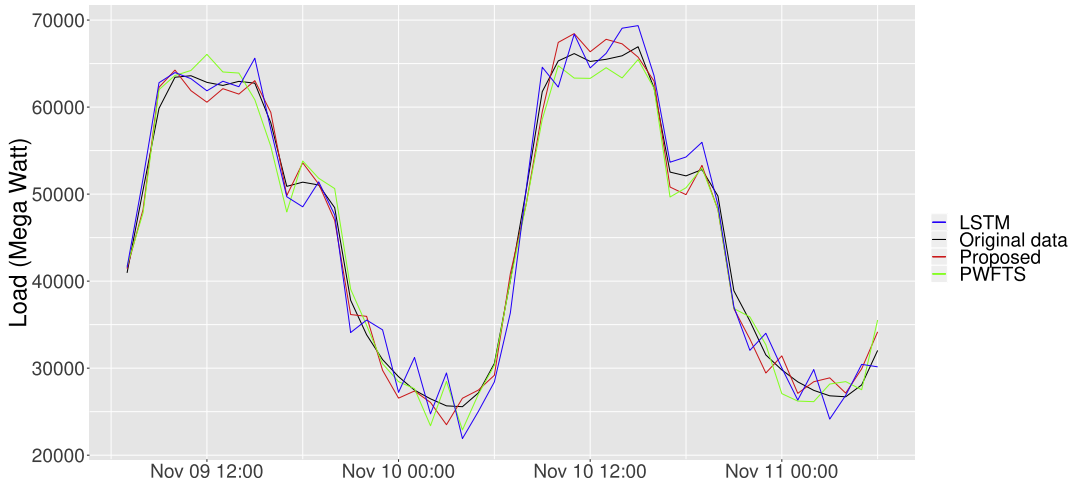


**Fig. 12.** Forecasted and actual time series comparison among top three methods for test dataset of year 2010.

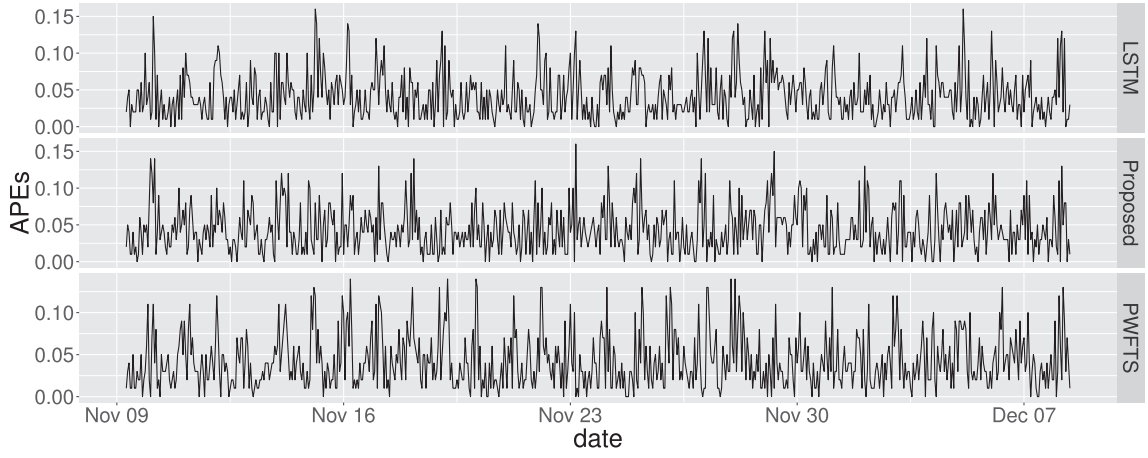


**Fig. 13.** APEs comparison among top three methods for test dataset.

VGG 16 had failed in this regards and the outcome is counterintuitive. Next section has provided some explanation about possible reasons.

### *5.5.1. Interpretation of the outcome of ResNet 50 and VGG 16 on data*

Findings from the former section suggest that the main reason for the fails was the small size of the training dataset. The small size of the dataset meant that it was not possible to tune the huge number of parameters that those advanced models required. Our dataset for each year of load data contains about 8000 images which are not enough for training very deep CNNs such as ResNet (for example complexity ResNet 34 can be seen in Fig. 14).

Further data collection is required to determine exactly if there is any possibility to improve the results by using ResNet and VGG.

Another major source of uncertainty lies in the necessity to employ too many parameters for our simple image manipulating problem. The proposed STLF use case, in comparison with many

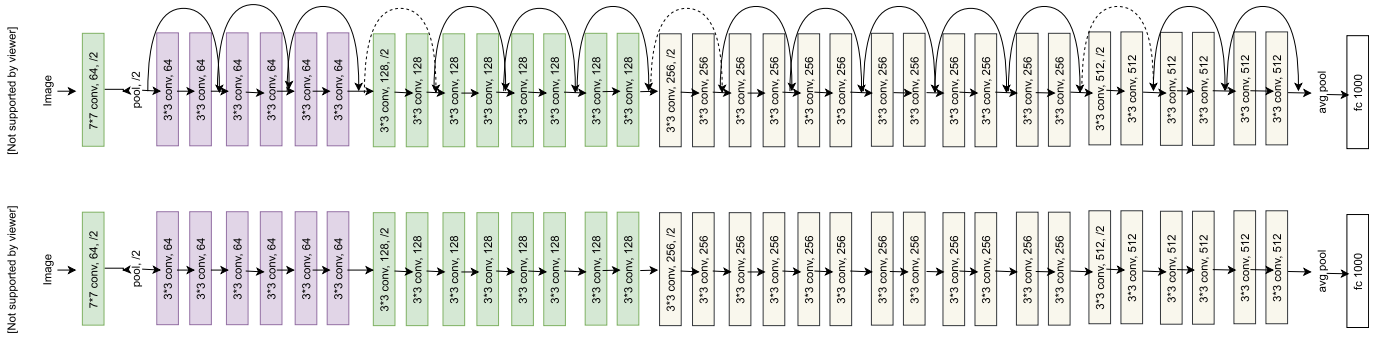


**Fig. 14.** Resnet architecture avoids the vanishing data problem (image from ResNet).

complicated computer vision applications, such as object detection and face recognition, image classification problems, which involve handling various images with different sizes, colors, objects that ResNet and VGG can solve, is too simple. The variation of the images in our use case is not that much, so there is no need to tune many parameters (for instance for ResNet 50 there are 25,610,216 parameters to be tuned). In our last experiment, even though the results were not satisfactory to be reported, simplified version of ResNet and VGG with fewer layers, i.e., 5 blocks, produced better results than full ResNet 50. This finding is also accordant with the parsimony rule, which says that the simplest solution tends to be the right one.

## 6. Conclusions

A new approach was suggested in this paper for short-term load forecasting (STLF) on the basis of a hybrid model of CNN and FTS. It was shown that the idea for converting multi-variate time series into images and then using CNNs for STLF can be considered to be useful and promising. In addition, the fuzzy logic which is implicitly used while constructing FTS from the original load data helped to decrease the negative effect of over-fitting, which was reflected in test results. This study attempted to advance the new idea of using images instead of time series. The results showed that this idea indeed succeeded in STLF. A further important finding was the importance of the temperature data that could be coupled with fuzzified information in improving the performance of the model. The findings of this research can be used to reference when performing further studies on the utilization of revised proposed methods by using various type CNNs with different typologies as well as other type of FTSs, such as weighted FTSa for other time series applications, such as mid-term and long-term load forecasting, stock index and temperature forecasting, number of tourism forecasting, etc. It is vividly clear that some other potential future studies can follow by applying a similar set up as the one introduced in this paper.

## Acknowledgement

This research was supported by Avenue Code and the following Brazilian agencies:

- National Council for Scientific and Technological Development (Abbreviation, CNPq). Grants no. 405840/2017-9 and no. 306850/2016-8.
- Coordination for the Improvement of Higher Education Personnel (Abbreviation, CAPES).
- Research Support Foundation of the State of Minas Gerais (Abbreviation, FAPEMIG).

The authors also would like to express their deepest gratitude to Universiti Teknologi Malaysia through grant number R.J130000.7301.4B359for their funding of this research.

## References

[1] Gross G, Galiana F. Short-term load forecasting. Proc IEEE 1987;75(12):1558–73. https://doi.org/10.1109/proc.1987.13927. https://doi.org/10.1109/proc.1987.13927.

[2] Al-Hamadi H, Soliman S. Short-term electric load forecasting based on kalman filtering algorithm with moving window weather and load model. Electr Power Syst Res 2004;68(1):47–59. https://doi.org/10.1016/s0378-7796(03)00150-0. https://doi.org/10.1016/s0378-7796(03)00150-0.

[3] Song K-B, Baek Y-S, Hong D, Jang G. Short-term load forecasting for the holidays using fuzzy linear regression method. IEEE Trans Power Syst 2005;20(1):96–101. https://doi.org/10.1109/tpwrs.2004.835632. https://doi.org/10.1109/tpwrs.2004.835632.

[4] Vähäkyla P, Hakonen E, Léman P. Short-term forecasting of grid load using box-jenkins techniques. Int J Electr Power Energy Syst 1980;2(1):29–34. https://doi.org/10.1016/0142-0615(80)90004-6. https://doi.org/10.1016/0142-0615(80)90004-6.

[5] Ranaweera D, Hubele N, Karady G. Fuzzy logic for short term load forecasting. Int J Electr Power Energy Syst 1996;18(4):215–22. https://doi.org/10.1016/0142-0615(95)00060-7. https://doi.org/10.1016/0142-0615(95)00060-7.

[6] He Y, Xu Q, Wan J, Yang S. Short-term power load probability density forecasting based on quantile regression neural network and triangle kernel function. Energy 2016;114:498–512. https://doi.org/10.1016/j.energy.2016.08.023. https://doi.org/10.1016/j.energy.2016.08.023.

[7] AMJADY N, KEYNIA F. Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm. Energy 2009;34(1):46–57. https://doi.org/10.1016/j.energy.2008.09.020. https://doi.org/10.1016/j.energy.2008.09.020.

[8] Ghofrani M, Ghayekhloo M, Arabali A, Ghayekhloo A. A hybrid short-term load forecasting with a new input selection framework. Energy 2015;81:777–86. https://doi.org/10.1016/j.energy.2015.01.028. https://doi.org/10.1016/j.energy.2015.01.028.

[9] Fan G-F, Peng L-L, Hong W-C. Short term load forecasting based on phase space reconstruction algorithm and bi-square kernel regression model. Appl Energy 2018;224:13–33. https://doi.org/10.1016/j.apenergy.2018.04.075. https://doi.org/10.1016/j.apenergy.2018.04.075.

[10] Mamlook R, Badran O, Abdulhadi E. A fuzzy inference model for short-term load forecasting. Energy Policy 2009;37(4):1239–48. https://doi.org/10.1016/j.enpol.2008.10.051. https://doi.org/10.1016/j.enpol.2008.10.051.

[11] Metaxiotis K, Kagiannas A, Askounis D, Psarras J. Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher. Energy Convers Manag 2003;44(9):1525–34. https://doi.org/10.1016/s0196-8904(02)00148-6. https://doi.org/10.1016/s0196-8904(02)00148-6.

[12] Fukushima K, Miyake S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural Nets. Springer Berlin Heidelberg; 1982. p. 267–85. https://doi.org/10.1007/978-3-642-46466-9_18. https://doi.org/10.1007/978-3-642-46466-9_18.

[13] LeCun Y, Haffner P, Bottou L, Bengio Y. Object recognition with gradient-based learning. In: Shape, contour and grouping in computer vision. Springer Berlin Heidelberg; 1999. p. 319–45. https://doi.org/10.1007/3-540-46805-6_19. https://doi.org/10.1007/3-540-46805-6_19.

[14] Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2013. arXiv:1312.6229.

[15] Abdulnabi AH, Wang G, Lu J, Jia K. Multi-task CNN model for attribute prediction. IEEE Trans Multimed 2015;17(11):1949–59. https://doi.org/10.1109/tmm.2015.2477680. https://doi.org/10.1109/tmm.2015.2477680.

[16] Levi G, Hassncer T. Age and gender classification using convolutional neural networks. In: 2015 IEEE conference on computer vision and pattern recognition workshops (CVPRW). IEEE; 2015. https://doi.org/10.1109/cvprw.2015.7301352. https://doi.org/10.1109/cvprw.2015.7301352.

[17] Xie L, Wang J, Wei Z, Wang M, Tian Q. DisturbLabel: Regularizing cnn on the loss layer. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). IEEE; 2016. https://doi.org/10.1109/cvpr.2016.514. https://doi.org/10.1109/cvpr.2016.514.

[18] Burnham KP, Anderson DR, editors. Model selection and multimodel inference. New York: Springer; 2004. https://doi.org/10.1007/b97636. https://doi.org/10.1007/b97636.

[19] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016. http://www.deeplearningbook.org.

[20] Guo Y, Liu Y, Oerlemans A, Lao S, Wu S, Lew MS. Deep learning for visual understanding: A review. Neurocomputing 2016;187:27–48. https://doi.org/10.1016/j.neucom.2015.09.116. https://doi.org/10.1016/j.neucom.2015.09.116.

[21] Song Q, Chissom BS. Forecasting enrollments with fuzzy time series — part i. Fuzzy Sets Syst 1993;54(1):1–9. https://doi.org/10.1016/0165-0114(93)90355-l. https://doi.org/10.1016/0165-0114(93)90355-l.

[22] Yu H-K. Weighted fuzzy time series models for TAIEX forecasting. Phys Stat Mech Appl 2005;349(3–4):609–24. https://doi.org/10.1016/j.physa.2004.11.006. https://doi.org/10.1016/j.physa.2004.11.006.

[23] Wang N-Y, Chen S-M. Temperature prediction and TAIFEX forecasting based on automatic clustering techniques and two-factors high-order fuzzy time series. Expert Syst Appl 2009;36(2):2143–54. https://doi.org/10.1016/j.eswa.2007.12.013. https://doi.org/10.1016/j.eswa.2007.12.013.

[24] Sadaei HJ, Enayatifar R, Abdullah AH, Gani A. Short-term load forecasting using a hybrid model with a refined exponentially weighted fuzzy time series and an improved harmony search. Int J Electr Power Energy Syst 2014;62:118–29. https://doi.org/10.1016/j.ijepes.2014.04.026. https://doi.org/10.1016/j.ijepes.2014.04.026.

[25] Sadaei HJ, Guimarães FG, da Silva CJ, Lee MH, Eslami T. Short-term load forecasting method based on fuzzy time series, seasonality and long memory process. Int J Approx Reason 2017;83:196–217. https://doi.org/10.1016/j.ijar.2017.01.006. https://doi.org/10.1016/j.ijar.2017.01.006.

[26] Efendi R, Ismail Z, Deris MM. A new linguistic out-sample approach of fuzzy time series for daily forecasting of malaysian electricity load demand. Appl Soft Comput 2015;28:422–30. https://doi.org/10.1016/j.asoc.2014.11.043. https://doi.org/10.1016/j.asoc.2014.11.043.

[27] Enayatifar R, Sadaei HJ, Abdullah AH, Gani A. Imperialist competitive algorithm combined with refined high-order weighted fuzzy time series (RHWFTS–ICA) for short term load forecasting. Energy Convers Manag

2013;76:1104–16. https://doi.org/10.1016/j.enconman.2013.08.039. https://doi.org/10.1016/j.enconman.2013.08.039.
[28] Lee W-J, Hong J. A hybrid dynamic and fuzzy time series model for mid-term power load forecasting. Int J Electr Power Energy Syst 2015;64:1057–62. https://doi.org/10.1016/j.ijepes.2014.08.006. https://doi.org/10.1016/j.ijepes.2014.08.006.
[29] Taylor JW, McSharry PE. Short-term load forecasting methods: an evaluation based on european data. IEEE Trans Power Syst 2007;22(4):2213–9. https://doi.org/10.1109/tpwrs.2007.907583. https://doi.org/10.1109/tpwrs.2007.907583.
[30] Taylor JW. Short-term electricity demand forecasting using double seasonal exponential smoothing. J Oper Res Soc 2003;54(8):799–805. http://www.jstor.org/stable/4101650.
[31] Huang S-J, Shih K-R. Short-term load forecasting via ARMA model identification including non-Gaussian process considerations. IEEE Trans Power Syst 2003;18(2):673–9. https://doi.org/10.1109/tpwrs.2003.811010. https://doi.org/10.1109/tpwrs.2003.811010.
[32] Gooijer JGD, Hyndman RJ. 25 years of time series forecasting. Int J Forecast 2006;22(3):443–73. https://doi.org/10.1016/j.ijforecast.2006.01.001. https://doi.org/10.1016/j.ijforecast.2006.01.001.
[33] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15:1929–58. http://jmlr.org/papers/v15/srivastava14a.html.
[34] Song Q, Chissom BS. Forecasting enrollments with fuzzy time series — part II. Fuzzy Sets Syst 1994;62(1):1–8. https://doi.org/10.1016/0165-0114(94)90067-1. https://doi.org/10.1016/0165-0114(94)90067-1.
[35] Zadeh L. Fuzzy Sets, Inf Control 1965;8(3):338–53. https://doi.org/10.1016/s0019-9958(65)90241-x. https://doi.org/10.1016/s0019-9958(65)90241-x.
[36] Chen S-M. Forecasting enrollments based on fuzzy time series. Fuzzy Sets Syst 1996;81(3):311–9. https://doi.org/10.1016/0165-0114(95)00220-0. https://doi.org/10.1016/0165-0114(95)00220-0.
[37] Sermanet P, Chintala S, LeCun Y. Convolutional neural networks applied to house numbers digit classification. 2012. arXiv:1204.3968.
[38] Zeiler MD, Fergus R. Stochastic pooling for regularization of deep convolutional neural networks. 2013. arXiv:1301.3557.
[39] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th conference on neural information processing systems, vol. 1. USA: Curran Associates Inc.; 2012. p. 1097–105. NIPS'12, http://dl.acm.org/citation.cfm?id=2999134.2999257.
[40] Cornelis C, Deschrijver G, Kerre EE. Implication in intuitionistic fuzzy and interval-valued fuzzy set theory: construction, classification, application. Int J Approx Reason 2004;35(1):55–95. https://doi.org/10.1016/s0888-613x(03)00072-0. https://doi.org/10.1016/s0888-613x(03)00072-0.
[41] Hájek P. Metamathematics of fuzzy logic. Springer Netherlands; 1998. https://doi.org/10.1007/978-94-011-5300-3. https://doi.org/10.1007/978-94-011-5300-3.
[42] Lee MH, Sadaei HJ, Suhartono. Improving TAIEX forecasting using fuzzy time series with Box–Cox power transformation. J Appl Stat 2013;40(11):2407–22. https://doi.org/10.1080/02664763.2013.817548. https://doi.org/10.1080/02664763.2013.817548.
[43] R. J. Hyndman, Y. Khandakar, Automatic time series forecasting: the forecast Package for R, J Stat Softw 27 (3). doi:10.18637/jss.v027.i03. URL https://doi.org/10.18637/jss.v027.i03
[44] De Lima E Silva Petrônio Cândido. Petroniocandido/pyfts: Export for zenodo doi generation (ref. pkg1.2.3). 2018. https://doi.org/10.5281/zenodo.1194859. https://zenodo.org/record/1194859.
[45] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735. https://doi.org/10.1162/neco.1997.9.8.1735.
[46] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). IEEE; 2016. https://doi.org/10.1109/cvpr.2016.90. https://doi.org/10.1109/cvpr.2016.90.
[47] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. arXiv:1409.1556.