Int. J. Fuzzy Syst.
https://doi.org/10.1007/s40815-023-01629-4

An Application of Fuzzy Symbolic Time-Series for Energy Demand Forecasting

D. Criado-Ramón[1] · L.G.B. Ruiz[2] · M. C. Pegalajar[1]

Received: 5 June 2023 / Revised: 9 October 2023 / Accepted: 19 October 2023

Abstract In this paper, we present a new fuzzy symbolization technique for energy load forecasting with neural networks, FPLS-Sym. Symbolization techniques transform a numerical time series into a smaller string of symbols, providing a high-level representation of time series by combining segmentation, aggregation and discretization. The dimensional reduction obtained with symbolization can speed up substantially the time required to train neural networks, however, it can also lead to considerable information losses that could lead to a less accurate forecast. FPLS-Sym introduces the use of fuzzy logic in the discretization process, maintaining more information about each segment of the neural network at the expense of requiring more space in memory. Extensive experimentation was made to evaluate FPLS-Sym with various neural-network-based models, including different neural network architectures and activation functions. The evaluation was done with energy demand data from Spain taken from 2009 to 2019. Results show that FPLS-Sym provides better quality metrics than other symbolization techniques and outperforms the use of the standard numerical time series representation in both quality metrics and training time.

Keywords Time series forecasting · Fuzzy logic · Symbolic representation · Energy demand · Artificial neural networks

1 Introduction

With all the technological advances in the last few decades, many real-life sectors generate massive amounts of temporal data daily, such as healthcare, finance, or energy. In the energy sector, accurately forecasting energy demand is critical in planning energy production and distribution. Processing this massive amount of data is not a trivial task. Therefore, it is frequent to use high-performance computational resources, such as clusters or GPUs; or to generate and use high-level representations of this data that allow for faster computations, such as symbolization.

Symbolization techniques provide a lower-length symbolic representation of time series using aggregation and discretization. The main challenge for a symbolization technique is to reduce the time series length as much as possible while not losing any relevant information. The first proposal of a symbolization technique is Symbolic Aggregate approXimation (SAX) [1], and it is still the most widely used symbolization technique. SAX splits the time series into equidistant segments using Piecewise Approximate Aggregation (PAA) [2] and transforms the mean value of each segment to a symbol. Each symbol in SAX represents an equiprobable interval assuming a normal distribution. Many other variants of the SAX idea have been proposed to specialize this technique for different

D. Criado-Ramón, L.G.B. Ruiz, and M.C. Pegalajar have contributed equally to this work.

✉ D. Criado-Ramón
dcriado@ugr.es

L.G.B. Ruiz
bacaruiz@ugr.es

M. C. Pegalajar
mcarmen@decsai.ugr.es

[1] Department of Computer Science and Artificial Intelligence, University of Granada, c/Periodista Daniel Saucedo Aranda s.n, 18014 Granada, Andalusia, Spain

[2] Department of Software Engineering, University of Granada, c/Periodista Daniel Saucedo Aranda s.n, 18014 Granada, Andalusia, Spain

Published online: 25 January 2024

fields or to address some of its main drawbacks. ESAX [3] was created to be used in the finance field and also preserves the maximum and minimum from each segment, as the authors considered that only preserving the mean value when working with financial data was insufficient. Adaptive SAX (aSAX) [4] was created to remove the time series normality assumption from SAX. In the energy field, it is common to use symbolization techniques for pattern-related tasks such as pattern extraction [5] and anomaly detection based on patterns [6, 7]. Still, they are not commonly used for the forecasting task [8].

Many different forecasting models have been used for energy forecasting over the last few decades. While classical models such as ARIMA have been used to forecast energy in various studies [9, 10], most recent works use neural networks and hybrid models [11]. Several different neural network architectures have been previously evaluated under different circumstances. Bagnasco et al. [12] used a multi-layer perceptron neural network to forecast energy consumption in a hospital in 2015. Naji et al. [13] used an extreme learning machine to predict energy consumption in buildings in 2016. In 2019 [14], a methodology to create ensembles of wavenets was proposed. The methodology was evaluated with hourly load datasets from Italy and the US. In 2020, Sajjad et al. [15] used a combination of Convolutional Neural Networks (CNN) and Gated Recurrent Unit (GRU) layers to forecast residential loads. In 2021, Zhang et al. [16], proposed a multi-layer model with CNN and Seq2Seq to simultaneously predict three different loads (cooling, heating, and electricity) of a Chinese industrial park. Hybrid approaches in the energy field mainly use combinations of clustering and other methods and ensembles. In 2011 [17], a hybrid model with K-means and pattern-based search forecasting was presented with remarkable results while forecasting energy data. In 2020 [18], a model using clustering and ARIMA was proposed to predict energy in buildings. Furthermore, an improved version of the K-means pattern-based forecasting model was presented for distributed computation with Spark the same year [19]. In 2022 [20], a hybrid model combining singular spectrum analysis and parallel long short term memory neural networks presented great results in building energy forecasting in comparison with other models. In 2023 [21], a theory-guided deep neural network using Attention, Long-Short Term Memory (LSTM) layers and CNN layers was presented for solar power forecasting. The theory-guided module of the framework consists of expert-provided photovoltaic power generation constraints that penalize the loss function of the neural network when they are not met. Results show that this approach outperformed several other deep learning alternatives to predict solar power generation in Asia.

However, even though the most accurate results are usually provided by neural network models, they can still be challenging to use due to the large amount of data and time required to train them. This can be a significant issue in real-time decision-making, where the model may need to be retrained frequently to provide the most accurate forecasts.

In order to address this issue, we found out in our previous study [8] that symbolization techniques were a powerful alternative time series representation, capable of providing faster training times although not yielding the same level of accuracy. Thus, in this study, we have developed and evaluated a new fuzzy symbolic time series representation to preserve more information about each segment to provide more accurate forecasts while still being faster than models that use the original time series without any dimensionality reduction. More specifically, this work provides the following contributions to the field:

1. We present a new symbolization technique, the first one that uses a fuzzy representation to preserve more information.
2. We provide a detailed analysis with statistical tests to evaluate whether our proposal is consistently better than previous symbolization techniques regardless of the neural network configuration used.
3. We evaluate the effect of using three different symbolization techniques in four neural network architectures using a publicly available big data dataset.

This manuscript is structured as follows: Sect. 2 provides the theoretical background for the methods used in this paper. Section 3 presents our fuzzy symbolization technique. Section 4 describes the experiments done to evaluate the performance of the proposed method. Section 5 analyzes the results obtained in those experiments and, lastly, Sect. 6 draws the most relevant accomplishments of our work and proposes future lines of research.

2 Background

2.1 Symbolization Techniques

Numerosity reduction techniques reduce data volume by using alternative smaller data representations. In the case of univariate time series, the use of this kind of technique would result in a new time series with the same number of variables but fewer observations.

Time series symbolization is a numerosity reduction technique that transforms a raw numerical time series $T = [T_0, T_1, T_2, \ldots, T_n]$ to a sequence of symbols of lower length $S = [S_0, S_1, S_2, \ldots, S_m]$, usually combining aggregation and

discretization. Any symbolization technique can be divided into the following components:

1. How to reduce the length of the time series. This step is usually done by splitting the time series into multiple segments [1, 3, 4, 22, 23].
2. Which information must be preserved from each segment. It may be a simple statistical value such as mean [1, 4], maximum or minimum, multiple statistical values [3] or something more sophisticated such as the linear regression of the segment [22, 23].
3. How to transform the preserved values into a symbolic string. This may be obtained via expert knowledge [22], some specific criteria such as probability distribution [1, 3, 22] or even optimization algorithms [4].
4. How long and how many symbols can be used for the symbolic representation. Most symbolization techniques provide this as a parameter that the user must decide [1, 3, 4, 22, 23].

SAX [1] was the first symbolization technique published and is still the most widely used. Segmentation in SAX is done using Piecewise Approximate Aggregation (PAA), splitting the time series into equidistant segments. The mean value from each segment is preserved and the discretization is made assuming the time series follows a normal distribution and each symbol of the symbols covers an equiprobable interval of values for the mean of the segments. The size of the segments and the number of symbols are provided by the user.

Many other symbolization techniques have been proposed based on the idea of SAX. Many authors claim that SAX does not preserve enough information as it just uses the mean value [3, 22, 23]. Extended SAX (ESAX) [3] uses three symbols per segment in order to preserve the mean, maximum and minimum of each segment. Trend-based SAX (TSAX) [22] and TFSAX [23] are different alternatives to add an extra symbol that represents the trend of the segments. Adaptive SAX (aSAX) [4] uses the Lloyd algorithm to find a new set of breakpoints that should better resemble the original data distribution than the assumption of normality from SAX. Since we want to use symbolization to forecast time series we will only make use of symbolization techniques that make use of one symbol: SAX and aSAX. This is made to create an experimental scenario where all techniques can be compared. This is due to the fact that, in the first place, it is not easy to decide whether they should be compared by making use of equal size segments or the same amount of symbols (if even possible) and, in the second place, many of this techniques, as they were intended for other tasks such as indexing or classification don't propose a way to transform the extra information hold on the new symbols into a numerical value (required for the forecasting task).

2.2 Artificial Neural Networks (ANN)

Artificial Neural Networks are machine learning models inspired by the human's brain neural system. ANNs are structured in multiple layers of neurons where each neuron can be connected to one or more neurons of another layer. Each neuron computes a weighted sum of the inputs and applies a usually nonlinear function chosen by the user named activation function. The learning process of a neural network consists of optimizing those weights to minimize the difference between the output layer and the desired output. In our experimentation, we compared four ANN architectures.

Multilayer perceptrons (MLP) [24] are one of the most simple and widely used feed-forward artificial neural networks. Due to lower complexity, they are easier to train and perform fast operations. Nevertheless, previous work has shown that classic feed-forward neural networks may outperform many modern architectures. Its architecture consists of at least three sequential fully connected layers: one input layer, one or more hidden layers and the output layer. In this architecture, the output of each layer y is a vector computed according to Eq. 1, where W is a matrix that contains all of the weights of the connections between the neurons from the previous layer and the current one, x is the input to the current layer and b is a vector of biases and g is the activation function. The biases b and weights W are learnable parameters that are optimized during the learning process.

$$y = g(Wx + b) \tag{1}$$

Elman's Simple Recurrent Network [25] incorporates a feedback loop on each hidden layer neuron, allowing it to manage sequences with variable lengths and to take into account the hidden output from the previous time-step $t - 1$ of the sequence in the computation of the current one. This feedback loop is portrayed by an additional layer denominated context layer. The connection from the hidden neurons to the context neurons always has a fixed weight of 1, indicating that they will hold a copy of the current hidden output h_t. However, the connection from the context neuron to the hidden neuron will have a new set of weights U that will be used to consider the effect of previous elements of the sequence in the computation of the next time-step. Mathematically, the output of a hidden layer for a time-step t can be computed as follows.

$$h_t = g(Wx_t + Uh_{t-1} + b) \tag{2}$$

Long-Short Term Memory neural networks were proposed by Hochreiter [26] and changed the simple feedback loop present in the previous architecture for a more complex one in an attempt to address the exploding gradient and

vanishing gradient problems [27]. In this architecture, two different feedback loops are present in each neuron, one for short-term memory (hidden state) h_t and one for long-term memory (cell state) c_t. Furthermore, three different gates are used to control the information flow between the inputs and outputs of the neuron. The input gate i_t is used to control the impact of the short-term memory in the creation of the new states, the forget gate f_t is used to control how much of the long-term memory is forgotten and the output gate $o_t t$ is used to create the relationship between the short-term memory and the long-term memory. The more complex architecture of the LSTM neural networks allows them to solve more complex problems at the expense of a slower training speed, as each hidden neuron will have four independent sets of weights W, recurrent weights U and biases b. Mathematically, the LSTM hidden layer works as follows ($\otimes$ represents the element-wise product for the remainder of the paper).

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$

$$c_t = f_t \cdot c_{t-1} + i_t \otimes g(W_c x_t + U_c h_{t-1} + b_c) \tag{6}$$

$$h_t = o_t \otimes g(c_t) \tag{7}$$

Lastly, GRU [28] neural networks follow a similar idea to LSTM neural networks with a lower complexity as they don't use the memory cell and make use of only two gates to control the information flow. The reset gate r_t decides how much of the past information needs to be forgotten to create the new intermediate state for the current time-step $\hat{h_t}$, acting as a short-term memory. The update gate z_t determines how much of the new state h_t should be created from the intermediate state $\hat{h_t}$ and the previous hidden state h_{t-1}. Mathematically, this is expressed as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{8}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{9}$$

$$\hat{h_t} = g(W_h x_t + U_h(r_t \otimes h_{t-1}) + b_h) \tag{10}$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \hat{h_t} \tag{11}$$

2.3 Training Artificial Neural Networks

The process of training any neural network consists of updating all the learnable parameters of the model (weights and biases) to optimize a specific loss function between the outputs of the neural network and their expected values. This process is usually done via a gradient-based optimizer, although any other optimization algorithms, such as metaheuristics, can be used. For this task, we chose to use the Adam [29] optimizer, as it is computationally efficient, has little memory requirements and has been the most widely used optimizer in neural network applications over the past years. A detailed pseudocode of the Adam optimizer can be found in Algorithm 1.

Require: $\alpha = 0.001$ ▷ Stepsize
Require: $\beta_1 = 0.9, beta_2 = 0.999$ ▷ Exponential decay for the moment estimates
Require: W_0 Initial learnable parameters
Require: $f(W)$ ▷ Output of objective function for W
Require: $\epsilon = 10^-8$ ▷ Small number to avoid division by zero.
$m_0 = 0$, $v_0 = 0$ ▷ Initialize moment vectors
$t = 0$ ▷ Time-step
while termination criteria not reached **do**
  $t = t + 1$ ▷ Increase time-step
  $g_t = \nabla_w f_t(W_{t-1})$ ▷ Get gradients at timestep
  $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_a) \otimes g_t$ ▷ First moment estimate
  $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \otimes g_t^2$ ▷ Second moment estimate
  $\hat{m}_t = m_t / (1 - \beta_1^t)$ ▷ Bias correction
  $\hat{v}_t = v_t / (1 - \beta_2^t)$ ▷ Bias correction
  $W_t = W_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t + \epsilon}$ ▷ Update parameters
end while
return W_t

Algorithm 1: Adam

3 Fuzzy Piecewise Linear Segments for Symbolization (FPLS-Sym)

FPLS-Sym is our proposal for a new symbolic time series representation based on the linguist description technique Fuzzy Piecewise Linear Segments [30]. A general overview of the steps required to obtain the representation can be found in Fig. 1 and its pseudocode can be found in Algorithm 2. The key novelty introduced in this representation is the use of a fuzzy set with a triangular membership function to withhold more information about each segment, while still maintaining most of the advantages of other symbolization techniques. This use of fuzzy logic will make it slightly slower than other symbolization techniques, such as SAX or aSAX, as they only use the mean of the segment. However, it should remain faster than the original time series, thanks to the segmentation process involved. The FPLS-Sym representation requires the user to provide the three hyperparameters specified in Table 1.

Require: α ▷ Number of symbols (Alphabet size)
Require: n ▷ Segment size
Require: b ▷ Triangular membership function overlap
Require: $Y = \{y_1, y_2, ..., y_N\}$ ▷ Original time series
$S = \{s_1, s_2, ..., s_n\}$ = Split Y in equidistant segments of size n
$X = \{x_1, x_2, ..., x_n\}$ = Mean of all observations in each segment s_i
G = Clustering(X, α) ▷ Apply Lloyd Clustering and return centers of the clusters
$fpls = \{\{\}\}$ ▷ Empty Matrix of size $|S|$ x α
for all $s_i \in S$ **do**
  m_i, c_i = LinearRegression(s_i)
  p_i = Compute Mean Error Ratio (Eq. 12)
  $segfpls = \{\{\}\}$ ▷ Create Empty Matrix of n x α
  for all $g_j \in G$ **do**
    for all $y_k \in s_i$ **do**
      $segfpls$[j][k] = Compute membership degree between observation y_k of segment s_i and symbol g_j with overlap b(Eq. 13).
    end for
  end for
  $fpls[i, :]$ = Mean of $segfpls$ across columns
end for
return $fpls$

Algorithm 2: FPLS-Sym

In the first step, FPLS-Sym uses a similar approach to the aSAX symbolization technique, splitting the time series into equidistant segments S of a fixed-sized n. Afterwards, the mean of the observations of each segment is computed and all of the means are clustered with the Lloyd algorithm. The desired number of clusters α will be provided by the user and it will also be the alphabet size of the symbolic representation (number of symbols used). Thus, there will be a direct relationship between each cluster and symbol, where each symbol represents the mean of its cluster observations. These values will also be the universe of discourse of the fuzzy set.

$$p_i = \sum_{k=1}^{n} \frac{|m_i \cdot k + c_i - y_{i,k}|}{y_k \cdot n} \tag{12}$$

Afterward, a piecewise linear approximation of each segment s_i is computed. This is done through the use of the Least Square Method for linear regression, where we preserve the slope m_i and intercept c_i. Additionally, we compute the mean error ratio p_i between the regression residuals and each original observation $y_{i,k}$ of the segment (Eq. 12). At last, a triangular membership function is applied to create a fuzzy relationship between each segment's linear approximation and each symbol of the alphabet. The membership function for the entire segment will be the mean of the membership function for all observations within that segment. Equation 13 defines the triangular membership function between the point k of the segment S_i and one of the elements of the universe of discourse g_j, where $r_{i,k} = |m_i \cdot k + c_i - g_j$ (distance between an element of fuzzy set and the linear approximation) and b is a tunable hyperparameter that controls the level of overlap between the elements of the fuzzy set and p_i.

$$\mu(s_{i,j}) = \begin{cases} 0 & \text{if } r_{i,k} \geq p_i + b \\ 1 & \text{if } r_{i,k} = 0 \\ 1 - \dfrac{r_{i,k}}{p_i + b} & \text{if } r_{i,k} \leq p_i + b \end{cases} \tag{13}$$

Thus, the FPLS-Sym time series representation requires storing a vector of membership degrees for each segment instead of only the mean value used in other symbolization techniques, such as SAX or aSAX. Thus, the neural network model may take into account three different factors while using the proposed representation. First, thanks to the use of the triangular membership function, the model can take into account how close each value is to each symbol's mean. Second, thanks to the use of the error rate of the linear regression p_i as a penalization, lower membership degrees will usually indicate a worse linear approximation. Lastly, since the vector of membership degrees of the segment is the mean of its equivalent for each of the observations, the model can take into account the trend of the segment as the membership degree for a symbol will be 0 if no part of the linear approximation is nearby. Thus, the FPLS-Sym contains a lot more information than the other classical symbolization techniques, that only use the mean of the segment and assign a symbol based on a Gaussian distribution (SAX) or a previous clustering process (aSAX). Figure 2 shows the computation of the FPLS-Sym representation for the segment displayed in the plot.

4 Experimentation

4.1 Data Description and Preprocessing

All the models compared in this work were trained and tested with energy demand data from the Spanish national electricity grid (Fig. 3), scrapped from the official website of the partly state-owned corporation that operates the grid, *REE* (Spanish Electricity Network) [31].

Their website provides information about the expected demand, the actual demand, and various other variables

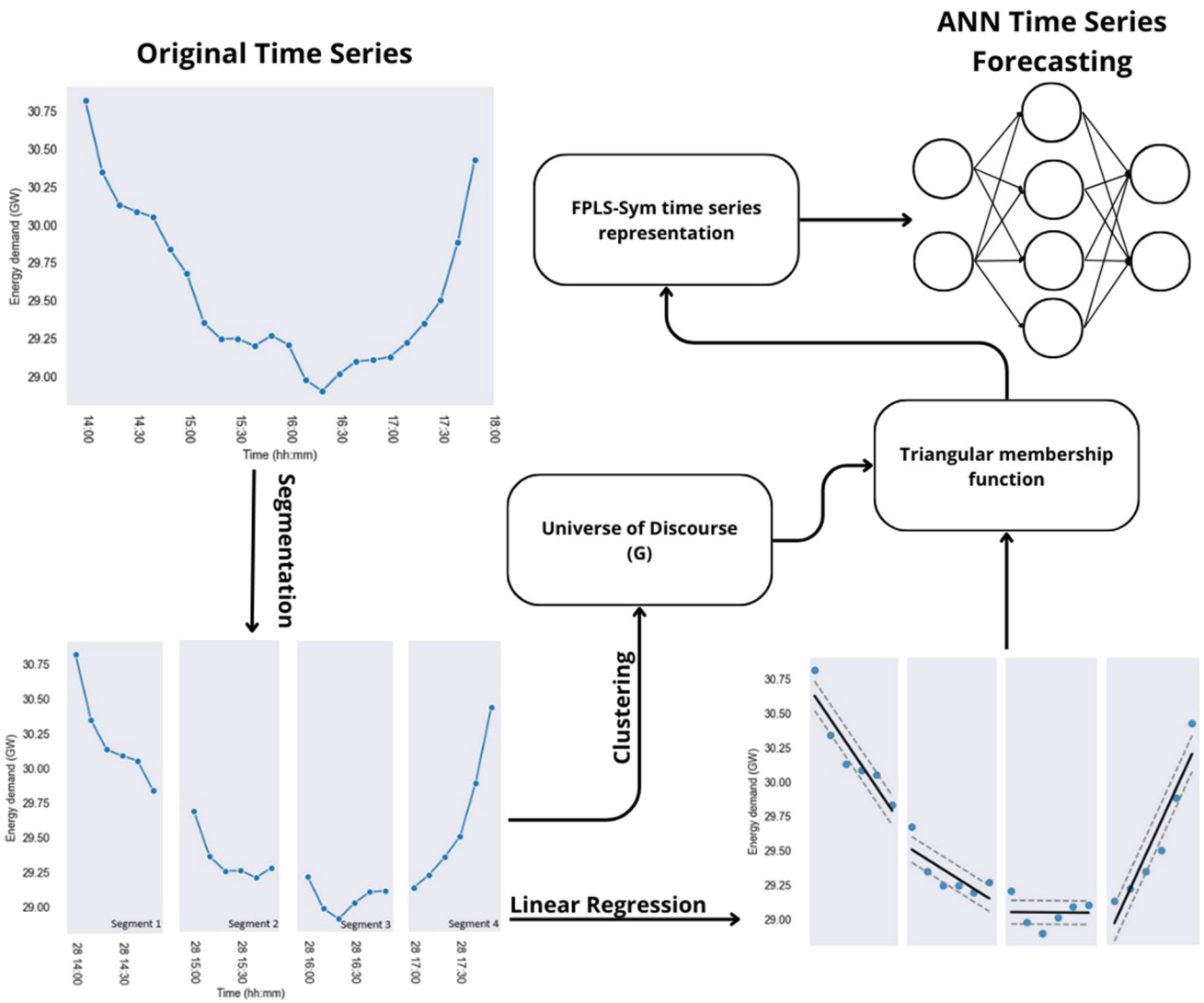


Fig. 1 A general overview of the steps required to obtain the FPLS-Sym representation

Table 1 Hyperparameters of FPLS-Sym

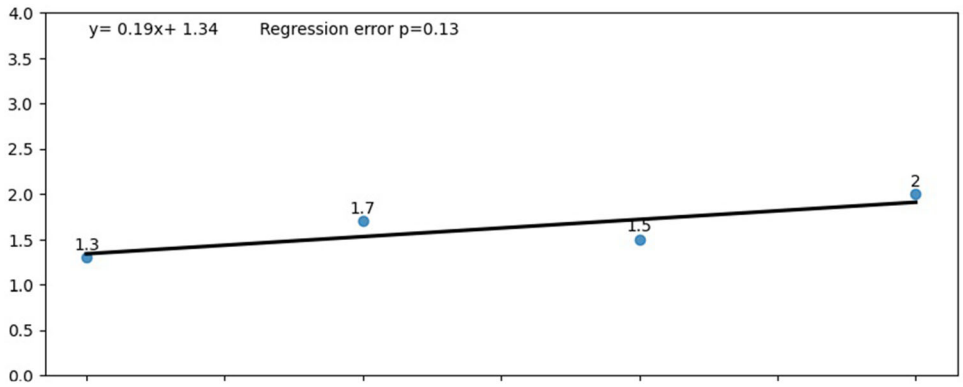
Hyperparameter	Meaning
α	Alphabet size
n	Segment size
b	Overlap of the membership function

related to gas emissions and energy production. Each observation on the dataset is made every 10 min and it provides information about energy demand from 2007 to the current date. However, emissions and energy production were not recorded until much later. For this paper, we only used the actual demand value from 2009 to 2019. The dataset did not present any missing values and the only preprocessing step was fixing the daylight saving time (DST) so every day had 24 h. This was done by adding an extra hour with the mean of the previous and the next one if the clock is advanced or by keeping the mean of the repeated hour if the clock is turned back to standard time.

The dataset was divided into three partitions preserving chronological order: 70% training data, 10% validation data, and 20% test.

4.2 Selection of Hyper-parameters for Symbolization Techniques

In order to use the symbolization techniques we are comparing (SAX, aSAX and FPLS-Sym) we need to provide an alphabet size (total amount of symbols available for the discretization process) and a segment size. Selecting any of these parameters is not trivial and there is no way to find an optimal value without doing trial and error.



	Symbol 1 (0.5)	Symbol 2 (1.5)	Symbol 3 (2.5)	Symbol 4 (3.5)
Obs 1 (1.3, 1.34)	0.0455	0.8182	0	0
Obs 2 (1.7, 1.53)	0	0.9659	0	0
Obs 3 (1.5, 1.72)	0	0.7500	0.1136	0
Obs 4 (2.0, 1.91)	0	0.5341	0.3295	0
FPLS-Sym	**0.0114**	**0.7670**	**0.1108**	**0**

Fig. 2 An example of the computation of FPLS-Sym for a segment using an overlap $b = 0.75$ and the symbol centers $G = \{0.5, 1.5, 2.5, 3.5\}$

In the case of the alphabet size, the use of larger alphabet sizes would provide more symbols, but each symbol would cover a smaller interval. As we have a high number of symbols, it is more likely to fail the symbolic forecast, but, whenever we predict the expected symbol, the difference between the expected numerical value and the numerical value provided by the symbol should be smaller in most cases. This would be further accentuated if models take into account some notion of order between symbols as it would reduce the number of times a predicted symbol represents an interval far away from the expected one. The exact opposite situation would happen for low alphabet sizes.

In the case of the segment size, the selection of a larger segment size will provide better training times, as they will provide a smaller sample size to train; however, it will provide worse numerical approximations to the original time series, as we would to need to repeat more times the numerical value used to represent the symbol.

Another important factor to take into account from using a symbolic representation is how easy they are to interpret. This approach is particularly useful when experts provide reasonable intervals for each symbol that represents something meaningful for posterior decision-making. In our experimentation, we have led more towards the interpretation approach for the selection of hyper-parameters. We selected the segment size and alphabet size, making use of the same criteria of our previous study [8]. As such, we studied the use of a segment size of 6 and alphabet sizes of 7 and 13.

4.2.1 Sample Extraction with Sliding Windows

Training a neural network with time series data requires a previous step in which we create samples with an input and its desired output. In order to create these samples, we made use of a sliding window that covers the number of observations corresponding to two consecutive days (the first for the input and the second for the output). This results in the use of a sliding window of size 144 when training models with the original time series and a sliding window of size 24 when training models with symbolic representations with the hyper-parameters we chose. Since the objective is to create models that predict energy demand always from 0:00 to 23:50 we took into account two alternative steps to move the sliding window. The use of a sliding window step of 1 creates models trained with more samples, capable of doing forecasts from any hour of the day but that requires more time to train while the use of a daily window step creates models trained with fewer samples, thus, they are trained faster but they are always limited to make forecasts from observations starting at 0:00.

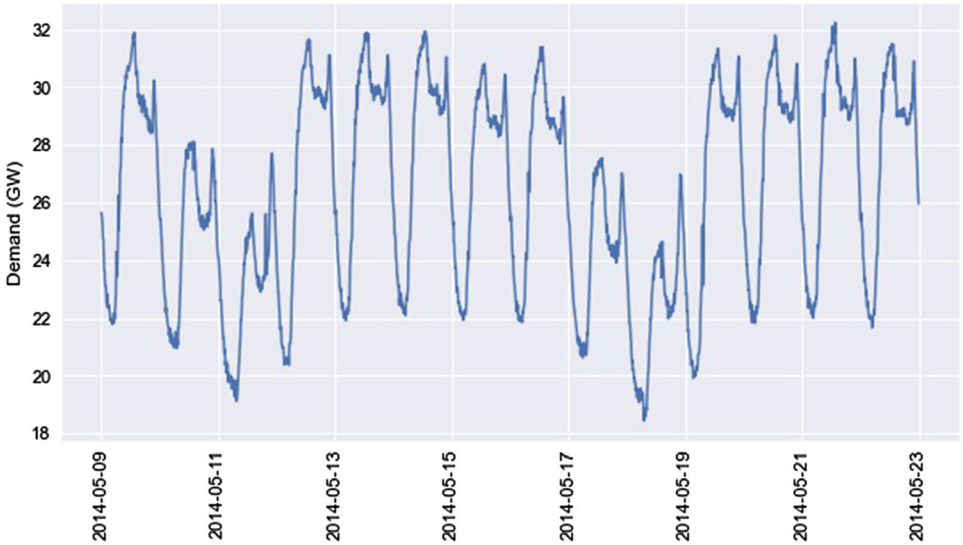


Fig. 3 Two weeks of demand data from REE

4.3 Experiment Description and Setup

All the experiments done in this work are done to compare how well neural network models perform with and without symbolization and how well they perform in comparison with other machine learning models. More specifically, we will evaluate the optimal way to integrate the proposed symbolization in the neural network, compare how accurately each symbolization technique is capable of predicting the next symbol and compare how accurately all of the models are capable of forecasting the energy consumption in its numerical representation.

In order to do so, extensive experimentation will be conducted with all neural network models using a trial-and-error approach to make the comparison as fair as possible. For reproducibility purposes, we provide all the hyperparameters evaluated in this section. Any unmentioned parameters were kept to the default value of the *TensorFlow* [32] framework, which was used for all experimentation. We tested topologies between 5 and 60 hidden neurons (increasing by 5) of the MLP, ELman, LSTM and GRU neural network architectures. Furthermore, in all of them, three different hidden activation functions were evaluated: hyperbolic tangent (tanh), sigmoid and ReLU. The random seed used to initialize the weights was 1996. After several preliminary experiments, the value of the overlap of the triangular membership function b was set to 3.5. All models were trained during up to 75 epochs with early stopping if the results did not improve for 10 epochs. We used the cross-entropy loss function for the symbolic time series and the mean squared error for the numeric time series. The learning rate (Adam's stepsize) when working with the symbolic representation was raised to 0.005 since with the default value of 0.001 it was not converging. The computer used to execute all the experiments had 32 GB of RAM and an AMD Ryzen 5 2600X running at 3.6 GHz.

Additionally, since we are working with time series data, we will also evaluate the optimal way to extract samples with a sliding window in the case of each symbolic representation and the numerical representation. More specifically, we will evaluate whether it is more interesting to use a window step of 1, in which we will create a new sample after each observation/simple or whether it is more useful to do a daily step, in which a sample is only created when the first observation/symbol happens at 0:00.

4.3.1 Integrating FPLS-Sym: Representation Encoding as Input and Output

Artificial neural networks require a numerical representation in order to make all the computations required in the architecture. While the proposed representation has a numerical representation that provides richer information than other symbolization techniques (the membership degrees) it is unclear if the use of that encoding will be optimal in the output layer. Thus, we will evaluate three different ways in which our symbolic representation can be encoded in the output layer.

In the first one, "*membership*", the neural network will try to forecast the next values for the membership function. In this case, the neural network does not learn anything about the defuzzification process, which will be done through the use of the *argmax* function. This should notably lead to more complex architectures and accurately forecasting the fuzzy membership will be a harder task than the other two alternatives.

The second alternative, "*one-hot encoding*" consists in transforming the α symbols of the alphabet into an output vector in $\{0,1\}^{\alpha}$ where the symbol g_i is represented with the vector that only has a value of 1 in position i. This would be the simplest encoding that can be used but will also remove any notion of order during the training process. Thus, the neural network will value in the same way errors that are close or far away from the expected value.

The third and last alternative, "*ordinal*", is an ordinal regression representation for neural network proposed in [33]. This encoding solves the issue of the second one, as it makes the neural network aware of the order between symbols during the training process. In this case, the representation of a symbol g_i of the alphabet is a vector in $\{0,1\}^{\alpha}-1$, where the symbol i is represented as a vector of ones until the $i-1$ position, and the remaining elements set to 0. Additionally, this encoding requires the use of the sigmoid activation function in the output layer and the use of the mean squared error loss as the objective function.

5 Results

5.1 Statistical Tests

Some of the results obtained in the experiments are supported by statistical tests performed with a significance level $\alpha = 0.05$. Particularly, for each comparison made, using a Shapiro-Wilk test, we could reject the normality assumption for at least one of the samples being compared. As such, we use a Wilcoxon signed-rank test to compare paired samples of the performance metrics. A pair is any case in which we use the exact same methodology steps (architecture, topology, activation, sliding window step, symbolization technique and encoding) except one step that defines the groups. There are multiple ways to formulate the hypothesis of the Wilcoxon signed-rank test. In our experimentation, we will consistently use the same hypothesis formulation. The null hypothesis will be that the pseudomedian of the differences between samples is negative. A *p-value* inferior to the significance level would lead us to reject the hypothesis, in which case, we could claim with confidence of $1-\alpha$ that the metric in the first sample usually has a greater value than the second sample. Table 2 shows the results of all Wilcoxon signed-rank tests conducted. These results will be discussed later in their corresponding sections.

5.2 Forecasting Performance Metrics

Since we want to evaluate our models under two different situations (accurately predicting the next symbol or approximating the original time series) we have two sets of performance metrics.

In order to evaluate symbolization metrics, we considered the rooted mean squared error (RMSE, Eq. 14) and the accuracy. In order to calculate the symbolic RMSE (the RMSE while using a symbolic representation), each symbol is replaced with an integer that represents its position on the alphabet. We will refer to the symbolic RMSE as RMSE-Sym for the remainder of this paper. The best topology was always selected based on the lowest RMSE-Sym as it will also penalize wrong symbols that represent intervals far away from the expected value. The accuracy is the percentage of predicted symbols that correspond with their expected symbol.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y_i} - y_i)^2}{N}} \tag{14}$$

where $\hat{y_i}$ is the predicted value, y_i is the expected value and N is the sample size.

In the case of predictions with a numerical representation, we used the RMSE metric (defined previously in Eq. 14) and the mean absolute percentage error (MAPE, Eq. 15)

$$MAPE = \frac{1}{N}\sum_{i=1}^{n}\left|\frac{y-\hat{y}}{y}\right| \tag{15}$$

5.3 Impact of the Encoding Used in the Symbolization Techniques

One of the most important aspects of the proposed experimentation, was to identify what was the optimal way to integrate the symbolic representations in the neural networks, as they require numerical input and output. Thus, we evaluated the use of two types of encoding for all symbolization techniques as well as the use of the membership degrees for FPLS-Sym.

The results obtained, displayed in the statistical tests (Table 2—rows 1 and 2) prove that the use of ordinal encoding in all symbolization techniques (SAX, aSAX and FPLS-Sym) improves both metrics (provides a lower RMSE-Sym and a higher accuracy). Additionally, the use of the membership representation in the output layer provided some interesting results for FPLS-Sym. When it was used with a feed-forward neural network the best results were provided, however, when working with recurrent neural networks the use of the membership function was usually more beneficial. However, the overall best models found (as can be seen in Table 3) still made use of the ordinal encoding.

5.4 Impact of the Sliding Window's Sample Generation

Another relevant factor before the application of the neural network architecture is how to prepare and feed the samples used to train the neural network given the nature of the time series data. Particularly, we evaluated two different ways of extracting the samples: one in which we only feed samples starting at the first hour of the day (daily) and one in which we feed as many samples as possible creating a new sample in every time-step (1-step). Table 4 shows the best models found while forecasting the time series in its symbolic form.

As expected, the use of the daily-step window significantly reduces the required training time since it provides less sample for training. However, it did provide significantly better performance metrics too (Table 2—rows 3 and 4). This behaviour is most likely caused by the creation of too many incoherent samples due to the daily seasonality of energy data. We define as incoherent samples any two or more training samples that force the model to always fail the forecast of at least one of them since they share the same input but require different outputs. Since energy consumption is highly correlated with human and industrial activity, we found the same symbolic string starting at different hours and it will usually require different outputs. Therefore, due to the nature of our data, objective and methodology, using a daily-step window should always be preferred although it restricts our model to make forecasts with inputs that always have to start at midnight, which is the most common use case of day-ahead forecasting models.

5.5 Symbolization Techniques Comparison

The last alternative in the proposed methodology is the selection of the symbolization technique. Particularly, we want to check whether any symbolization technique provides better quality metrics than the others when the neural network is used to forecast the next 24 symbols. A comparison of all of them using alphabets with 7 and 13 symbols is provided in Table 4. The result show that FPLS-Sym outperformed both SAX and aSAX independently of the other hyperparameters evaluated in our methodology. This is the expected behaviour as FPLS-Sym is capable of providing more accurate information to the input layer of the neural network at the expense of more space to be stored and some additional computational power, that explains the slightly slower training time required when using this symbolization technique. The best models for each of the alphabet sizes used the previously discussed optimal training methodology for our data: a MLP architecture with ordinal encoding and a daily-step sliding window to provide the training samples.The only difference between them is the number of neurons on their hidden layer and the activation function used. Another important factor to highlight is that even FPLS-Sym models that don't use ordinal encoding (Table 3) or use a daily-step sliding window with recurrent neural networks provide better results than the best models found for the other symbolization techniques. Therefore, also taking into account the results of the statistical tests conducted (Table 2—last 5 rows) we can conclude that the use of FPLS-Sym will provide a significant improvement on RMSE-Sym and accuracy over the other symbolization techniques used at the expense of a small increase in calculations.

5.6 Forecasting the Time Series in Its Numerical Form

At last, we will compare the performance of symbolization techniques with the same neural network architectures trained with the numerical representation as well as other machine learning models. Symbolic forecasts are transformed into numerical forecasts by replacing each symbol with its center value and repeating that value as many times as long is its corresponding segment. Figure 4 displays how the best model for aSAX, FPLS-Sym and the numerical representation prediction in one week of the test dataset. Table 5 compares the performance of the numerical forecast between the best models found for each symbolization technique, the best models obtained with the numerical representation and other regression algorithms.

Among all the methods evaluated, FPLS-Sym provided the best forecast, improving the results of all symbolization techniques and the other algorithms that used a numerical representation. The best model with FPL-Sym provided a RMSE of 1.1655 and a MAPE of 3.29%, obtaining a reasonable improvement over the second-best performant model and being capable of training in just under 7 s. The second-best performant model was the best neural-network-based model that provided better performance metrics than the other symbolization techniques at the expense of a much higher training time. This high training time is easily explainable due to the higher dimensionality of the numerical representation, the use of more training samples through a one-step sliding window and the complexity of the recurrent LSTM units. The third-best performant model was the neural network trained with aSAX with a daily window and 13 symbols, providing quality metrics slightly worse than the numerical LSTM but also training much faster. Lastly, most neural-network-based models provided better quality metrics than the other machine learning algorithms evaluated in Table 5.

Table 2 Wilcoxon signed-rank test

Metric	X_1	X_2	T	p value
RMSE-Sym	One-hot	Ordinal	961660.0	1.41×10^{-173}
Accuracy	Ordinal	One-hot	971110.5	7.48×10^{-183}
RMSE-Sym	1-step window	Daily window	969941.0	4.85×10^{-180}
Accuracy	Daily window	1-step window	972253.0	7.24×10^{-182}
RMSE-Sym (MLP+FPLS)	Membership	Ordinal	350.0	0.0078
Accuracy (MLP+FPLS)	Ordinal	Membership	450.	0.0002
RMSE-Sym (RNN+FPLS)	Ordinal	Membership	3061.0	2.27×10^{-5}
Accuracy (RNN+FPLS)	Membership	Ordinal	2927.0	0.0002
RMSE-Sym	SAX	aSAX	222294.0	5.65×10^{-62}
Accuracy	SAX	aSAX	231280.5	3.68×10^{-74}
RMSE-Sym	aSAX	FPLS-Sym	434728.0	5.50×10^{-125}
Accuracy	FPLS-Sym	SAX	244789.0	0.0498
Training time (s)	FPLS-Sym	SAX	292600.0	2.79×10^{-13}

$H_0 : X_1 - X_2$ are symmetric about $\mu < 0$

Table 3 Comparative of FPLS-Sym neural network training defuzzification strategies making use of daily-step sliding window and ordinal encoding

Alphabet size	ANN output	Architecture	Activation	Neurons	RMSE (Sym)	Accuracy
7	Membership	MLP	sigmoid	45	0.5337	0.7113
7	Membership	Elman	tanh	25	0.6000	0.6634
7	Membership	LSTM	tanh	60	0.5430	0.7103
7	Membership	GRU	ReLU	55	0.5382	0.7151
7	Ordinal	MLP	tanh	50	**0.5047**	**0.7489**
7	Ordinal	Elman	ReLU	50	0.5924	0.6701
7	Ordinal	LSTM	ReLU	35	0.5594	0.7067
7	Ordinal	GRU	ReLU	25	0.5768	0.6862
13	Membership	MLP	sigmoid	55	0.8585	0.5295
13	Membership	Elman	tanh	55	0.9595	0.4900
13	Membership	LSTM	tanh	50	0.8766	0.5111
13	Membership	GRU	tanh	40	0.8773	0.5233
13	Ordinal	MLP	tanh	55	**0.8077**	**0.5620**
13	Ordinal	Elman	ReLU	25	0.9654	0.4631
13	Ordinal	LSTM	ReLU	45	0.9320	0.5117
13	Ordinal	GRU	ReLU	45	0.9123	0.4990

Best metrics per alphabet size in bold

5.7 Uses Cases and Limitations of the Proposed Approach

As can be observed in the results displayed in this section, the use of the FPLS-Sym technique provides more accurate results in a faster time than the models trained with the numerical representation. Furthermore, even though it is a more complex symbolization technique, the time required to train the neural network still remains competitive, especially when using the MLP architecture. This partially thank to the fact that after taking into account the encodings used, the FPLS-Sym representation only requires an additional neuron in the input layer in comparison with the ordinal input from SAX and aSAX. In general, there are three main use cases of the proposed symbolization technique. The first one would be to train quickly an initial model that may be used as a baseline to compare other models. A second use would be for better interpretation of the results. In that instance, the forecast is provided in its symbolic form, helping experts in the field make decisions and learn frequent patterns in the studied time series. A third extremely useful use case of the proposed approach would be instances in which a model needs to be retrained frequently. For example, in the energy sector, major

Table 4 Best topologies for all models trained with ordinal encoding

Representation (alphabet size)	Window step	Architecture	Neurons	Activation	Symbolic RMSE	Accuracy	Training time (s)
SAX (7 symbols)	Daily	MLP [Ordinal]	60	sigmoid	0.6366	0.6888	4.9495
aSAX (7 symbols)	Daily	MLP [Ordinal]	45	ReLU	0.6181	0.6664	3.4701
FPLS-Sym (7 symbols)	Daily	MLP [Ordinal]	50	tanh	**0.5047**	**0.7436**	6.6461
SAX (7 symbols)	1	GRU [Ordinal]	55	sigmoid	0.7269	0.6470	459.7355
aSAX (7 symbols)	1	LSTM [Ordinal]	45	ReLU	0.6578	0.6371	590.4094
FPLS-Sym (7 symbols)	1	LSTM [Ordinal]	45	tanh	0.5654	0.6940	806.4591
SAX (13 symbols)	Daily	MLP [Ordinal]	60	ReLU	0.9893	0.5584	5.0936
aSAX (13 symbols)	Daily	MLP [Ordinal]	25	ReLU	0.9548	0.5250	6.8402
FPLS-Sym (13 symbols)	Daily	MLP [Ordinal]	45	ReLU	**0.8077**	**0.5585**	6.9993
SAX (13 symbols)	1	GRU [Ordinal]	60	tanh	1.1634	0.4823	419.7400
aSAX (13 symbols)	1	GRU [Ordinal]	25	sigmoid	1.0421	0.4919	1054.4451
FPLS-Sym (13 symbols)	1	LSTM [Ordinal]	55	ReLU	0.8701	0.5298	1406.6871

Best metric per alphabet size in bold

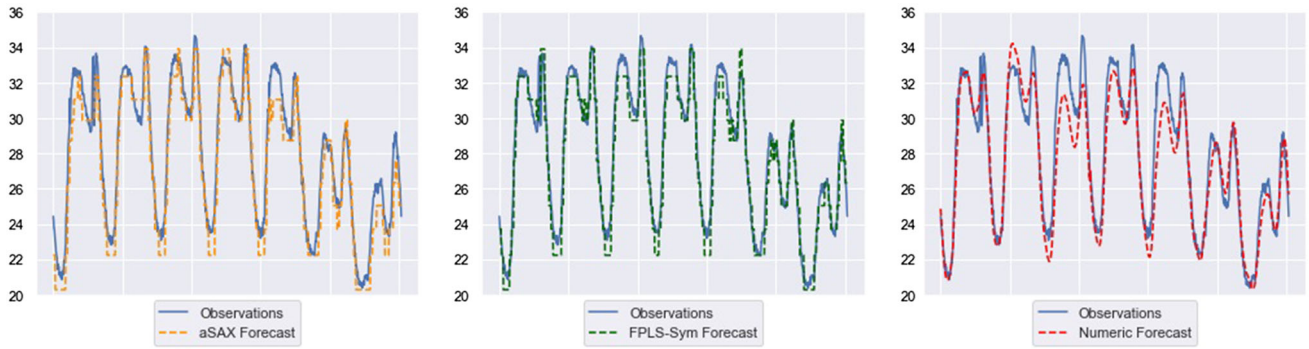


Fig. 4 Predictions of the best model for aSAX (on the left), FPLS-Sym (on the middle) and the numeric representation (on the right) over the span of a week of the test partition

changes to energy policies or the infrastructure used will most likely lead to a major change in the time series behavior. In those cases, using this type of model can be extremely useful, as it can be retrained much faster than other complex models that will require a much larger training time and many more observations until it can learn properly the new behavior. Additionally, thanks to the lower complexity of models trained with FPLS-Sym, they may also be deployed in edge devices at the consumer household, reducing the cost of frequent and large communication between sensors and data centers and helping to preserve the privacy of the consumer (federated learning).

Nevertheless, the main limitation is the kind of data used to train the models. Since the symbolization process will inevitably compact the information of the numerical representation, the use of the proposed technique in time series with low granularity or trivial problems will most likely yield underwhelming results.

6 Conclusion

In this paper, we applied symbolization techniques to forecast the energy demand in Spain in a fast and precise manner and presented a new algorithm, FPLS-Sym, that outperformed the other techniques for the forecasting task. The proposed symbolization technique was evaluated with Spanish energy demand data from 2009 to 2019 with observations gathered every 10 min. Extensive experimentation was done on this dataset comparing different input encodings, window size, neural network architectures and topologies, symbolization techniques and other forecasting algorithms with three main objectives in mind. First, we wanted to evaluate how valuable would be to integrate a fuzzy representation in symbolization techniques. Second, we wanted to apply the proper statistical tests to verify the optimal way to incorporate the symbolic representations in a neural network model. Third, we wanted to do everything in a publicly available big data

Table 5 Original/numerical time series forecast and training time for the best numeric and symbolic models

Representation (alphabet size)	Window step	Architecture	Neurons	Activation	RMSE	MAPE	Training time (s)
SAX (7 symbols)	Daily	MLP [Ordinal]	60	sigmoid	1.6291	0.0475	4.9495
SAX (13 symbols)	Daily	MLP [Ordinal]	60	ReLU	1.4307	0.0408	5.0936
aSAX (7 symbols)	Daily	MLP [Ordinal]	45	ReLU	1.6618	0.0484	3.4701
aSAX (13 symbols)	Daily	MLP [Ordinal]	25	ReLU	1.3655	0.0390	6.8402
FPLS-Sym (7 symbols)	Daily	MLP [Ordinal]	50	tanh	1.8401	0.0502	6.6461
FPLS-Sym (13 symbols)	Daily	MLP [Ordinal]	55	tanh	**1.1655**	**0.0329**	6.9993
SAX (7 symbols)	1	LSTM [Ordinal]	60	ReLU	1.8391	0.0532	667.5387
SAX (13 symbols)	1	LSTM [Ordinal]	60	ReLU	1.5903	0.0454	584.8650
aSAX (7 symbols)	1	LSTM [Ordinal]	45	ReLU	1.8402	0.0531	441.6372
aSAX (13 symbols)	1	GRU [Ordinal]	25	sigmoid	1.5306	0.0439	1054.4451
FPLS-Sym (7 symbols)	1	LSTM [Ordinal]	45	tanh	2.9634	0.0829	575.4878
FPLS-Sym (13 symbols)	1	LSTM [Ordinal]	55	ReLU	1.3884	0.0391	1406.6871
Numeric	Daily	MLP	60	ReLU	1.5542	0.0434	8.5964
Numeric	1	LSTM	55	tanh	1.2889	0.0363	40,959.7513

Representation	Window step	Prediction model	Optimal parameters	RMSE	MAPE	Training time (s)
Numeric	1	Decision Tree	max_depth: 15	2.6410	0.0733	87.4397
Numeric	1	Random Forest	max_depth:20 n_estimators: 150	1.7465	0.0492	15,484.5837
Numeric	1	Gradient Boosting Trees	max_depth: 20 n_estimators: 150 learning_rate: 0.1	1.4900	0.0422	22,284.1009

Parameters evaluated: $max\_depth \in [10, 15, 20, 25, 30]$; n_estimators $\in [50, 100, 150, 200]$; learning_rate $\in [0.05, 0.1, 0.15, 0.2, 0.3]$

Any other parameter not mentioned corresponds to scikit-learn default values. Multi-step forecast is done recursively

dataset, verifying the usefulness of the approach with large amounts of data and allowing easy reproduction of our findings in future research. The results from the experimentation done in this paper showed that the use of the proposed fuzzy technique clearly outperformed the other classic symbolization techniques, pointing out how useful it is the use of the fuzzy representation to improve the accuracy of the model. Secondly, thanks to the multiple Wilcoxon signed-rank test applied, we saw that FPLS-Sym consistently outperform the other alternatives and it should ideally be used with an MLP architecture, ordinal encoding and a daily sliding window. Lastly, the results showed that our best FPLS-Sym model did not only outperform the other symbolization techniques but was also capable of providing better metrics to forecast the original time series representation and required much less training time than the models trained with the numerical representation.

Future lines of work may study the inclusion of exogenous variables, different machine learning models, other fuzzy representations, i.e. using other membership functions; or accelerating the selection of all the parameters evaluated using the GPU.

Author Contributions All authors contributed equally to this work.

Funding The authors acknowledge financial support from "Ministerio de Ciencia e Innovación" (Spain) (Grant PID2020-112495RB-C21 funded by MCIN/ AEI /10.13039/501100011033) and from "Consejería de Universidad, Investigación e Innovación de la Junta de Andalucía" (I+D+i FEDER 2020 project B-TIC-42-UGR20).

Data Availability The datasets generated during and/or analysed during the current study are available in an OpenScienceFramework repository, https://osf.io/v2zfm.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Min. Knowl. Disc. **15**, 107–144 (2007). https://doi.org/10.1007/s10618-007-0064-z
2. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowl. Inf. Syst. (2002). https://doi.org/10.1007/PL00011669
3. Lkhagva, B., Suzuki, Y., Kawagoe, K.: New time series data representation ESAX for financial applications. In: 22nd International Conference on Data Engineering Workshops (ICDEW'06), pp. 115–115 (2006). https://doi.org/10.1109/ICDEW.2006.99
4. Pham, N.D., Le, Q.L., Dang, T.K.: Two novel adaptive symbolic representations for similarity search in time series databases. In: 2010 12th International Asia-Pacific Web Conference, pp. 181–187 (2010). https://doi.org/10.1109/APWeb.2010.23
5. Reinhardt, A., Koessler, S.: PowerSAX: Fast motif matching in distributed power meter data using symbolic representations. In: 39th Annual IEEE Conference on Local Computer Networks Workshops, pp. 531–538 (2014). https://doi.org/10.1109/LCNW.2014.6927699
6. Chen, Y., Wen, J.: Whole building system fault detection based on weather pattern matching and PCA method. In: 2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE), pp. 728–732 (2017). https://doi.org/10.1109/CCSSE.2017.8088030
7. Miller, C., Nagy, Z., Schlueter, A.: Automated daily pattern filtering of measured building performance data. Automat. Cosntr. **49**, 1–17 (2015). https://doi.org/10.1016/j.autcon.2014.09.004
8. Criado-Ramón, D., Ruiz, L.G.B., Pegalajar, M.C.: Electric demand forecasting with neural networks and symbolic time series representations. Appl. Soft Comput. **122**, 108871 (2022). https://doi.org/10.1016/j.asoc.2022.108871
9. Ediger, V.Ş, Akar, S.: ARIMA forecasting of primary energy demand by fuel in Turkey. Energ. Policy **35**(3), 1701–1708 (2007). https://doi.org/10.1016/j.enpol.2006.05.009
10. Li, S., Li, R.: Comparison of forecasting energy consumption in Shandong, China Using the ARIMA model, GM model, and ARIMA-GM model. Sustainability **9**(7), 1181 (2017). https://doi.org/10.3390/su9071181
11. Wang, H., Lei, Z., Zhang, X., Zhou, B., Peng, J.: A review of deep learning for renewable energy forecasting. Energ. Convers. Manage. **198**, 111799 (2019). https://doi.org/10.1016/j.enconman.2019.111799
12. Bagnasco, A., Fresi, F., Saviozzi, M., Silvestro, F., Vinci, A.: Electrical consumption forecasting in hospital facilities: an application case. Energ. Build. **103**, 261–270 (2015). https://doi.org/10.1016/j.enbuild.2015.05.056
13. Naji, S., Keivani, A., Shamshirband, S., Alengaram, U.J., Jumaat, M.Z., Mansor, Z., Lee, M.: Estimating building energy consumption using extreme learning machine method. Energy **97**, 506–516 (2016). https://doi.org/10.1016/j.energy.2015.11.037
14. Ribeiro, G.T., Mariani, V.C., Santos Coelho, L.: Enhanced ensemble structures using wavelet neural networks applied to short-term load forecasting. Eng. Appl. Artif. Intel. **82**, 272–281 (2019). https://doi.org/10.1016/j.engappai.2019.03.012
15. Sajjad, M., Khan, Z.A., Ullah, A., Hussain, T., Ullah, W., Lee, M.Y., Baik, S.W.: A novel CNN-GRU-based hybrid approach for short-term residential load forecasting. IEEE Access **8**, 143759–143768 (2020). https://doi.org/10.1109/ACCESS.2020.3009537
16. Zhang, G., Bai, X., Wang, Y.: Short-time multi-energy load forecasting method based on CNN-Seq2Seq model with attention mechanism. Mach. Learn. Appl. **5**, 100064 (2021). https://doi.org/10.1016/j.mlwa.2021.100064
17. Martinez Alvarez, F., Troncoso, A., Riquelme, J.C., Aguilar Ruiz, J.S.: Energy time series forecasting based on pattern sequence similarity. IEEE Trans. Knowl. Data Eng. **23**(8), 1230–1243 (2011). https://doi.org/10.1109/TKDE.2010.227
18. Nepal, B., Yamaha, M., Yokoe, A., Yamaji, T.: Electricity load forecasting using clustering and arima model for energy management in buildings. Jpn. Archit. Rev. **3**(1), 62–76 (2020). https://doi.org/10.1002/2475-8876.12135
19. Pérez-Chacón, R., Asencio-Cortés, G., Martínez-Álvarez, F., Troncoso, A.: Big data time series forecasting based on pattern sequence similarity and its application to the electricity demand. Inf. Sci. **540**, 160–174 (2020). https://doi.org/10.1016/j.ins.2020.06.014
20. Jin, N., Yang, F., Mo, Y., Zeng, Y., Zhou, X., Yan, K., Ma, X.: Highly accurate energy consumption forecasting model based on parallel LSTM neural networks. Adv. Eng. Inf. **51**, 101442 (2022). https://doi.org/10.1016/j.aei.2021.101442
21. Du, J., Zheng, J., Liang, Y., Liao, Q., Wang, B., Sun, X., Zhang, H., Azaza, M., Yan, J.: A theory-guided deep-learning method for predicting power generation of multi-region photovoltaic plants. Eng. Appl. Artif. Intell. **118**, 105647 (2023). https://doi.org/10.1016/j.engappai.2022.105647
22. Zhang, K., Li, Y., Chai, Y., Huang, L.: Trend-based symbolic aggregate approximation for time series representation. In: 2018 Chinese Control And Decision Conference (CCDC), pp. 2234–2240 (2018). https://doi.org/10.1109/CCDC.2018.8407498
23. Yu, Y., Zhu, Y., Wan, D., Liu, H., Zhao, Q.: A novel symbolic aggregate approximation for time series. In: Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM) 2019, pp. 805–822 (2019). https://doi.org/10.1007/978-3-030-19063-7_65
24. Almeida, L.B.: Multilayer Perceptrons. IOP Publishing Ltd and Oxford University Press, Bristol (1997)
25. Elman, J.: Finding structure in time. Cogn. Sci. **14**, 179–211 (1990). https://doi.org/10.1016/0364-0213(90)90002-E
26. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
27. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzz. **06**(02), 107–116 (1998). https://doi.org/10.1142/S0218488598000094
28. Cho, K., Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014). https://doi.org/10.3115/v1/D14-1179
29. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
30. Moreno-Garcia, A., Moreno-Garcia, J., Jimenez-Linares, L., Rodriguez-Benitez, L.: Time series represented by means of

fuzzy piecewise lineal segments. J. Comput. Appl. Math. **318**, 156–167 (2017). https://doi.org/10.1016/j.cam.2016.11.003
31. Red Eléctrica de España: Spanish peninsula electric network demand. https://demanda.ree.es/visiona/peninsula/demanda/total . Accessed 21 June 2021
32. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: Tensorflow (version 2.0.4). Zenodo (2021). https://doi.org/10.5281/zenodo.4725924
33. Cheng, J., Pollastri, G.: A neural network approach to ordinal regression. In: IEEE International Joint Conference on Neural Networks 2008 (IJCNN 2008) The IEEE World Congress on Computational Intelligence, pp. 1279–1284 (2008). https://doi.org/10.1109/IJCNN.2008.4633963

David Criado Ramón received his B.S. degree in Computer Science, specializing in Computing and Artificial Intelligence from the University of Granada, in 2019. He is currently pursuing the Ph.D. degree in Information and Communication Technologies at the University of Granada. His current research interests include neural networks, parallel computing and evolutionary algorithms.

Luis Gonzaga Baca Ruiz is an Associate Professor at the University of Granada in the Department of Software Engineering. He received his B.S. degree in Computer Science, specializing in Computing and Artificial Intelligence from the University of Granada in 2014. He obtained his Ph.D. in Information and Communication Technologies at the University of Granada in 2019. His current research interest includes time series, artificial neural networks, data mining and metaheuristic algorithms.

M. C. Pegalajar is a professor at the University of Granada at the ETSI Computer Science and Telecommunications, where she has been a professor and researcher since 1995. She obtained her bachelor's degree in Computer Science in 1993. In 1994 she was awarded a grant in the Computing Services of the University of Granada. In 1995 was hired as an associate professor in her current department, Computer Science and Artificial Intelligence, and she finished her Ph.D. project in 1997. Her current research interest includes Edge and cloud computing, big data time series, artificial neural networks and metaheuristic algorithms, psychology and machine learning.