The published version of this accepted manuscript can be found at
https://doi.org/10.1016/j.asoc.2022.108871

Electric demand forecasting with neural networks and symbolic time series representations

D. Criado-Ramón[a,*], L.G.B. Ruiz[b], M.C. Pegalajar[a]

[a]*Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain*
[b]*Department of Software Engineering, University of Granada, Granada, Spain*

Abstract

This paper addresses the electric demand prediction problem using neural networks and symbolization techniques. Symbolization techniques provide a time series symbolic representation of a lower length than the original time series. In our methodology, we incorporate the use of encoding from ordinal regression, preserving the notation of order between the symbols and make extensive experimentation with different neural network architectures and symbolization techniques. In our experimentation, we used the total electric demand data in the Spanish peninsula electric network, taken from 2009 to 2019 with a granularity of 10 minutes. The best model found making use of the symbolization methodology offered us slightly worse quality metrics (1.3655 RMSE and 0.0390 MAPE instead of the 1.2889 RMSE and 0.0363 MAPE from the best numerical model) but it was trained 6826 times faster.

Keywords: time series, forecasting, symbolic representation, energy demand, artificial neural networks

1. Introduction.

Energy has become one of the most important resources of our time. It is present in most aspects of our time and, due to its relevance, has a big environmental impact and heavily affects our economy. As such, finding ways

*Corresponding author at: c/Periodista Daniel Saucedo Aranda s.n, 18071, Granada, Spain.

Email addresses: davidcr96@correo.ugr.es (D. Criado-Ramón), bacaruiz@ugr.es (L.G.B. Ruiz), mcarmen@decsai.ugr.es (M.C. Pegalajar)

to produce and distribute energy in a sustainable and efficient way has been one of the main objectives of many governments, institutions and private parties over the last decade. Advances in storage and sensor technology have conducted a wide availability of energy data from different sources that result in really large time series. This huge amount of information may sometimes be useful but also presents some disadvantages, mainly the computational power required to process them. Thus, it is frequent to add a preprocessing stage to reduce the length of the time series.

A common approach to reduce the computational complexity when working with time series is the use of techniques that reduce the number of variables used (dimensionality reduction) or reduce the length of the time series (numerosity reduction). Syan et all [1] evaluated different feature extraction methods for dimensionality reduction (PCA, ICA, tSNE and UMAP, among others) to make a short-term forecast of the London Households dataset. Elsworth and Güttel [2] used a symbolization technique named ABBA to reduce the length of the time series and evaluated its performance when used in conjunction with LSTM neural networks. Symbolization techniques transform the original time series to a lower length sequence of discrete symbols from a finite alphabet, trying to preserve the most relevant information. In our study, we evaluate two symbolization techniques (SAX [3] and aSAX [4]) to create a short-term forecasting model for the Spanish electric demand. Using this approach, we evaluate whether the symbolization offers us faster training, better forecasts and the differences between various ways of training different neural network architectures (MLP [5], Elman [6] and LSTM [7]) with different hidden activation functions.

Neural networks are a popular approach to forecast energy demand and production as they usually offer better forecasts but are usually harder to train. Siridhipakul and Vateekul [8] used a Dual-Stage Attentional LSTM to forecast Thailand's power consumption. Their model outperformed every other traditional model for that task. Azadeh, Ghadrei, and Nokhandan [9] used seasonal artificial neural networks to do short-term load forecasting of the energy consumption in Iran one day ahead. Ehsan Simon and Venkateswaran [10] used a multilayer perceptron architecture to predict the energy output of a solar photovoltaic power plant one day ahead.

Symbolization techniques have been previously used in the energy sector.

However, they are not commonly used for forecasting task but for pattern extraction/recognition related tasks. Reinhardt and Koessler [11] created a SAX-based method to extract consumption patterns from distributed power systems. Chen and Wen [12] used SAX to find similar weather patterns in a database and use it in a PCA model to detect HVAC system faults in buildings. Miller, Nagy and Schlueter [13] used SAX to detect infrequent daily consumption patterns that could represent faults in energy systems in buildings.

However, up to date, there is only one unpublished work [2] on the use of symbolization techniques for forecasting tasks in which they use LSTM neural networks to forecast different datasets with the symbolic time series being provided to the neural network with one-hot encoding. The goal of our research is to find how suitable symbolization techniques to forecast a massive amount of energy data and expand upon the preprocessing ideas used in the previously mentioned paper by making a comparative analysis of how different approaches in the model training pipeline (sample selection, data encoding, symbolization technique and neural network architecture) can improve the obtained results.

The rest of the paper is structured as follows: the data, algorithms and methodology used are described in section 2; the results obtained are shown and discussed in section 3; and section 4 compiles the conclusions obtained from this research.

2. Materials and Methods.

2.1. Data Analysis and Preparation.

For this paper, the historical record of Spanish energy consumption was scrapped from the official REE website [14], which provides data from 2007 to present of the amount of electricity demanded every 10 minutes in the Spanish peninsula electric network. A first exploratory visual analysis provides us with information about the seasonality of the data.

As we can observe in the figure 1, the data presents seasonal patterns in three levels: daily, weekly and annually. This is due to the fact that we can observe high autocorrelation values for each 144-time step (24 hours) on the

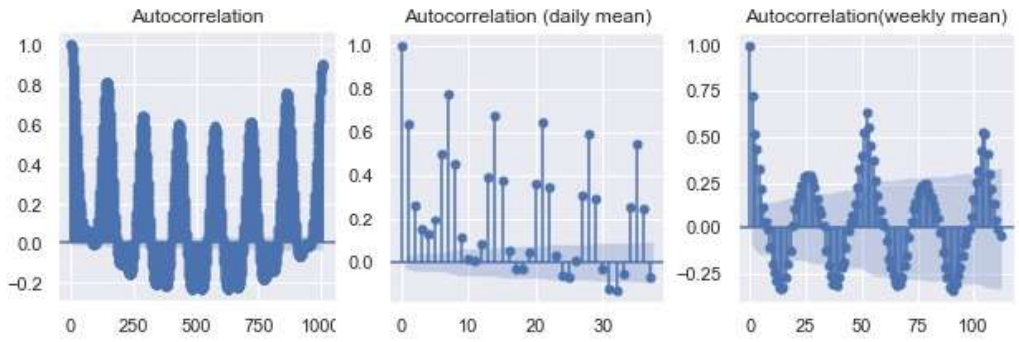

Figure 1: Autocorrelation function plots (ACF). On the left, ACF every 10 minutes. In the middle, daily mean demand ACF. On the right, weekly mean demand ACF.

left one, each 7-time step on the middle one (1 week) and every 52 weeks on the right one (1 year). Furthermore, by evaluating figure 2 we observe two relevant factors about the daily and weekly seasonality. Most days, peak demand is reached between 12 and 13 hours or between 20 and 21 hours. Sunday is the day of the week with the least electric demand followed by Saturday while the rest of the days have a quite similar demand, hinting that workdays may be an important factor in electric demand.

Figure 2: Box plots. On the left, the hourly demand box plot. On the right, box plot of the demand each day of the week.

The data for the experimentation was gathered from January 1st 2009 to December 31th 2019. A first preprocessing stage was made to fix any missing values and unwanted data. Issues with missing hours and repeated hours from daylight saving time (DST) were solved 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 data.

2.2. Artificial Neural Networks (ANN).

Artificial neural networks (ANN) are machine learning models inspired by the human nervous system. An ANN consists of many computational nodes, named neurons, and weighted connections between them. Usually, these neurons are aggregated into layers where the first layer (or input layer) provides the input data for training or forecasting, and the last layer (or output layer) provides the corresponding output. During the training process, the ANN optimizes its weights to minimize a loss function between the output from the last layer and the desired output. In our experimentation, we tried out three architectures of ANNs frequently used for time series forecasting, each with its own advantages and disadvantages.

Multilayer perceptrons (MLP) 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. The architecture of the MLP consists of at least three sequential layers: one input layer, one or more hidden layers and the output layer. Every neuron in a MLP model is fully connected, which is, each neuron is connected to all neurons from the previous and next layer. Associated with those connections there is a weight that the neural network will learn during the training process. Each neuron j (except those on the input layer) performs the sum of the inputs from the previous layer multiplied by their respective weights w_{ji} and applies a non-linear activation function f to the output.

$$h_j = f\Big(\sum_i w_{ji} x_i\Big) \tag{1}$$

A recurrent neural network (RNN) is a type of ANN in which the connections between nodes from a graph along a temporal sequence, allowing them to use their internal state (memory) to process sequences of variable length. The Elman Recurrent Neural Network [6] is a RNN that adds a new type of layer: the context layer. The context layer contains as many neurons as the hidden layer and serves as a memory by storing the output of the hidden

layer from the previous time point h_{t-1} and sending it back to all neurons on the hidden layer on the current time point t alongside the corresponding element of the sequence x_t. Mathematically, the Elman Neural Network can be described as follows:

$$h_{j,t} = f(w_{h_j} x_t + w_{c_j} h_{j,t-1} + b_{h_j}) \tag{2}$$

where $h_{j,t}$ is the output of the unit j from the hidden layer at time point t, f an activation function (usually $tanh$), w_{h_j} the learned weights on that unit for the input, w_{c_j} the learned weights on that unit for its own output on the previous time point (stored in the context unit) and b_{h_j} the bias of the unit.

Long-Short Term Memory (LSTM) [7] neural networks were created by Hochreiter and replace the standard hidden neuron with LSTM units. Each LSTM unit is formed by two recurrent data vectors: the hidden state and the cell state; and three gates: input gate, forget gate and output gate. The hidden state works as a short-term memory whilst the cell state works as long-term memory. The three gates work as masks that control the information flow in and out of the cell state. All three gates have their own weights and use the sigmoid function to ensure the [0,1] range. Mathematically, the LSTM cell works as follows:

- **Forget gate:** $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ (3)
- **Input gate:** $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ (4)
- **Candidate cell state:** $\widetilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$ (5)
- **Output gate:** $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$ (6)
- **Cell state:** $C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t$ (7)
- **Hidden state:** $h_t = o_t \cdot tanh(C_t)$ (8)

2.3. Time series symbolization.

Time series symbolization techniques transform a raw numerical time-series $T = [T_0, T_1, T_2, ..., T_n]$ to a sequence of symbols of lower length $S = [S_0, S_1, S_2, ..., S_m]$. Symbolization is used as a numerosity reduction technique in many time series data mining tasks, especially those that rely on distance computation, such as pattern mining and anomaly detection. Their

objective is to provide a simpler representation of time series that reduce computational complexity and storage requirements while preserving relevant information.

Time series symbolization often relies on a two-step process based on time series segmentation and extracting symbols that represent the relevant information from each segment. Symbolic Aggregate approXimation (SAX) [3] is the most used symbolization technique in the literature. The segmentation in SAX is made making use of Piecewise Aggregate Approximation (PAA), which divides the original time series into equidistant segments and returns the mean value from each segment. Then, the mean value from each segment is discretized making use of an interval-based lookup table. Each interval is an equiprobable area under the Gaussian curve. The number of intervals in the lookup table corresponds to the number of unique symbols that can appear on the resulting sequence (size of the alphabet). SAX works under the assumption of normality in the original time series and two parameters provided by the algorithm user: the size of the alphabet and the size of the segments.

While SAX has offered great results in many applications [15, 16], it is a common opinion from various authors the information from just the mean may not suffice depending on its application. Thus, there are many proposals of SAX variants that try to address some of its issues. Extended SAX (ESAX) [17] uses the minimum, maximum and mean of every segment instead of just the mean. Trend-based SAX (TSAX) [18] uses the mean and a new symbol to represent the trend. The trend is calculated by splitting each segment in half and looking at the subtraction of the means of the subsegments. It can take three values: stable (the absolute difference is lower than a minimum value), up or down. TFSAX [19] incorporates a new symbol that represents the trend. The trend is calculated as the arctangent of the ratio between the trend distance and the number of turning points in the segment and is discretized by using the same procedure as SAX's mean value. Adaptive SAX (aSAX) [4] uses the Lloyd algorithm to find a new set of breakpoints that should resemble better the original data distribution than the assumption of normality from SAX.

In our experimentation, we implemented the SAX and aSAX symbolization techniques. The use of other SAX variants was discarded because

they use more than one symbol per segment. While the use of more symbols per segment preserves more information, it also makes the prediction harder since we would need to forecast accurately multiple symbols at the same time. Furthermore, most proposals do not offer a way to transform the symbolic representation into a numerical one as they were designed for other data mining tasks, such as classification or clustering, providing almost no benefit for the forecasting task and requiring more time to train.

2.4. Methodology

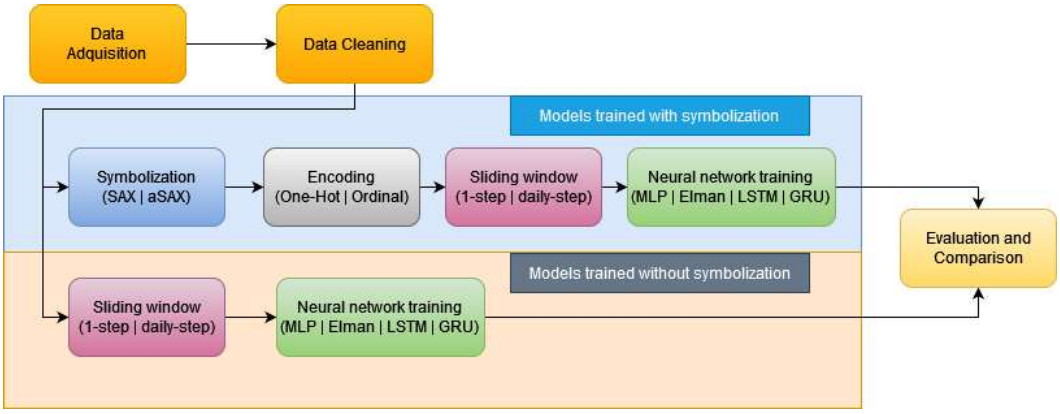

Figure 3: Applied methodology flowchart.

In order to see which approach works better (neural networks with or without symbolization), we trained models under similar conditions, as we can observe in figure 3. In the case of models without symbolization, after the data was cleaned, a model is trained with the samples provided by a sliding window whose size corresponds to two days and the next sample is obtained moving the sliding window either to the next observation (1-step) or the number of observations corresponding to two days (daily-step). Then, the selected neural network architecture is trained by making use of the mean squared error as loss function. In the case of models with symbolization, prior to the sliding window, the corresponding symbolization technique is applied and an encoding algorithm transforms the symbolic representation into a suitable input for the neural network. Afterwards, the sliding window and neural network will be applied in the same way that models without symbolization except by the fact that the loss function will be determined by

the encoding algorithm used. Once all models were trained, we made a comparative analysis to compare both the differences between models with and without symbolization and what parameters in the preprocessing pipeline lead to better models. In this section, we will provide and explain the different parameters tested in the methodology.

2.4.1. Symbolization parameters.

The selected symbolization techniques, SAX and aSAX require the user to provide a segment size and an alphabet size. Finding those values is a non a non-trivial task. Longer segments will speed up the posterior training process but could offer less accurate forecasts while extremely short segments will barely give any benefit over using the time series without symbolization. High cardinality alphabets will represent better the original time series but could make the forecasting task more difficult. Particularly cases in which some symbols appear too many times in comparison with the others or some symbols barely used. However, if the alphabet size is too small we may be losing relevant information.

The selection of these parameters has a huge impact on the interpretation of the symbolic time series. In our experimentation, we decided to use a segment size of 6, since each symbol of the time series will be representing the mean demand during an hour and can provide a considerable speedup while preserving enough information. For the alphabet size, we tested two values: 7 and 13. These two values correspond to the largest alphabet size in which each symbol is observed in at least 10% and 5% of the observations, respectively. We decided not to use larger alphabets since, for our data, they lead to excessively imbalanced distributions in which some symbols may appear too many times and other symbols may not appear at all.

2.4.2. Encoding.

A well-known way to train an artificial neural network with a symbolic sequence is by using a hot-encoding representation. Each symbol s of the a different symbols is represented by a unique vector of all zeros except a one in the s^{th} position of the vector. This is a common and successful approach in a task such as text mining, where this encoding is combined with the softmax activation in the output layer and the cross-entropy loss function to train neural networks. This approach usually leads to high accuracy models

but does not take into account how far each predicted symbol is from the expected symbol.

In the case of energy demand, we deem more useful forecasts that may be slightly less accurate but penalize the distance between predicted and expected value. This is the reason we propose the incorporation of the ordinal regression encoding proposed by Cheng et al. [20] to forecast symbolic time series. In this approach, the neural network is trained to learn the probability $o = (o_1, o_2, ..., o_i, ..., o_a)$ of a given value x being lower than the i^{th} symbol, where $o_i(i \leq a)$ is close to one and $o_i(i \geq a)$ is near zero. The encoding represents the i^{th} symbol with a vector of a numbers with all ones up to position number i filled with zeros afterwards and requires the use of the sigmoid activation on the output layer and the mean squared error as loss function.

2.4.3. Sliding Window.

Since our objective is to make daily forecasting with an univariate time series, a sliding window algorithm was used to extract the samples. The size of the sliding window was set to two days. The observations corresponding to the first 24 hours are provided to neural network as the input signals and the next 24 hours are the desired output signals. This was set after trying out multiple input sizes from the use of just a few hours to an entire week. Two values were selected for the sliding window step during our experimentation. Choosing a step size of 1 provides us with the maximum amount of training samples, giving us more information at the expense of more training time. This approach also creates flexible models that can be used to forecast the next 24 hours independently of the hour the sample start. Choosing a daily step size (24 for symbolic representations or 144 for numerical representations) will provide us with fewer samples, thus granting faster training and a lower risk of overfitting. This approach makes models that only work properly when the first observation of a sample is at 0:00. The daily step size was used for the validation and testing partitions.

2.4.4. Neural network parameters.

All neural networks models were trained using a many-to-many approach using the samples providing by the sliding window. We tested topologies with one hidden layer with a number of hidden units between 10 to 60 neurons (each value every 5 neurons was tested). Three different hidden activation

functions where tested: hyperbolic tangent, sigmoid and ReLU. Models were trained during up to 75 epochs with early stopping if the results do not improve for 10 epochs. We use the cross entropy loss function for the symbolic time series and mean squared error for the numeric time series. The learning rate when working with the symbolic representation was raised to 0.005 since with the default value of 0.001 it was not converging. All other parameters were kept to the default value of the *TensorFlow Keras* [21] framework, which was used for all experimentation. All calculations were made on a desktop computer with 8 GB of RAM and an AMD Ryzen 5 2600x running at 3.6 GHz. For reproducibility purposes, the random seed to initialize the weights of each model was set to 1996.

3. Discussion.

3.1. Forecasting performance metrics.

To evaluate the performance of the models we used the training time, three metrics for models with symbolization and two metrics for models without symbolization. For models with symbolization, we used the root mean squared error (RMSE), MINDIST and accuracy. Since RMSE was used for models with and without symbolization, we will refer to the RMSE used for models with symbolization as RMSE (Sym) for the remainder of the paper while RMSE alone refers to the numeric representation. In order to calculate the RMSE (Sym), since the metric requires a numerical value, each symbol is replaced with the integer that represents its position on the alphabet. For example, the first symbol, A, would be replaced with integer 1. The best model with symbolization for a specific alphabet size was selected making use of this metric, while the others are used to provide complementary information for the discussion. RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{N}} \tag{9}$$

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

MINDIST [3] is a distance measure proposed alongside SAX that lower bounds the euclidean distance of the corresponding PAA representation of

the time series. The benefit from it is that it allows us to compare results even if we make use of different alphabet size or segment size. MINDIST is calculated as follows:

$$MINDIST(a,b) = \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(dist(a_i,b_i))^2} \tag{10}$$

where a and b are two SAX (or aSAX) sequences, n is the numeric time series length, w is the symbolic time series length and $dist$ is defined as follows:

$$dist(r,c) = \begin{cases} 0, & if|r-c| \leq 1 \\ \beta_{max(r,c)-1} - \beta_{min(r,c)}, & otherwise \end{cases} \tag{11}$$

where β is a breakpoint from the symbolization lookup table.

The accuracy metric tells us the percentage of correct predictions and its purpose is to complement the RMSE metric. Accuracy is defined as follows:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{12}$$

In the case of models without symbolization, we made use of the RMSE and MAPE metric. MAPE is defined as follows:

$$MAPE = \frac{1}{N}\sum_{i=1}^{n}|\frac{y-\hat{y}}{y}| \tag{13}$$

Since we cannot directly compare models with and without symbolization, it is required to transform the representation after the forecast is done. We can evaluate how well a model with symbolization forecasts the numerical time series by transforming each symbol to the central value of the interval it represents and repeating that value as many times as long as the segment size, and we can evaluate how well the models without symbolization (numerical) can provide a symbolic forecast by using the symbolization technique after the forecast is done.

3.2. Preprocessing pipeline.

The preprocessing pipeline proposed in our methodology features multiple alternatives in each of its steps, such as the use of different encoding

algorithms. The first part of the discussion will be focused on the study of this preprocessing pipeline and, particularly, if there is any alternative that always outperforms the others. A summary of the models trained using the SAX symbolization algorithm is shown in table 1. For clarity reasons, the table only shows the model with the best number of neurons and activation (according to the RMSE (Sym) metric) per combination of all other parameters.

Table 1: Best topologies found for SAX based training. Bold values represent best metrics found for each alphabet size.

Alphabet size	**Architecture and Encoding**	**Window step**	**Neurons**	**Activation**	**RMSE (Sym)**	**MINDIST**	**Accuracy**
7	MLP [One-Hot]	Daily	40	ReLU	0.6903	1.0549	**0.7021**
7	Elman [One-Hot]	Daily	60	sigmoid	0.7421	1.1977	0.6759
7	LSTM [One-Hot]	Daily	25	ReLU	0.705	1.0249	0.682
7	MLP [One-Hot]	1	60	ReLU	0.8116	1.4945	0.6548
7	Elman [One-Hot]	1	45	sigmoid	0.906	1.8182	0.6085
7	LSTM [One-Hot]	1	60	ReLU	0.7592	1.272	0.6674
13	MLP [One-Hot]	Daily	50	tanh	1.188	1.7053	0.5566
13	Elman [One-Hot]	Daily	55	sigmoid	1.2734	1.8804	0.5166
13	LSTM [One-Hot]	Daily	60	ReLU	1.1936	1.6802	0.5421
13	MLP [One-Hot]	1	60	ReLU	1.4039	2.2482	0.5068
13	Elman [One-Hot]	1	60	sigmoid	1.4129	2.2889	0.484
13	LSTM [One-Hot]	1	55	ReLU	1.3105	1.9718	0.4952
7	MLP [Ordinal]	Daily	60	sigmoid	**0.6366**	**0.7121**	0.6888
7	Elman [Ordinal]	Daily	45	ReLU	0.704	0.8755	0.6469
7	LSTM [Ordinal]	Daily	60	ReLU	0.6678	0.7472	0.6676
7	MLP [Ordinal]	1	55	sigmoid	0.7638	1.1066	0.6195
7	Elman [Ordinal]	1	50	sigmoid	0.7866	1.1643	0.6077
7	LSTM [Ordinal]	1	60	ReLU	0.7279	1.0484	0.655
13	MLP [Ordinal]	Daily	60	ReLU	**0.9893**	**1.1032**	**0.5584**
13	Elman [Ordinal]	Daily	50	ReLU	1.1399	1.417	0.4591
13	LSTM [Ordinal]	Daily	55	ReLU	1.0132	1.1006	0.5133
13	MLP [Ordinal]	1	45	sigmoid	1.2852	1.795	0.4451
13	Elman [Ordinal]	1	50	sigmoid	1.3575	2.0101	0.4402
13	LSTM [Ordinal]	1	60	ReLU	1.1763	1.5321	0.4888

The summary table shows that for every pair of models that share architecture, alphabet size and sliding window step but have different encoding, most models with one-hot encoding provide better accuracy than ordinal models while all models with ordinal encoding provide better RMSE (Sym) than models with ordinal encoding. This was the expected behaviour and verifies that whenever how far away the prediction is from the expected value

is relevant the ordinal encoding should be preferred. In the case of the sliding window, the use of a daily step outperforms the use of a step size of one. Therefore, the use of a daily step sliding window offers both better metrics and faster training time, since the daily step size will generate a lower amount of samples to use for training. Also, the most simple neural network architecture, the MLP, provides better results than all the other recurrent architectures.

While the RMSE (Sym) metric allows us to compare models with the same alphabet size, we cannot directly use it to compare models with alphabet sizes that differ. The MINDIST metric, on the other hand, is suitable to compare different alphabet sizes but is only a lower bound of their true PAA euclidean distance. The MINDIST metric is usually worse for the models that use symbolization with alphabets of 13 symbols than 7. This is expected since a higher alphabet size makes the forecast more difficult. However, since it is a lower bound, it is completely possible that an alphabet size of 13 outperforms the alphabet size of 7 if we compare the results after transforming the symbolic representation back to a numerical one.

3.3. Comparison between SAX and aSAX.

With aSAX, the use of daily step size for the sliding window and the use of the ordinal encoding did also outperform the other alternatives. Table 2 contains a summary of the best models found making use of aSAX.

Table 2: Best topologies found for aSAX. All models in this table use ordinal encoding and a daily sliding window step.

Alphabet size	**Architecture**	**Neurons**	**Activation**	**RMSE (Sym)**	**MINDIST**	**Accuracy**
7	MLP	45	ReLU	**0.6181**	**0.6678**	**0.6664**
7	Elman	45	ReLU	0.6794	0.8843	0.6322
7	LSTM	40	ReLU	0.6496	0.6356	0.6303
13	MLP	25	ReLU	**0.9548**	**1.088**	**0.525**
13	Elman	35	ReLU	1.1094	1.4289	0.4258
13	LSTM	55	ReLU	0.9873	1.0948	0.4698

The use of aSAX instead of SAX leads us to models that forecast better their symbolic representation than their SAX counterparts. They also make

use of a lower amount of neurons, providing lower complexity models. The best models with aSAX always make use of the ReLU activation function. Since the only difference between SAX and aSAX, is the interval each symbol covers, we can understand the reason behind the better performance by taking a closer look into them.We can observe the interval breakpoints for our training data in table 3 and how many times each symbol appears on the training data in figure 4.

Table 3: SAX and aSAX breakpoints for our training data.

		1	2	3	4	5	6	7
SAX	Lower bound	$-\infty$	23.3326	25.8686	27.8197	29.6381	31.5909	34.1268
	Upper bound	23.3326	25.8686	27.8197	29.6381	31.5909	34.1268	∞
aSAX	Lower bound	$-\infty$	22.2939	24.9574	27.7179	30.4538	33.148	36.287
	Upper bound	22.2939	24.9574	27.7179	30.4538	33.148	36.287	∞

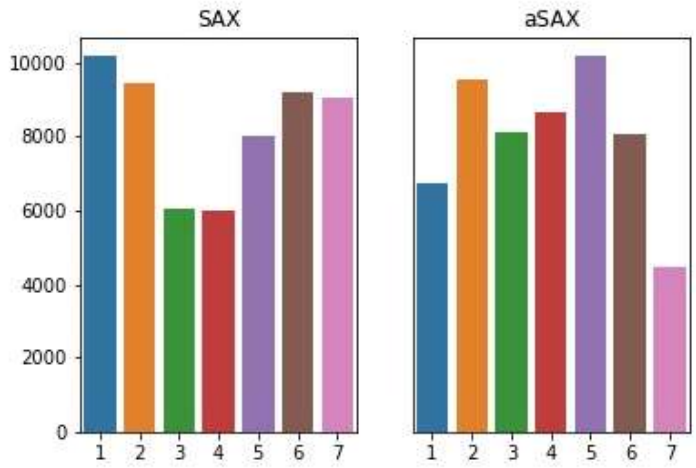

Figure 4: Symbol distribution on training data for symbolization techniques with an alphabet size of 7.

While using the SAX symbolization technique, the symbols that cover the extreme values appear many more times than symbols that cover areas in the center of the distribution. The selection of these breakpoints is a byproduct of the fact that the original data did not have a normal distribution and will result in a bias towards predicting symbols 1 and 7. Since

aSAX does not require the normality assumption we can observe how the algorithm finds a more balanced set of breakpoints that deals with the imbalance problem although it creates certain imbalance particularly against forecasting the symbols of highest electric demand. Another way to see the impact of this interval selection is to observe the density plot provided in figure 5, where the area between two vertical lines (including the vertical edges of the figure) represents the density covered by each symbol). In SAX, we can easily observe how most density is under the extreme symbols while aSAX provides a much more balanced interval distribution.

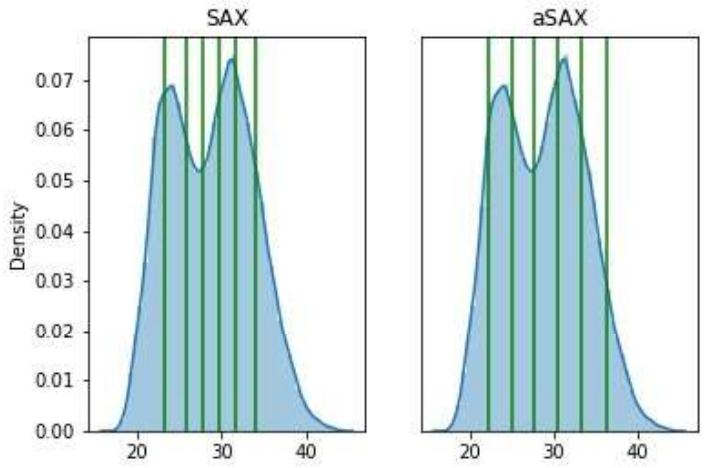

Figure 5: Comparison of intervals provided by SAX and aSAX.

3.4. Comparison between models with and without symbolization.

After concluding that for models with symbolization, the use of ordinal encoding, a sliding window with a daily step size, the MLP architecture and aSAX provided better symbolic forecasts, we need to train models without symbolization in order to compare both approaches. Table (table 4) displays the metrics of the best models trained without symbolization.

The best model found when training models without symbolization techniques is a LSTM with 55 units in its hidden layer and the hyperbolic tangent activation function, which provides an RMSE of 1.2889 on our test data.

Table 4: RMSE and MAPE on test partition for the best models without symbolization.

Architecture	**Window step**	**Neurons**	**Activation**	**RMSE**	**MAPE**
MLP	Daily	60	ReLU	1.5542	0.0434
Elman	Daily	30	ReLU	2.0766	0.0601
LSTM	Daily	20	ReLU	1.8408	0.0531
MLP	1	25	ReLU	1.553	0.0445
Elman	1	55	ReLU	2.0146	0.0591
LSTM	1	55	tanh	1.2889	0.0363

Contrary to the use of the symbolic representation, the best models with the numerical representation make use of a step size of one. Thus, the models without symbolization required the use of a much higher amount of training samples and, therefore, a higher training time. Lastly, we will compare the performance of the trained models with and without symbolization. This comparison will be split in two parts: the forecast of the symbolic representation and the forecast of the original time series. Table 5 shows the performance that the models trained without symbolization offered to forecast the symbolic representation. This implies that after the model did the forecast with the numerical representation the symbolization techniques were applied.

As expected, when working with models trained without symbolization, the best performance to forecast the symbolic representations is provided by the best model found to forecast the numeric representation (LSTM with 55 units in its hidden layer and the hyperbolic tangent activation function). However, it underperforms in comparison with the models trained with symbolization. For example, a MLP with 60 neurons, the sigmoid activation function, ordinal encoding and a daily step sliding window (table 1) scored a symbolic RMSE of 0.6366 and an accuracy of 68.88 % being trained with SAX and an alphabet size of 7. However, its counterpart trained without symbolization and being symbolized according to SAX algorithm after the numeric forecast only scored a symbolic RMSE of 0.7209 and an accuracy of 59.99 %, much better than the 0.7209 and 59.99 %. Similar situations occur when comparing each model without symbolization against the corresponding model with symbolization. Therefore, whenever the symbolic output can be interpreted in a useful way models with symbolization should be preferred as they work better and faster.

Table 5: Symbolic forecast metrics for the best models trained without symbolization.

Symbolization (Alphabet size)	Window step	Architecture	Neurons	Activation	RMSE (Sym)	MINDIST	Accuracy
SAX (7 symbols)	Daily	MLP	60	ReLU	0.8166	1.3498	0.5193
SAX (13 symbols)	Daily	MLP	60	ReLU	1.3535	2.2961	0.3646
aSAX (7 symbols)	Daily	MLP	60	ReLU	0.7195	0.8507	0.564
aSAX (13 symbols)	Daily	MLP	60	ReLU	1.2293	1.8506	0.3799
SAX (7 symbols)	Daily	Elman	30	ReLU	1.0485	2.582	0.4218
SAX (13 symbols)	Daily	Elman	30	ReLU	1.8149	3.7256	0.2821
aSAX (7 symbols)	Daily	Elman	30	ReLU	0.9165	1.7876	0.4476
aSAX (13 symbols)	Daily	Elman	30	ReLU	1.6416	3.0854	0.2866
SAX (7 symbols)	Daily	LSTM	20	ReLU	0.91	1.9002	0.4755
SAX (13 symbols)	Daily	LSTM	20	ReLU	1.5608	2.9747	0.3148
aSAX (7 symbols)	Daily	LSTM	20	ReLU	0.8221	1.2837	0.4869
aSAX (13 symbols)	Daily	LSTM	20	ReLU	1.4237	2.4285	0.3027
SAX (7 symbols)	1	MLP	25	ReLU	0.8586	1.466	0.4915
SAX (13 symbols)	1	MLP	25	ReLU	1.4244	2.4491	0.3319
aSAX (7 symbols)	1	MLP	25	ReLU	0.7514	0.9146	0.5347
aSAX (13 symbols)	1	MLP	25	ReLU	1.2873	1.9861	0.3522
SAX (7 symbols)	1	Elman	55	ReLU	0.9908	2.3953	0.4693
SAX (13 symbols)	1	Elman	55	ReLU	1.7386	3.4882	0.3144
aSAX (7 symbols)	1	Elman	55	ReLU	0.8771	1.6637	0.4907
aSAX (13 symbols)	1	Elman	55	ReLU	1.5838	2.901	0.3204
SAX (7 symbols)	1	LSTM	55	tanh	0.7209	1.0039	0.5999
SAX (13 symbols)	1	LSTM	55	tanh	1.1581	1.7187	0.4419
aSAX (7 symbols)	1	LSTM	55	tanh	**0.6502**	**0.6281**	**0.6297**
aSAX (13 symbols)	1	LSTM	55	tanh	**1.0597**	**1.3673**	**0.456**

Another use case of the symbolization techniques is to provide a fast approximation to forecast the numeric time series. In this case, after the symbolization technique is applied each symbol is replaced by the central value of the interval it represents and the same value is repeated as many times as long was the segment size. Table 6 showcases the scored offered by the models trained with symbolization after transforming them to a numerical representation as well as different models without symbolization.

With the use of the daily step sliding window, symbolization techniques outperform the numerical representation while the opposite happens with the use of a step of one. The use of a bigger alphabet size improves the performance of the symbolic models. The best performing symbolic model is a MLP with 25 hidden neurons and the ReLU activation making use of the aSAX symbolization method with an alphabet size of 13 unique symbols offering a RMSE of 1.3655 and a MAPE of 0.0390 on test data. The best numeric model offers slightly better performance with an RMSE of 1.2889

Table 6: Original 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
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	MLP [Ordinal]	55	ReLU	1.8384	0.0531	58.4093
aSAX (13 symbols)	1	MLP [Ordinal]	25	ReLU	1.7743	0.0503	100.6284
Numeric	Daily	MLP	60	ReLU	1.5542	0.0434	8.5964
Numeric	1	LSTM	55	tanh	**1.2889**	**0.0363**	40959.7513
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	15484.5837
Numeric	1	Gradient Boosting Trees	max_depth: 20 n_estimators: 150 learning_rate: 0.1		1.4900	0.0422	22284.1009

*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 correspond to scikit-learn default values. Multi-step forecast is done recursively.

and a MAPE of 0.0363. However, the training time required for the numeric model was 40599.7513 seconds while the best symbolic model required only 6.8402 seconds to train. However, the best symbolic model still outperforms other algorithms that can be used for forecasting such as Random Forests or Gradient Boosting Tress [22] while being trained faster than them. Figure 6 depicts the accuracy differences between the symbolic forecast and the numeric forecast over the span of a week. Further improvements may be accomplished by exploring different alphabet and segment sizes and other symbolization techniques.

3.5. Advantages, limitations and use cases of the proposed approach.

As we can observe in the conducted experimentation, the main advantage of using symbolization techniques for time series forecasting is the training time speedup. The ideal scenario to apply the symbolization techniques happens whenever the symbolic forecast can be used in a posterior decision-making process. For example, with the data we studied, since each symbol represents an interval for the mean demand during an hour, the symbolic forecast could help plan the production and importation of energy as long as an expert can establish a relationship between the symbols and the available production and importation for the electric grid or if instead of the proposed

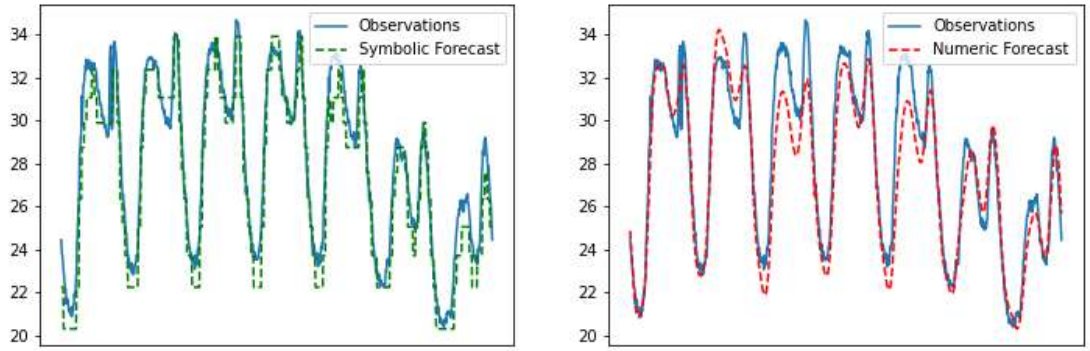

Figure 6: Prediction plot over the span of one week of the test partition.

algorithms the intervals were already provided by an expert.

Nevertheless, the main use case for these techniques is to speed up the training of models when we have massive amounts of data. The use of any symbolization technique will usually lead to slightly worse but relatively accurate performance metrics [2]. This is due to the fact that when are using symbolization techniques we are limited to the forecasting of an approximation of the time series.

Lastly, this approach is not appropriate for all kind of data. Due to the fact that the symbolization will always lead to some information loss, there will be an instance in which transforming the symbolic forecast to a numerical one will barely resemble the expected results. This will usually happen when the difference between consecutive observations is too big, hence the mean value will not properly represent the segment. Therefore, obtaining good results with symbolization techniques require a certain degree of smoothness from the time series used.

4. Conclusion.

In this paper, we studied the use of symbolization techniques for electric demand forecasting. Experimentation made use of the demand data of the main Spanish electric network with observations taken from 2009 to 2019 every 10 minutes. We evaluated different ways to train neural networks with

symbolic time series and compared our best symbolic models with our best numeric models. The use of an ordinal encoding, preserving the notion of order, improved the performance metrics when compared to the classical one-hot encoding. We evaluated which approach performed better to forecast the time series symbolic and numerical representations. Symbolic models outperformed numerical models when forecasting the symbolic representation. When forecasting the numerical representation, symbolic models provided us with a comparable but slightly worse forecast. However, symbolic models had a lower complexity and trained much faster than the best numerical models. Future improvements may be made with the development of new symbolization techniques, other machine learning models, including the symbolization on more complex methodologies or by adding relevant external information for our problem.

Acknowledgments

We acknowledge financial support from the Ministerio de Ciencia e Innovación (Spain) (Research Project PID2020-112495RB-C21) and the I+D+i FEDER 2020 project B-TIC-42-UGR20. LGB Ruiz was supported by "Next Generation EU" Margaritas Salas aids.

Abbreviations

ANN	Artificial Neural Network
aSAX	Adaptive SAX
LSTM	Long-Short Term Memory
MLP	Multi-Layer Perceptron
SAX	Symbolic Aggregate Approximation

References

[1] D. Syed, S. S. Refaat, H. Abu-Rub, O. Bouhali, Short-term power forecasting model based on dimensionality reduction and deep learning techniques for smart grid, in: 2020 IEEE Kansas Power and Energy Conference (KPEC), 2020, pp. 1–6. doi:10.1109/KPEC47870.2020.9167560.

[2] S. Elsworth, S. Güttel, Time series forecasting using lstm networks: A symbolic approach, Unpublished results (Preprint). (03 2020).

[3] J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing sax: A novel symbolic representation of time series, Data Mining and Knowledge Discovery 15 (2007) 107–144. doi:10.1007/s10618-007-0064-z.

[4] N. D. Pham, Q. L. Le, T. K. Dang, Two novel adaptive symbolic representations for similarity search in time series databases, in: 2010 12th International Asia-Pacific Web Conference, 2010, pp. 181–187. doi:10.1109/APWeb.2010.23.

[5] L. B. Almeida, Multilayer perceptrons, in: Handbook of Neural Computation, IOP Publishing Ltd and Oxford University Press, 1997.

[6] J. Elman, Finding structure in time, Cognitive Science 14 (1990) 179–211. doi:10.1016/0364-0213(90)90002-E.

[7] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–80. doi:10.1162/neco.1997.9.8.1735.

[8] C. Siridhipakul, P. Vateekul, Multi-step power consumption forecasting in thailand using dual-stage attentional lstm, in: 2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE), 2019, pp. 1–6. doi:10.1109/ICITEED.2019.8929966.

[9] A. Azadeh, S. Ghadrei, B. P. Nokhandan, One day ahead load forecasting for electricity market of iran by ann, in: 2009 International Conference on Power Engineering, Energy and Electrical Drives, 2009, pp. 670–674. doi:10.1109/POWERENG.2009.4915144.

[10] R. Ehsan, S. P. Simon, P. R. Venkateswaran, Day-ahead forecasting of solar photovoltaic output power using multilayer perceptron, Neural Computing and Applications 28 (2016) 3981–3992.

[11] A. Reinhardt, S. Koessler, Powersax: Fast motif matching in distributed power meter data using symbolic representations, in: 39th Annual IEEE Conference on Local Computer Networks Workshops, 2014, pp. 531–538. doi:10.1109/LCNW.2014.6927699.

[12] Y. Chen, J. Wen, 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), 2017, pp. 728–732. doi:10.1109/CCSSE.2017.8088030.

[13] C. Miller, Z. Nagy, A. Schlueter, Automated daily pattern filtering of measured building performance data, Automation in Construction 49 (2015) 1–17. doi:10.1016/j.autcon.2014.09.004.

[14] Red Eléctrica de España, Spanish peninsula electric network demand, `https://demanda.ree.es/visiona/peninsula/demanda/total` (accessed 21 June 2021).

[15] S. Benabderrahmane, N. Mellouli, M. Lamolle, On the predictive analysis of behavioral massive job data using embedded clustering and deep recurrent neural networks, Knowledge-Based Systems 151 (03 2018). doi:10.1016/j.knosys.2018.03.025.

[16] N. Potha, M. Maragoudakis, D. Lyras, A biology-inspired, data mining framework for extracting patterns in sexual cyberbullying data, Knowledge-Based Systems 96 (01 2016). doi:10.1016/j.knosys.2015.12.021.

[17] B. Lkhagva, Y. Suzuki, K. Kawagoe, New time series data representation esax for financial applications, 2006, pp. x115 – x115. doi:10.1109/ICDEW.2006.99.

[18] K. Zhang, Y. Li, Y. Chai, L. Huang, Trend-based symbolic aggregate approximation for time series representation, in: 2018 Chinese Control And Decision Conference (CCDC), 2018, pp. 2234–2240. doi:10.1109/CCDC.2018.8407498.

[19] Y. Yu, Y. Zhu, D. Wan, H. Liu, Q. Zhao, A novel symbolic aggregate approximation for time series, in: Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication, IMCOM 2019, 2019, pp. 805–822. doi:10.1007/978-3-030-19063-7_65.

[20] J. Cheng, G. Pollastri, A neural network approach to ordinal regression, in: IEEE Int. Jt. Conf. Neural Networks 2008 IJCNN 2008 IEEE World Congr. Comput. Intell, 2008, pp. 1279–1284. doi:10.1109/IJCNN.2008.4633963.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow,

A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, Tensorflow (version 2.0.4), Zenodo, 2021. doi:10.5281/zenodo.4725924.

[22] A. Galicia, R. Talavera-Llames, A. Troncoso, I. Koprinska, F. Martínez-Álvarez, Multi-step forecasting for big data time series based on ensemble learning, Knowledge-Based Systems 163 (2019) 830–841. doi:https://doi.org/10.1016/j.knosys.2018.10.009.