





Embedded Temporal Feature Selection for Time Series Forecasting Using Deep Learning

M. J. Jiménez-Navarro¹✉ , M. Martínez-Ballesteros¹ ,
F. Martínez-Álvarez² , and G. Asencio-Cortés² 

¹ Department of Computer Science, University of Seville, 41012 Seville, Spain
{mjimenez3, mariamartinez}@us.es

² Data Science and Big Data Lab, Pablo de Olavide University, 41013 Seville, Spain
{fmaralv, guaasecor}@upo.es

Abstract. Traditional time series forecasting models often use all available variables, including potentially irrelevant or noisy features, which can lead to overfitting and poor performance. Feature selection can help address this issue by selecting the most informative variables in the temporal and feature dimensions. However, selecting the right features can be challenging for time series models. Embedded feature selection has been a popular approach, but many techniques do not include it in their design, including deep learning methods, which can lead to less efficient and effective feature selection. This paper presents a deep learning-based method for time series forecasting that incorporates feature selection to improve model efficacy and interpretability. The proposed method uses a multidimensional layer to remove irrelevant features along the temporal dimension. The resulting model is compared to several feature selection methods and experimental results demonstrate that the proposed approach can improve forecasting accuracy while reducing model complexity.

Keywords: feature selection · embedded · neural network · time series · forecasting

1 Introduction

Embedded feature selection has become the preferred approach for feature selection due to its combination of simplicity, efficiency, and remarkable results. However, just some techniques include feature selection in its design, which force to rely on less efficient or effective methods. Deep learning is one example of technique which does not embed an effective feature selection, which is one of the causes for obtaining poor results for tabular data in spite of the remarkable results in areas like artificial vision and natural language processing.

Time series forecasting [7] is one of the most common tabular data in the industry and a critical area of research that aims to predict future values of a variable based on its past information. While traditional time series forecasting@

T.

https://doi.org/10.1007/978-3-031-43078-7_2

models often use all available variables, including potentially irrelevant or noisy features, this can lead to overfitting and poor performance. Feature selection can help address this issue by selecting the most informative variables in the temporal and feature dimensions. However, selecting the right features and the right moments from the past information can be challenging for time series models.

In this paper, we introduce a novel method for time series forecasting called Time Selection Layer (TSL)¹, which extends the FADL framework proposed by Jimenez Navarro et al. [6]. A deep learning-based method is proposed for time series forecasting that incorporates feature selection to improve the efficacy and interpretability of the model. The proposed method includes an additional layer at the top of the model which selects the relevant features and time steps. This layer is trained during the backpropagation process, which, once the training process has finished, removes irrelevant features without additional training steps nor complex mechanisms. The previous methodology selected the features and all the past moments, which was incompatible with time series forecasting scenarios where a univariate time series or all the input features were used in the prediction.

The resulting model is compared to a baseline model that uses all available features and three filter-based methods because they do not require extra computation, and experimental results demonstrate that the proposed approach can improve forecast accuracy while reducing model complexity in all scenarios. Additionally, the selected features provide insight into the underlying patterns and drivers of the time series, aiding in the interpretation of the forecast results.

The main contributions of this paper are as follows:

- Simple and effective solution for feature selection in time series with minimal/no parametrization.
- General-purpose approach, which may be applied to almost any type of time series independently of its nature or task.
- Efficient interpretability approach which requires no extra computation.
- Comparison between 9 different time series forecasting datasets and 4 baseline approaches with a remarkable improvement in efficacy.

The rest of the article is organized as follows. In Sect. 2, we provide a brief overview of related work on time series forecasting and feature selection. In Sect. 3, we describe the proposed method in detail. Section 4 presents the experimental setting used to compare the different datasets and feature selection approximations. In Sect. 5, the experimental results and an analysis of the selected features of the proposed method between all datasets and feature selection methods are reported. Finally, Sect. 6 discusses the main conclusions of this work.

¹ Python implementation and experimentation have been included in the following repository <https://github.com/manjimnav/TSSLayer>.

2 Related Works

This section conducts a literature review of embedded feature selection methods applied to neural networks. We will review the advantages and limitations of each technique and discuss its applicability to different types of datasets and problems.

Yuan et al. [10] propose an embedded feature selection method for neural networks using the point-centered convolutional neural network. The study is applied to moldy peanuts identification problem using hyperspectral images. The authors propose using the weights in the first convolutional layer as an estimator of the relevance of the neural network for each band. The method provided competitive accuracy compared to conventional approaches. In our work, the goal is to filter irrelevant features by removing or keeping the input with the same value. In the work proposed by the authors, it is possible to assign a weight with a continuous value, which may difficult to determine if a feature is relevant.

Zhang et al. [11] propose to use the Group Lasso penalty to embed the feature selection in a neural network. The authors apply their method to a set of well-known baseline datasets compared with other regularization approaches. The groups in the Group Lasso penalty represent all the weights in the first layer connected to a layer. Each group has a regularization using a smoothing approximation to the Lasso penalty to make it fully differentiable. However, the proposal described may be only applied to feed-forward layers, which limits the applicability to other models. In our work, we propose a general purpose method for any type of neural networks.

Cancelan et al. [2] propose the E2E-FS method to embed feature selection in neural networks. The authors apply this method to different microarray challenges and artificially modified datasets for feature selection. The E2E-FS includes an additional loss function method in order to filter a fixed number of features and remove the influence of irrelevant features. In our work, the method automatically selects the amount of features, which usually is a subset of the total features. In addition, no change to the loss function is needed to filter the features as in our approach, as we approximate the Heaviside function to make it differentiable.

Borisov et al. [1] propose a general-purpose layer called CancelOut to filter irrelevant features. The method is applied to three baseline datasets compared to other embedded and not embedded feature selection methods. The methodology proposed by the authors consists of the use of an element-wise multiplication between the inputs and a set of filters. The filters are built by applying a sigmoid function to a set of weights, and the goal is to remove the influence of the irrelevant input features. However, for feature selection, a threshold is needed, which may have a great influence in the results. In our work, no threshold must be parameterized, which makes our method more generalizable.

3 Methodology

This section is divided into three subsections to describe the main methodology. In Sect. 3.1, the nomenclature used during the figures, formulas, and explanations is provided. In Sect. 3.2, the methodology is described in detail, focusing on the changes from the previous work and its implications. In Sect. 3.3 the weight initialization and regularization strategies for the TSL are described.

3.1 Nomenclature

In this section, the main elements used in the description of the TSL are reported.

- D represents the number of features used for the neural network. Note that in univariate time series $D = 1$.
- M represents the number of past moments used in the input, also called the window.
- X represents the input matrix with size MxD .
- W_L represents the TSL weights with size MxD .
- \hat{H} represents the Heaviside function approximation.
- \circ represents the Hadamard product.
- H represents the Heaviside function.
- σ represents the hard sigmoid function.
- δ represents the “detach” function which ignores the propagation of the gradient in the backpropagation algorithm.

3.2 Description

Figure 1 summarizes the suggested approach which involves the introduction of a new layer, referred to as the TSL, just after the input layer. Specifically, this layer operates on a per-element basis by linking each input to a distinct neuron within the TSL. These neurons act as a filter for the input, enabling to remove the irrelevant features from the network when required. Initially, all inputs are selected. Then, the TSL weights and the rest of the weights in the network are optimized together via backpropagation, with positive and negative weights assigned to each feature. Positive weights preserve the influence of the feature, while negative weights nullify the influence by setting it to zero. Due to the weights in TSL are optimized during the backpropagation process, the filter must collaborate to minimize the loss function embedding the feature selection process into the neural network design.

The filter in TSL is implemented by applying a Heaviside approximation to the matrix W_L setting the weights to zero or one. The resulting binarized weights are defined as the mask, which is multiplied by the input X using the Hadamard product, which removes the influence of the filtered inputs.

Therefore, the operation performed by the TSL is defined as follows:

$$TSL(X^{MxD}) = \hat{H}(W_L^{1xD}) \circ X^{MxD}. \quad (1)$$

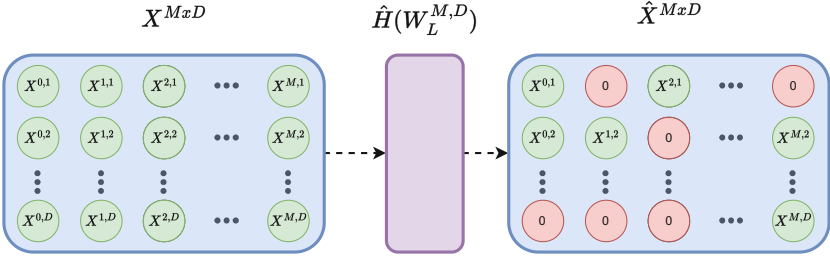


Fig. 1. TSL layer applied to an input matrix X with M past moments for D features. Note that the output \hat{X} has the same dimensions, but some moments were set to zero. Finally, the output is used as input for a neural network.

where the function H applies the following operation to the weight $W_L^{i,j}$, $i \in M, j \in D$:

$$H(W_L^{i,j}) = \begin{cases} 1, & \text{if } W_L^{i,j} \geq 0, \\ 0, & \text{if } W_L^{i,j} < 0, \end{cases} \quad (2)$$

Note that if a feature mask contains a zero weight for a feature at a specific moment, it will affect all the remaining layers and reduce the influence of the removed features. This fact may lead to a higher variance in training time compared to a neural network without TSL.

As the Heaviside function is not differentiable, an approximation was implemented for this function with a differentiable version. The operation was performed for each weight $W_L^{i,j}$ in W_L as follows:

$$\hat{H}(W_L^{i,j}) = \sigma(W_L^{i,j}) - \delta(W_L^{i,j} - H(W_L^{i,j})). \quad (4)$$

Using this approximation, the gradients can propagate through the TSL using the Heaviside function without requiring extra parameterization or regularization.

3.3 Weight Initialization and Regularization

The weight initialization and regularization are essential to achieve a good selection. Note that this initialization and regularization are applied to the TSL independently of the rest of the neural network.

As mentioned in the previous section, initially all features and moments are selected, and during the training the features are selected. In this work, we may consider that selection uses a backward approximation for feature selection. For this reason, the weights are initialized as a positive value. However, a large positive value would introduce a bias to keep the features, as reducing the weights would require more iterations than near-zero values. Thus, the weights are initialized as a near-zero positive value of 0.01.

Regularization is responsible for penalizing the amount of selected inputs. In this case, we considered the same penalization independently of the feature or moment to avoid any inductive bias, which will introduce a penalization of 0.01 for every selected feature and moment.

4 Experimental Setting

This section aims to provide a comprehensive understanding of the experimental design and implementation, allowing for reproducibility and evaluation of the results divided into five subsections. Section 4.1 outlines the datasets used in the study, including their sources, sizes, and characteristics. Section 4.2 describes the various transformations applied to the datasets, such as normalization and division. Section 4.3 provides a detailed explanation of the hyperparameters used in the study, including their values and how they were chosen. Section 4.4 outlines the different feature selection methods used in the study, including any modifications or adaptations made. Finally, Sect. 4.5 describes the performance metrics used to evaluate the results of the study.

4.1 Datasets

The experiments were performed by selecting eight data sets consisting of different time series from various sources and fields, such as maintenance of the power transformer, consumption of electricity, pollution, etc. In Table 1, each dataset will be described in detail, including the number of features, sample frequency, and forecasting strategy. The forecasting strategy can be either many-to-many or many-to-one. Many-to-many strategy uses multiple features as input and outputs multiple features that may be the same as the input or a subset. Many-to-one strategy uses multiple features as input and only one feature as output, usually one of the input features.

Table 1. Summary of datasets used in the experiment

Dataset	Instances	Time Span	Features	Frequency	Strategy
ETT	17420	2016–2018	7	1 h	Many-to-one
Electricity	26304	2011–2014	320	15 min	Many-to-many
ExchangeRate	7588	1990–2016	8	1 day	Many-to-many
TorneoCO	96409	2005–2015	4	10 min	Many-to-one
TorneoNO ₂	96409	2005–2015	4	10 min	Many-to-one
TorneoO ₃	96409	2005–2015	4	10 min	Many-to-one
TorneoPM ₁₀	96409	2005–2015	4	10 min	Many-to-one
Traffic	7544	2015–2016	862	1 h	Many-to-many

The Electricity Transformer Temperature (ETT) [12] datasets consist of insulating oil temperature samples obtained from two power transformer in China.

The dataset has 17,420 records collected from 2016 to 2018, with data sampled every minute and hour. In this work, the hourly version from both transformers was used, called ETTh1 and ETTh2. The datasets consist of a multivariate time series containing five features: High UseFul Load (HUFL), High UseLess Load (HULL), Middle UseFul Load (MUFL), Middle UseLess Load (MULL), Low UseFul Load (LUFL), Low UseLess Load (LULL), and Oil Temperature (OT), which is the target. Therefore, the forecasting strategy used is the many-to-one approach, where the seven features are used as input, and only the oil temperature is the output. In this work, single-horizon forecasting is used for all datasets.

The Electricity [4] dataset contains electricity consumption measurements obtained from 370 clients in Portugal. In this work, the grouped version [8] is used, containing 320 time series collected from 2011 to 2014, with 26,304 records for each time series sampled every 15 min. The dataset consists of 320 consumption time series as features. The target is to predict all the time series; therefore, the forecasting strategy used is the many-to-many approach.

The ExchangeRate [4] dataset contains exchange rate measurements obtained from eight countries. The dataset has 7,488 records sampled every day from 1990 to 2016. The features of the data set consist of daily exchange rates from the eight countries. Australia, United Kingdom, Canada, Switzerland, China, Japan, New Zealand, and Singapore. The forecasting strategy used in this case is the many-to-many approach, which means that the eight features will be used as output.

The Torneo [5] dataset contains pollution and meteorological measurements obtained in Seville, Spain. The dataset has 96,409 records sampled every 10 min from 2005 to 2015. The features in the dataset consist of four pollutants (CO , NO_2 , O_3 , and PM_{10}) and three meteorological variables (Temperature, Wind direction, and Wind speed). The dataset was divided into four different datasets, each using a many-to-one forecasting strategy for each pollutant: Torneo CO , Torneo NO_2 , Torneo O_3 , and Torneo PM_{10} . As the number of target features is considerably small, this division was made to study the relevant features based on just one pollutant.

The Traffic [3] dataset contains the road occupancy rate of the California Department of Transportation. The dataset has 17,544 records collected from 862 sensors placed in California during 2015 and 2016, sampled every hour. The features consist of 862 sensors, using the many-to-many forecasting strategy.

4.2 Preprocessing

The data must be processed before using it in the experiment. First, the data are standardized to ensure that each feature has a mean of zero and a standard deviation of one. Then, a windowing process is applied to the time series, which divides the time series into fixed size windows with contain the information and is fed into the neural network. The size of the window is an important parameter that needs to be optimized based on the specific problem and data characteristics. The dataset is then divided into training, validation, and testing

sets using the 70%, 10%, and 20% of the data, respectively. Each split preserves the temporality between the instances and between themselves.

4.3 Hyperparameters Definitions

The hyperparameters configure the training process and have a great impact on its behavior. The hyperparameters have been divided into three groups: data, model, and feature selection hyperparameters.

The primary hyperparameter of the data is the window size. In this study, we used a different sequence size for each dataset. For ETT and Torneo datasets the window size is 12 or 24, in Electricity is 4 or 8, in ExchangeRate is 2 or 7 and Traffic 3 or 6. Furthermore, during the windowing process, we subsampled the datasets by building a window for all X records, as defined by the shift which is 24 h for ETT, Electricity and Torneo datasets while a shift of 3 instances were selected for ExchangeRate and Traffic. For the forecast step, only one value was selected.

Regarding the model, the hyperparameters were consistent across all datasets and methods. A simple neural network with two hidden layers was selected, where the number of neurons in the first hidden layer was equal to half of the input features, while a quarter of the input features was employed in the second hidden layer. During training, we utilized the Adam optimizer for 100 epochs. In addition, we used the early stop technique to stop the training process once the training loss did not decrease for at least 10 epochs.

For feature selection hyperparameters, two types exist depending on the approach. For filter methods, a threshold is employed as a hyperparameter, which can contain values ranging from 50% to 100% of total relevance, in increments of 5%. For the TSL, the single hyperparameter employed is the regularization term, which can be set to $1e^{-3}$, $1e^{-4}$, or $1e^{-5}$.

A grid search is used for all possible combinations of hyperparameters in order to find the best set of hyperparameters for each feature selection method and model. The search uses the training data to optimize the model with the set of hyperparameters and the metrics calculated over the validation set are used as the quality measure.

4.4 Methods

In this section, the main methods used for the comparison are detailed. First, the neural network is evaluated without feature selection as the baseline method. Then, three filter-based feature selections are employed: Linear, Correlation (Corr) and Mutual Information (MI). For the filter methods, a threshold must be optimized, as mentioned in Sect. 4.3. This threshold will determine the amount of features which represents the *thr* total relevance, being $thr \in [50, 100]$ the threshold percentage.

Linear selection uses the weights of a linear model with L1 regularization to determine the importance of features. Corr selection uses the Pearson correlation

with the target variable to determine the importance of the characteristic. MI selection uses the Mutual Information [9] between the inputs and the target variable for determining the feature importance.

4.5 Metrics

To evaluate the performance of the model, three different metrics will be used. To evaluate the efficacy of the different methods, the mean squared error (MSE) will be employed. The number of selected features will be used to relate the method with the best efficacy with the number of relevant features needed. The formula is as follows: $MSE(Y_{pred}^n, Y_{true}^n) = \frac{1}{N} \sum_{n=1}^N (Y_{pred}^n - Y_{true}^n)^2$. Finally, the total time used for each method including the hyperparameter optimization process will be detailed.

5 Results and Discussion

In this section, the results obtained for each selection method and dataset are reported. The discussion is divided into three sections. First, the best efficacy results are analyzed for each dataset and method. Then the best hyperparameters obtained based on the best configuration are commented. Finally, the number of features of the best methods in each dataset.

5.1 Efficacy

Table 2 presents the MSE obtained for each selection method and dataset, as well as the corresponding improvement in MSE compared to the NS case (no selection), which is reported in parentheses.

Table 2. Mean squared error obtained for each selection method and dataset. Note that the improvement (between 0 and 1) with respect to the nonselection is reported in parentheses.

Dataset	NS	Corr	Linear	MI	TSL
ETTh1	0,399	0,157 (0,61)	0,058 (0,85)	0,073 (0,82)	0,053 (0,87)
ETTh2	0,590	0,135 (0,77)	0,060 (0,9)	0,166 (0,72)	0,111 (0,81)
Electricity	2,366	2,366 (0,00)	1,963 (0,17)	2,366 (0,00)	0,869 (0,63)
ExchangeRate	0,645	0,645 (0,00)	0,408 (0,37)	0,645 (0,00)	0,387 (0,40)
TorneoCO	0,512	0,253 (0,51)	0,228 (0,55)	0,243 (0,53)	0,188 (0,63)
TorneoNO2	0,802	0,422 (0,47)	0,415 (0,48)	0,494 (0,38)	0,276 (0,66)
TorneoO3	0,919	0,420 (0,54)	0,251 (0,73)	0,373 (0,59)	0,278 (0,70)
TorneoPM10	0,715	0,507 (0,29)	0,440 (0,38)	0,538 (0,25)	0,415 (0,42)
Traffic	0,299	0,213 (0,29)	0,220 (0,26)	0,227 (0,24)	0,186 (0,38)

For the NS method, the MSE is consistently higher or equal to that of the other methods, indicating that feature selection generally enhances prediction performance in the context of time series forecasting.

The Corr and MI methods demonstrate a significant improvement in MSE compared to NS, except for the Electricity and ExchangeRate datasets, where they perform similarly to NS. This suggests that Corr and MI are not always reliable indicators of feature relevance.

The TSL method produces the best results in seven out of nine datasets, followed by Linear, which yields the best results in the remaining two datasets. It seems that there are datasets with a linear relationship between the target and the features, which justifies the remarkable results.

In conclusion, TSL appears to exhibit the most consistent and optimal performance overall, which seems to indicate that our method is more generalizable to other problems without losing efficacy.

5.2 Best Hyperparameters

This section reports the hyperparameters found for the selection methods with the best efficacy. The goal is to identify general patterns for the different selection methods.

Table 3. Best hyperparameters found for each selection algorithm and dataset based on the efficacy.

Dataset	Window size					Threshold			Regularization
	NS	Corr	Linear	MI	TSL	Corr	Linear	MI	TSL
ETTh1	24	12	24	12	12	0.60	0.60	0.60	5e-3
ETTh2	12	12	24	12	12	0.80	0.55	0.65	5e-4
Electricity	8	8	8	8	8	0.60	0.60	0.50	1e-2
ExchangeRate	3	3	7	3	3	0.50	0.85	0.50	5e-5
TorneoCO	24	24	24	24	24	0.50	0.50	0.70	1e-2
TorneoNO2	12	24	12	24	24	0.75	0.50	0.70	1e-2
TorneoO3	12	12	12	12	24	0.55	0.55	0.90	1e-2
TorneoPM10	12	24	12	24	24	0.95	0.60	0.85	5e-3
Traffic	6	6	6	6	6	0.60	0.75	0.85	5e-4

Table 3 displays the best hyperparameters identified for each dataset and selection method, including the window size for the input time series, the threshold for the feature selection methods, and the regularization term used in our proposed approach.

Regarding window size, no consistent pattern emerges between methods. In some cases, the optimal window size is the maximum allowed, while in others, a smaller size is preferred.

For the threshold, no consistent pattern is observed for each feature selection method. The values range from 50% to 95%, making it challenging to optimize this parameter as no universally applicable value can be identified.

Concerning regularization in TSL, the optimal values typically fall within the range of $5e-3$ to $1e-2$. Thus, a regularization term between these two values may be a suitable initial value.

5.3 Selected Features

In this section, the number of features selected for each feature selection method with optimal hyperparameters is analyzed.

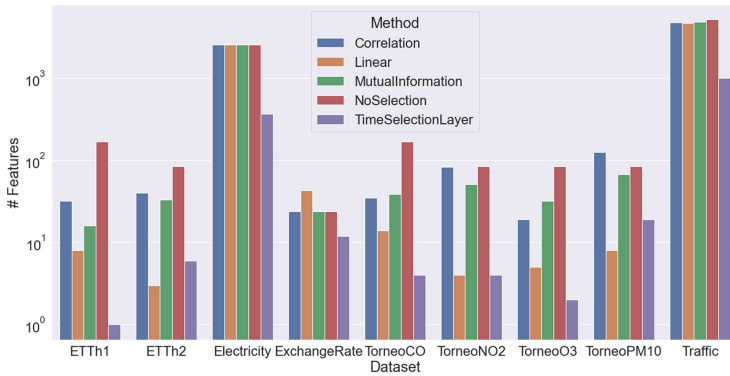


Fig. 2. Features obtained for each method using the best hyperparameters. Note the logarithmic scale.

Figure 2 displays the number of features obtained by each selection method. It is important to note that the number of features selected is not related to efficacy.

Overall, we observe that TSL tends to select fewer features than other methods. We hypothesize that the excellent efficacy results are a consequence of this selection strategy. Methods such as Corr or MI may not effectively filter out irrelevant features due to the high degree of interdependence among the features and moments. Linear appears to filter adequately when the feature space is small, but it may encounter difficulties when the number of original features is large.

6 Conclusions and Future Works

In this study, we have introduced a simple embedded feature selection approach for deep learning by adding a new layer. This layer acts as a filter that eliminates the impact of features with a temporal dimension. Our findings demonstrate, our proposed TSL outperforms other methods in most of the tested datasets with a straightforward parametrization and less information.

As future work, we intend to enhance the layer to consider the locality principle of selected moments. This involves increasing the likelihood of selecting moments around the chosen moments.

Acknowledgements. The authors would like to thank the Spanish Ministry of Science and Innovation for the support under the projects PID2020-117954RB and TED2021-131311B, and the European Regional Development Fund and Junta de Andalucía for projects PY20-00870, PYC20 RE 078 USE and UPO-138516.

References

1. Borisov, V., Haug, J., Kasneci, G.: CancelOut: a layer for feature selection in deep neural networks. In: Proceedings of 28th International Conference on Artificial Neural Networks. Artificial Neural Networks and Machine Learning - ICANN 2019: Deep Learning, pp. 72–83 (2019)
2. Cancela, B., Bolón-Canedo, V., Alonso-Betanzos, A.: E2E-FS: an end-to-end feature selection method for neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* pp. 1–12 (2020)
3. CDT: California department of transportation (2015). <https://pems.dot.ca.gov/>
4. Godahewa, R., Bergmeir, C., Webb, G., Hyndman, R., Montero-Manso, P.: Electricity hourly dataset (2020)
5. Gómez-Losada, A., Asencio-Cortés, G., Martínez-Álvarez, F., Riquelme, J.C.: A novel approach to forecast urban surface-level ozone considering heterogeneous locations and limited information. *Environ. Model. Softw.* **110**, 52–61 (2018)
6. Jiménez-Navarro, M.J., Martínez-Ballesteros, M., Sousa, I.S., Martínez-Álvarez, F., Asencio-Cortés, G.: Feature-aware drop layer (FADL): a nonparametric neural network layer for feature selection. In: Proceedings of 17th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2022), pp. 557–566 (2023)
7. Jiménez-Navarro, M.J., Martínez-Ballesteros, M., Martínez-Álvarez, F., Asencio-Cortés, G.: PHILNet: a novel efficient approach for time series forecasting using deep learning. *Inf. Sci.* **632**, 815–832 (2023)
8. Lai, G., Chang, W., Yang, Y., Liu, H.: Modeling long- and short-term temporal patterns with deep neural networks. *ACM*, pp. 95–104 (2018)
9. Shannon, C.E.: A mathematical theory of communication. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **5**(1), 3–55 (2001)
10. Yuan, D., Jiang, J., Gong, Z., Nie, C., Sun, Y.: Moldy peanuts identification based on hyperspectral images and point-centered convolutional neural network combined with embedded feature selection. *Comput. Electron. Agric.* **197**, 106963 (2022)
11. Zhang, H., Wang, J., Sun, Z., Zurada, J.M., Pal, N.R.: Feature selection for neural networks using group lasso regularization. *IEEE Trans. Knowl. Data Eng.* **32**(4), 659–673 (2020)
12. Zhou, H., et al.: Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11106–11115 (2021)