Random Hyper-parameter Search-Based Deep Neural Network for Power Consumption Forecasting

J. F. Torres(✉), D. Gutiérrez-Avilés, A. Troncoso(✉), and F. Martínez-Álvarez

Division of Computer Science, Pablo de Olavide University, Seville, Spain
{jftormal,dguvati,atrolor,fmaralv}@upo.es

Abstract. In this paper, we introduce a deep learning approach, based on feed-forward neural networks, for big data time series forecasting with arbitrary prediction horizons. We firstly propose a random search to tune the multiple hyper-parameters involved in the method perfor-mance. There is a twofold objective for this search: firstly, to improve the forecasts and, secondly, to decrease the learning time. Next, we pro-pose a procedure based on moving averages to smooth the predictions obtained by the different models considered for each value of the pre-diction horizon. We conduct a comprehensive evaluation using a real-world dataset composed of electricity consumption in Spain, evaluating accuracy and comparing the performance of the proposed deep learning with a grid search and a random search without applying smoothing. Reported results show that a random search produces competitive accu-racy results generating a smaller number of models, and the smoothing process reduces the forecasting error.

Keywords: Hyperparameters · Time series forecasting · Deep learning

1 Introduction

Deep learning is an emerging branch of machine learning that extends artificial neural networks. One of the main drawbacks that classical artificial neural networks exhibit is that, with many layers, its training typically becomes too complex. In this sense, deep learning consists of a set of learning algorithms to train artificial neural networks with a large number of hidden layers.

Deep learning models are also sensitive to a large numbers of hyper-parameters and much attention must be paid at this stage [6]. For Deep Feed Forward Neural Network (DFFNN), these hyper-parameters include the number of hidden layers, the number of neurons for hidden layers, the batch size and other parameters related to the optimization method used to compute the weights of the DFFNN in the training phase. There are many optimization methods such as gradient descend, gradient descend with momentum, RMSProp or Adam optimization algorithm, among others [14]. But the convergence of all of these algorithms depend on the learning rate, being one of the most important parameters.

Therefore, the task of selecting an appropriate set of hyper-parameters is critical for the performance of the DFFNN.

In this context, we propose a DFFNN for time series forecasting that implements a random search to find the best values for the most relevant parameters related to the network structure and optimization method to compute the weights of the network. With this strategy, we aim at improving the performance of the DFFNN in terms of both learning time and accuracy. In addition, we propose a smoothing technique as last step of the proposed methodology, in order to minimize the prediction error. To evaluate the performance of the proposed approach, we use a real-world dataset composed of electricity consumption in Spain, and we compare the results with those generated by a grid search and a random search without smoothing.

The rest of the paper is organized as follows. Section 2 reviews relevant works related to time series forecasting based on deep learning and to the tuning of hyper-parameters in deep learning. Section 3 introduces the methodology proposed in this paper. The most relevant results obtained by the methodology are discussed in Sect. 4. Finally, the conclusions drawn from this research work are summarized in Sect. 5.

2 Related Work

In this section, we analyze recent and relevant state-of-the-art proposals in the fields of deep learning time series forecasting and the hyper-parameter tuning and optimization of deep neural networks.

Deep learning approaches for time series analysis have been widely applied during the last years and, indeed, several strategies to predict future values with deep neural networks models have been developed. The authors in [7] presented, in 2015, a novel deep learning-based solution to forecast event-driven stock market values. In particular, a deep convolutional neural network was used obtaining a remarkable performance.

A paradigmatic example of an effort for improving the predictions performance through the network architecture can be found in [10]. There, the authors designed a stacked auto-encoder model for feature extraction to predict air quality. In the proposal presented in [5], a full revision of the input variables was carried out to decrease the computational time related to the training of the proposed deep learning approach for time series forecasting.

Due to the nature of these neural networks architectures and the considerable length of the current time series, distributed computation and data storage approaches play a relevant role in this field of study. In this sense, the authors in [15] proposed a deep feed-forward solution deployed along with the Apache Spark [17] platform for distributed computing to predict electricity consumption in Spain.

The hyper-parameter tuning and optimization of the deep neural networks is a fundamental factor for obtaining a competitive performance of the results. In this regard, the authors in [9] introduced a Bayesian method for hyper-parameter

optimization in which model the loss and the execution time in function of the dataset size. Random search and greedy methods for hyper-parameter tuning were applied in [1]. The authors concluded that the random search method can be useful in deep learning environments. The authors in [2] made a comparative study of three hyper-parameter optimization techniques: grid, experience-based, and random search methods. They concluded that the random one establishes a baseline to judge the performance of other hyper-parameter optimization algorithms.

Evolutionary strategies for optimization problems have been widely used, yielding competitive results. The authors in [16] addressed the hyper-parameter optimization problem with the approach mentioned above. Another specific approach for hyper-parameter optimization can be found in [8] where an efficient and deterministic method using radial functions was presented. Finally, in [11], the authors proposed a mixed strategy called Covariance Matrix Adaptation Evolution.

3 Methodology

This section describes the proposed methodology for time series forecasting using the DFFNN, which has been implemented in the H2O framework [3], under R language. It is also proposed an alternative method to the one implemented in H2O for the optimization of hyper-parameters and the use of a smoothing filter in order to minimize the impact of the time gap on each prediction. First, Sect. 3.1 describes a method for optimizing neural network hyper-parameters. After, Sect. 3.2 details the formulation that allows the multi-step forecast of a time series. Finally, the use of a smoothing filter to modulate the frequency of the prediction is introduced in Sect. 3.3. A complete workflow of the methodology proposed is illustrated in Fig. 1.

3.1 Hyper-parameters Tuning

It is well-known that the values of the hyper-parameters of the deep learning algorithm highly influence on the results. The algorithm implemented in H2O allows adjusting a large number of them, being worth highlighting some, such as the number of hidden layers or the number of neurons per layer or the learning rate.

In order to optimize the hyper-parameters described above, H2O implements two search options. One of them is a grid search, which performs an exhaustive search through the whole set of established hyper-parameters. The other one is a random search, which makes combinations of the defined hyper-parameters without a specific order or criteria. However, both search methods work with discrete values, which greatly limits the fine-tuning of the vast majority of hyper-parameters.

To avoid this problem, a random search is proposed in this article with continuous values. That is, given a set of hyper-parameters and a range for each

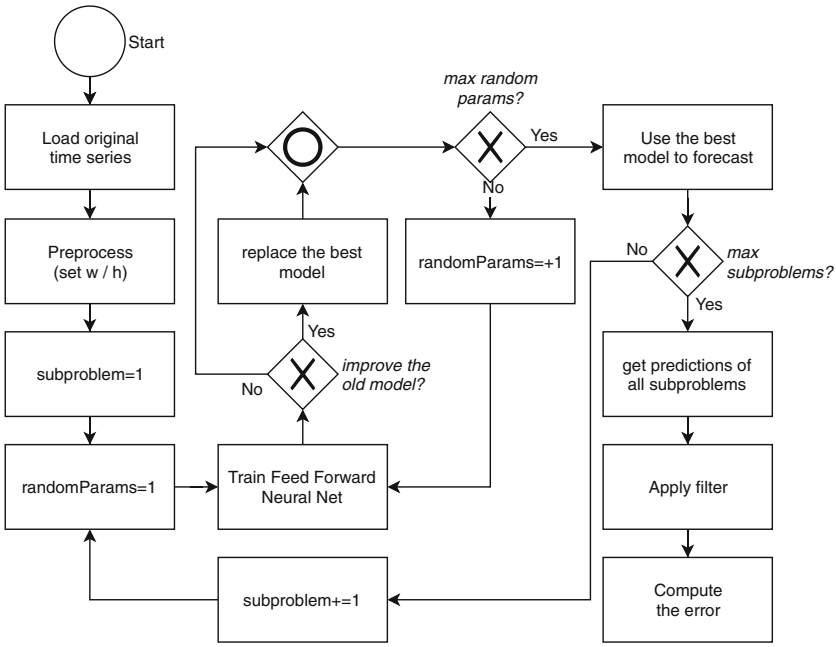


Fig. 1. Complete work-flow of the proposed methodology.

one, a random value is generated for each hyper-parameter and it is validated by computing the forecasting error using a validation set. This process is repeated during a certain number of iterations, storing the model that obtains the smallest error. Finally, a single model is stored for each sub-problem, corresponding to the one whose hyper-parameters offer the best results.

3.2 Multi-step to Single-step Regression

Given a time series expressed as $[x_1, x_2, \ldots, x_t]$, the main goal of this research is to forecast the future values of the time series. To do this, a predictive model is formed based on a historical window composed of w past values that allow the prediction of the h following values, also called the prediction horizon. This kind of problem is known as multi-step forecasting and can be formulated as:

$$[x_{t+1}, x_{t+2}, \ldots, x_{t+h}] = model(x_{t-(w-1)}, \ldots, x_{t-1}, x_t) \tag{1}$$

Regretfully, the deep learning algorithm included in the H2O framework does not support multi-step forecasting. To achieve this goal, a methodology has been developed. This methodology consists in focusing on the prediction of each instant of time individually, dividing the multi-step prediction into h predictions of a single step. This methodology is formulated in Eqs. (2)–(5):

$$x_{t+1} = model_1(x_{t-(w-1)}, \ldots, x_{t-1}, x_t) \tag{2}$$

$$x_{t+2} = model_2(x_{t-(w-1)}, \ldots, x_{t-1}, x_t) \tag{3}$$
$$\ldots \tag{4}$$
$$x_{t+h} = model_h(x_{t-(w-1)}, \ldots, x_{t-1}, x_t) \tag{5}$$

As can be seen from these Equations, there is a gap in the data used in each prediction (e.g. the prediction of x_{t+2} is not used to predict x_{t+3}). However, if these predictions were taken into account to forecast the next point of data, it would cause a propagation of the error, giving rise to a wrong prediction [4].

This formulation involves the training of h different models instead of a single model, requiring a high computational cost. However, the implementation of the deep learning algorithm in H2O is optimized and parallelized, which minimizes this shortcoming.

3.3 Smoothing Filter

Once the hyper-parameters are calculated, the final task can be accomplished. The estimation of individual and independent models to forecast a set of values representing a prediction horizon has a consequence: the predicted values exhibit some significant ripple because the estimated values have no information about neither previous nor subsequent estimations. That is, sharp variations from one value to another may be generated.

For this reason, the application of a smoothing filter is also proposed, as the last step of the methodology. Different strategies can be chosen. For instance, filters based on Fourier transform are quite popular [12] but their quadratic cost function, $O(n^2)$, turn these filters into a not particularly suitable solution in the big data context.

Another much simpler, but powerful, filter has been selected: the one based on moving averages with linear cost function, $O(n)$, and, in particular, the one implemented in the *Stats* R package [13]. This low-pass filter is a common finite impulse response type that removes high frequencies, i.e. the sharp variations. It only needs to adjust the number of previous data that will be used to calculate the average, N.

Mathematically, the calculation of the first filtered value is formulated as follows:

$$x'(t) = \frac{1}{N} \sum_{i=1}^{N} x(t-i) \tag{6}$$

where $x(t)$ is the current smoothed value and x(t − i), for $i = 1$, are the N values preceding $x(t)$. Then, $x(t+i)$, for $i > 0$, are calculated by shifting forward x′(t) but excluding the first number of the time series and including its next value.

To adjust this parameter, N is trained using values from 1 to 12 (as it will explained in Sect. 4, $N = 12$ involves the two previous hours).

4 Results

This section presents the results obtained after applying the methodology described in Sect. 3 to the dataset detailed in Sect. 4.1. All the experiments have been executed into a Intel Core i7-5820K at 3.3 GHz with 15 MB of cache, 12 cores and 16 GB of RAM, working under an Ubuntu 18.04 operating system.

4.1 Dataset Description

The time series considered in this study is related to electrical electricity consumption in Spain, from January 2007 to June 2016. There is a total of 9 years and 6 months with a frequency of 10 min between each measure. This fact makes a time series with a total length of 497832 measures, stored into a 33 MB file in CSV format with a single column. For this reason, a preprocess has been applied to transform the time series into a supervised dataset with $w+h$ columns, where w refers to the historical window of data used to predict the following h values, called the prediction horizon. The whole dataset was split into 70% for the training set and 30% for the test set. In addition, a 30% from the training set has also been selected as the validation set in order to optimize the hyper-parameters of the deep learning algorithm as well as the smoothing filter.

4.2 Error Metrics

To measure the error of the methodology proposed in Sect. 3, the most used metrics in the literature for time series forecasting problems have been used. These metrics are the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Relative Error (MRE). The formulation of these error metrics is shown below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2 \tag{7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2} \tag{8}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - a_i| \tag{9}$$

$$MRE = 100 \cdot \frac{1}{n}\sum_{i=1}^{n}\frac{|p_i - a_i|}{a_i} \tag{10}$$

where n, p and a mean the number of samples, predicted values and actual values, respectively.

4.3 Performance in Terms of Error

The experimental setting of the proposed methodology is as follows:

1. The historical window size used to predict the following four hours (24 values) has been set to 168, which represents a whole day and 4 h. This value has been chosen because the larger the historical window of data, the better results will be obtained, as demonstrated in [15].
2. The hyper-parameters that have been optimized are the number of layers, the number of neurons per layer and the value of learning ratio (rho). The hyper-parameters search ranges have been set to $[1, 5]$, $[10, 100]$ and $[0.9, 0.999]$, respectively.
3. A number of 50 epochs was established in the training phase of the deep learning algorithm. The rest of the deep learning hyper-parameters have default values.
4. To find the optimal hyper-parameters, a total of 50 iterations over each problem was carried out during the training and validation phase.
5. The possible values of N for the smoothing filter have been set between 1 and 12. After the training phase of this parameter, the value has been set to 7.

The configuration of the experiments described above results in a total of 1200 calculated models. The best model for each sub-problem will be used to predict the test set. In order to have a reference point, the results obtained with the proposed methodology have been compared with the methodology proposed by the authors in [15]. This methodology applies an exhaustive search to optimize the size of the historical window, the value of the L1 penalty, distribution function, number of layers and number of neurons, calculating a total of 3120 different models. If only the optimized parameters proposed in this article are taken into account, the grid search calculates 1320 models, 120 more than the methodology proposed in this research.

After completing the training and validation step, 24 different network configurations were obtained, each corresponding to a sub-problem, as detailed in Table 1. It can be seen that the error increases when the timestamp to forecast increases. This fact is due to the time gap between the data to train the model and the timestamp to forecast.

Table 2 summarizes the errors reached by the different approaches. It can be seen how the use of the methodology proposed in this article improves by 20% the mean relative error obtained by the exhaustive search. This is because the exhaustive search only allowed the search for hyper-parameters in a discrete set of values. It is also observed how the application of the smoothed filter significantly improves the error.

A graphical comparison between the real data, non-smoothed predictions and smoothed predictions (described in Sect. 3.3) for an arbitrary day in the test set has been depicted in Fig. 2. It can be seen how the smoothed predictions remove the peaks of the non-smooth predictions, thus significantly decreasing the error.

Table 1. Best hyper-parameters for each subproblem (without smoothing).

SP[1]	Hyper-parameters			Error in test phase			
	#hidden	#neurons	Rho	MSE	RMSE	MAE	MRE (%)
#1	4	[66, 44, 99, 98]	0.971	57099.12	238.95	186.20	0.69
#2	3	[91, 82, 11]	0.922	90365.86	300.61	235.87	0.87
#3	5	[53, 59, 96, 29, 47]	0.961	114441.50	338.29	265.31	0.96
#4	5	[79, 96, 94, 22, 44]	0.937	121272.40	348.24	270.05	0.99
#5	3	[76, 86, 62]	0.971	141457.60	376.11	288.54	1.07
#6	5	[3, 43, 27, 82, 53]	0.928	157920.10	397.39	307.77	1.14
#7	4	[91, 48, 89, 83]	0.988	178831.50	422.88	323.95	1.20
#8	3	[57, 99, 46]	0.981	245192.60	495.17	383.04	1.43
#9	4	[41, 85, 46, 80]	0.970	246930.00	496.92	383.03	1.42
#10	5	[49, 69, 62, 22, 27]	0.917	245124.70	495.10	381.89	1.39
#11	3	[68, 47, 71]	0.927	310147.90	556.91	430.91	1.59
#12	4	[89, 23, 96, 90]	0.966	309112.60	555.98	432.56	1.60
#13	4	[36, 77, 45, 92]	0.961	325379.70	570.42	438.93	1.64
#14	3	[77, 72, 81]	0.969	336707.90	580.27	435.58	1.63
#15	5	[55, 61, 34, 91, 85]	0.941	401978.60	634.02	478.17	1.77
#16	5	[45, 73, 38, 71, 61]	0.963	373900.30	611.47	464.31	1.70
#17	5	[44, 41, 46, 98, 43]	0.978	406642.40	637.69	489.32	1.80
#18	2	[88, 24]	0.966	407873.10	638.65	482.05	1.79
#19	5	[91, 48, 89, 76, 46]	0.907	395915.50	629.22	468.49	1.75
#20	5	[88, 37, 62, 78, 56]	0.928	526235.70	725.42	541.03	2.01
#21	3	[53, 82, 33]	0.962	657200.40	810.68	582.92	2.17
#22	5	[99, 89, 57, 27, 69]	0.986	808235.20	899.02	648.59	2.43
#23	4	[75, 52, 88, 56]	0.997	753634.70	868.12	622.51	2.33
#24	3	[82, 74, 63]	0.941	689790.30	830.54	600.27	2.23

[1] Sub-problem

Figure 3 shows a comparison between actual and predicted data using the models obtained in Table 1. Figure 3(a) shows the prediction of the best day (144 values) for the entire test set. On the other contrary, Fig. 3(b) shows the forecast of the worst day.

Table 2. Comparison of the search metrics and the proposed methodology.

	MSE	RMSE	MAE	MRE (%)
Grid	380486.80	616.84	451.96	1.68
Random	345891.20	588.13	422.55	1.57
Random + Filter	251143.90	501.14	369.19	1.36

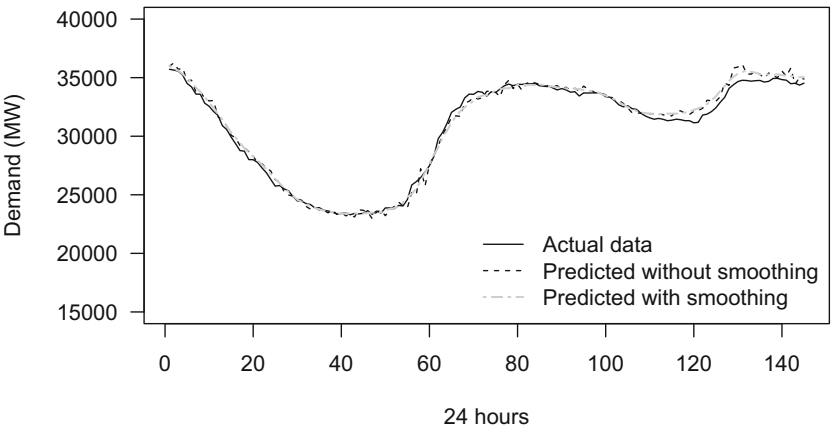
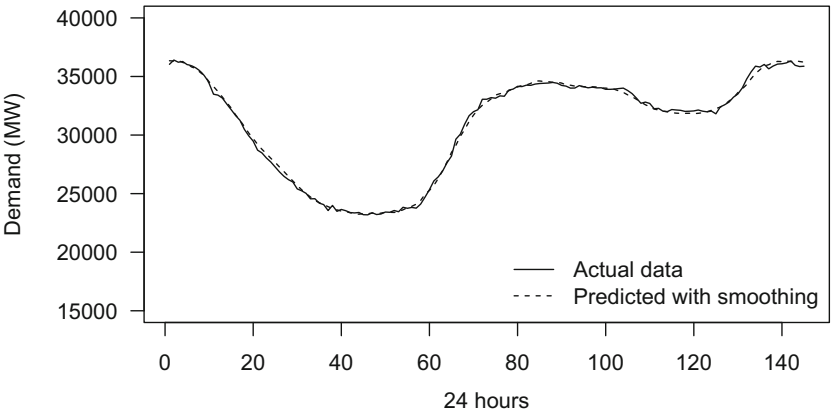
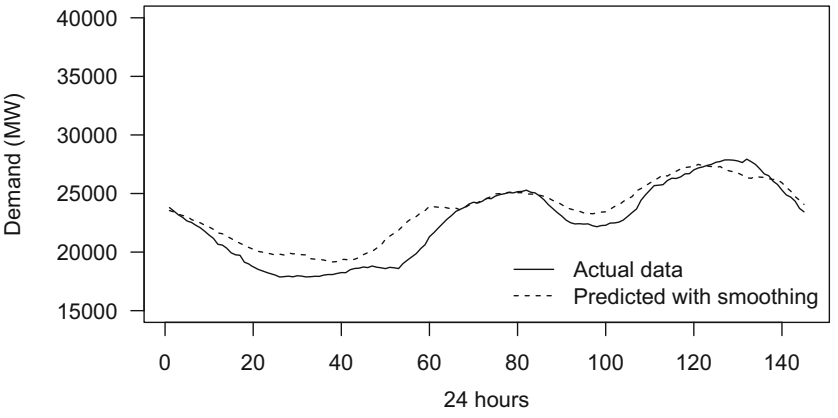


Fig. 2. Comparison between real data, non-smoothed and smoothed predictions.



(a) Best day.



(b) Worst day.

Fig. 3. The best and worst day predicted by the proposed methodology.

5 Conclusions

A method based on deep learning is proposed to forecast big data time series with arbitrary prediction horizon in this work. In particular, a deep feed forward neural network has been used. The tuning of a set of hyper-parameters has been done through a random search approach, as suggested in the literature. Given the nature of the proposed method which estimates different models for every sample included in the prediction horizon, a smoothing procedure based on moving averages is also applied in order to reduce high frequencies in the outputs. The electricity demand forecasting from Spain has been addressed so that the methodology performance can be assessed, reporting two main achievements: acute decrease in the execution time and reduced forecasting error (1.36%).

Acknowledgements. The authors would like to thank the Spanish Ministry of Economy and Competitiveness for the support under the project TIN2017-88209-C2-1-R.

References

1. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS'11, pp. 2546–2554. Curran Associates Inc., New York (2011)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
3. Candel, A., LeDell, E., Parmar, V., Arora, A.: Deep learning with H2O. H2O.ai, Inc., California (2017)
4. Cheng, H., Tan, P.-N., Gao, J., Scripps, J.: Multistep-ahead time series prediction. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 765–774. Springer, Heidelberg (2006). https://doi.org/10.1007/11731139_89
5. Dalto, M., Matusko, J., Vasak, M.: Deep neural networks for ultra-short-term wind forecasting. In: Proceedings of the IEEE International Conference on Industrial Technology (ICIT), pp. 1657–1663 (2015)
6. Diaz, G.I., Fokoue-Nkoutche, A., Nannicini, G., Samulowitz, H.: An effective algorithm for hyperparameter optimization of neural networks. IBM J. Res. Dev. **61**(4/5), 9:1–9:11 (2017)
7. Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 2327–2334 (2015)
8. Ilievski, I., Akhtar, T., Feng, J., Shoemaker, C.A.: Efficient hyperparameter optimization for deep learning algorithms using deterministic RBF surrogates. In: Proceedings of the AAAI Conference on Artificial Intelligence (2017)
9. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast bayesian optimization of machine learning hyperparameters on large datasets. CoRR abs/1605.07079 (2016)
10. Li, X., Peng, L., Hu, Y., Shao, J., Chi, T.: Deep learning architecture for air quality predictions. Environ. Sci. Pollut. Res. Int. **23**, 22408–22417 (2016)

11. Loshchilov, I., Hutter, F.: CMA-ES for hyperparameter optimization of deep neural networks. arXiv preprint arXiv:1604.07269 (2016)
12. Manolakis, D.G., Ingle, V.K.: Applied Digital Signal Processing. Cambridge University Press, Cambridge (2011)
13. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2013). http://www.R-project.org/. ISBN 3-900051-07-0
14. Ruder, S.: An overview of gradient descent optimization algorithms. CoRR abs/1609.04747 (2016)
15. Torres, J., Galicia, A., Troncoso, A., Martínez-Álvarez, F.: A scalable approach based on deep learning for big data time series forecasting. Integr. Comput.-Aid. E. **25**(4), 335–348 (2018)
16. Young, S.R., Rose, D.C., Karnowski, T.P., Lim, S.H., Patton, R.M.: Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, p. 4. ACM, New York (2015)
17. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., et al.: Apache spark: a unified engine for big data processing. Communications of the ACM **59**(11), 56–65 (2016)