PBIL for Optimizing Hyperparameters of Convolutional Neural Networks and STL Decomposition

Roberto A. Vasco-Carofilis[1], Miguel A. Gutiérrez-Naranjo[1], and Miguel Cárdenas-Montes[2](✉)

[1] Department of Computer Science and Artificial Intelligence, University of Seville, Seville, Spain
andresvasc@gmail.com, magutier@us.es

[2] Department of Fundamental Research, Centro de Investigaciones Energéticas Medioambientales y Tecnológicas, Madrid, Spain
miguel.cardenas@ciemat.es

Abstract. The optimization of hyperparameters in Deep Neural Net-works is a critical task for the final performance, but it involves a high amount of subjective decisions based on previous researchers' expertise. This paper presents the implementation of Population-based Incremen-tal Learning for the automatic optimization of hyperparameters in Deep Learning architectures. Namely, the proposed architecture is a combina-tion of preprocessing the time series input with Seasonal Decomposition of Time Series by Loess, a classical method for decomposing time series, and forecasting with Convolutional Neural Networks. In the past, this combination has produced promising results, but penalized by an incre-mental number of parameters. The proposed architecture is applied to the prediction of the ^{222}Rn level at the Canfranc Underground Labora-tory (Spain). By predicting the low-level periods of ^{222}Rn, the potential contamination during the maintenance operations in the experiments hosted in the laboratory could be minimized. In this paper, it is shown that Population-based Incremental Learning can be used for the choice of optimized hyperparameters in Deep Learning architectures with a reasonable computational cost.

Keywords: Hyperparameters optimization · Convolutional Neural Networks · STL decomposition · PBIL · ^{222}Rn measurements · Canfranc Underground Laboratory · Forecasting

1 Introduction

Adjusting machine learning hyperparameters is a tedious but crucial task, as the performance of an algorithm may depend on the choice of hyperparameters. Manual optimization is a time-consuming process that can be invested in other tasks of the algorithm development process. This paper presents an automatic approach for the generation of hyperparameters of a machine learning model, which, in a few training sessions, allows to reach a local minimum among the possible combinations.

The learning model is a combination of Seasonal Decomposition of time series by Loess (STL) and Convolutional Neural Networks (CNN). Both methods have been integrated and applied in the study of time series [18], but their hyperparameters need to be carefully tuned in order to obtain accurate predictions. In order to fit some hyperparameters of CNN and parameters of STL decomposition, population-based incremental learning (PBIL) method has been used.

In our proposal, the arquitecture of the CNN is fixed and PBIL is used in order to optimize the search of some of their hyperparameters and some of the parameters of STL for a given task. In such way, a concrete dataset is considered and the task is to find a combination of parameters that minimizes the loss of the CNN on the dataset. PBIL is an optimization method of the family of genetic algorithms and, therefore, in order to use PBIL, each set of parameters of the CNN is encoded as a binary sequence, in other words as an *individual* of the population. In order to obtain the *fitness* of an individual which guides the evolutinary procees, firstly the hyperparameters encoded by the individual are obtained. The fitness associated to the individual is the loss of a single run of the CNN on the dataset when these parameters are considered.

The problem of automatically tuning hyperparameters in machine learning has been considered from many different point of views. One of the simplest approaches is the random search, which was explored in [3]. A different point of view is based in Bayesian optimization. This method considers on an iterative evaluation and update of promising hypermarameters [22]. Gradient-based optimization [16], radial basis functions [8] and spectral methods [13] have also been explored.

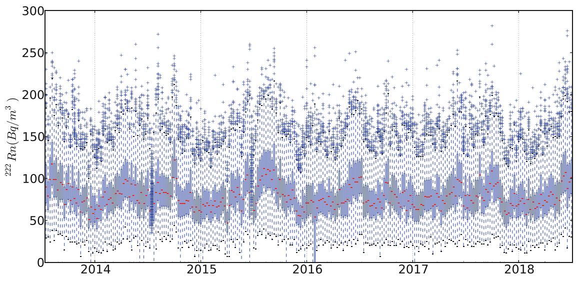
Special mention deserves the optimization of hyperparameters inspired in evolutionary and population based methods. Evolutionary optimization considers ideas coming from evolutionary algorithms to explore a hyperparameters space [19]. In population based methods applied to neural networks, the model tries to optimize network weights and hyperparameters simultaneously [15]. In our approach, an optimization method which takes ideas from competitive learning and population based methods is also considered.

1.1 Previous Efforts

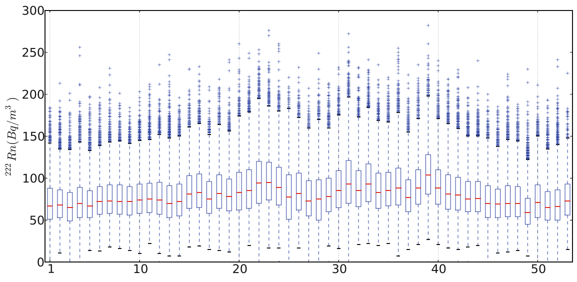
In the past, the ^{222}Rn time series at Canfranc Underground Laboratory (LSC) has been analysed and forecast using non-stochastic algorithms: Holt-Winters, AutoRegressive Integrated Moving Averages, and Seasonal and STL (see [17] and references therein). Also in [17] and [18], the forecasting capacity of Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks is evaluated for this problem. In [5], improvements in the forecasting capacity is reported when implementing an Ensemble Deep Learning approach. In this study the ensemble is composed of Bidirectional Recurrent Neural Networks (BRNN), CNN, and a variant of CNN, termed CNN+STL, in which the original observations used as input are replaced by the components generated by Seasonal and Trend decomposition using Loess (STL): trend, seasonal and remainder components [18].

In [18] the first implementation of using STL decomposition and CNN for improving the forecasting capacity was presented. The promising results obtained only for some test was penalized by the lack of an optimal configuration. In this work, the analysis of the ^{222}Rn time series at LSC was based on the monthly medians of 5 years, 60 observations. The reduced dataset also penalized the final performance[1]. The sub-optimal performance achieved at the same time that the promising results motivates the additional effort presented in the current work.

The LSC is composed of two main experimental halls for hosting scientific experiments with requirements of very low-background. The ^{222}Rn concentration is monitored every 10 minutes in both halls. An accumulated record from July 2013 to September 2018 for Hall A is available (Fig. 1). Due to the high level of noise of the measurements, the weekly medians are considered for composing



(a)



(b)

Fig. 1. Weekly box-plots of ^{222}Rn level at Hall A of the LSC, by week (Fig. 1(a)) and gathering the weeks independently of the week of the year (Fig. 1(b)). Depicted data corresponds to the period from July 2013 to June 2018.

[1] In the current work, the dataset is composed of weekly means of 5 years, which increments the number of observations up to 259.

the time series (red lines in the boxplot centre in Fig. 1(a)). Finally, the dataset contains 259 observations corresponding to the weekly medians. The faint annual modulation observed in Fig. 1(b) embodies in the noisy observations indicates that ^{222}Rn concentration at LSC is not fully random, and therefore it can be treated by machine learning algorithms.

The paper is organized as follows: Sect. 2 gives a brief description of the different Artificial Intelligence techniques used in this paper: PBIL as optimizer, STL for decomposing time series into components and CNN as predictive model. Section 3 describes the proposed approach. In Sect. 4, our application to the case study is shown, and finally, Sect. 5 contains the conclusions of this work.

2 Methods

2.1 Population-Based Incremental Learning

Population-based incremental learning is an optimization method which combines genetic algorithms with competitive learning [1,2]. It belongs to the so-called estimation of distribution algorithms (EDAs). The main difference with standard evolutionary algorithms is that EDAs do not create a population of solutions from the previous generation by using crossover and mutations. EDAs consider global information of the whole population in order to build a probabilistic distribution which is updated step by step. The optimization method is based on local search and it has been proved that it converges to local optima [21]. In this paper, binary-valued encoding for genotypes have been used. In our approach, PBIL is configured for using the best 5 individuals to build the distribution.

During the execution of the algorithm, individuals, representing the hyper-parameters and parameters being optimized, are created from a probability distribution. Through the generations, only the best individuals of a generation are used for updating the parameters of the probability distribution. Later, based on the updated probability distribution parameters, a new generation of individuals is created by sampling the probability distribution and evaluated, restarting the process. The initial value of each probability is set at 0.5. In our approach, the individuals are created as binary vectors. Mean Squared Error (MSE, $MSE = \sum_i (\hat{y}_i - y_i)^2$, where y_i are the observations and $\hat{y}_i$ the predictions) of a single run of the CNN is used as fitness function in PBIL algorithm.

2.2 Gray Coding

Gray coding is a type of binary conding which is usually used in order to avoid Hamming cliffs (Table 1). A Hamming cliff is formed when two numerically adjacent values have bit representations that are far apart by using the Hamming distance. For example, number 3 and 4 differ in binary representation in three bits: 0011 and 0100, having a Hamming distance of 3; or for 15 and 16, corresponding the binary representations 01111 and 10000, which have a Hamming distance of 5.

Table 1. Example of correspondence among decimal, binary and Gray coding.

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110

A large Hamming distance is a barrier for the evolution of the individuals in the evolutionary algorithm. The change of one unit in a parameter under optimization requires a large amount of simultaneous modification of the individual binary coding, but not in Gray coding. This improbable process degrades the performance of evolutionary algorithm.

In our approach, the optimization of the hyperparameters will be performed by PBIL and each set of parameters will be encoded as a binary sequence by using Gray coding. For each hyperparameter to optimize, it is necessary define the minimum and maximum feasible values, and the step for moving in this range. This determines the size of the individual, since the binary representation must be able to express all the feasible values and hence, also the length of the vector of probabilities. For each parameter x, the number of bits n_x used in the individual for representing it can be calculated as $n_x = \lceil log_2((max - min)/step) \rceil$. For example, if the chosen region for searching a parameter p for the CNN has the bounds 20 and 64 and the chosen step is 4, then the number of used bits is $4 = \lceil log_2(11) \rceil$. If the 4 bits in a concrete individual are 1100, the decodification c (by using Gray conding) of such sequence is 8 and hence the concrete value of the parameter p can be obtained as $p = min + (c * step)$. In our case $p = (8 * 4) + 20 = 52$.

Each individual of the evolutionary algorithm is a binary vector concatenating the hyperparameters and parameters being optimized. For each generation, decimal representation of these hyperparameters and parameters are Gray-coded and concatenated. Later, when they have been manipulated by the evolutionary algorithm, they are decoded and evaluated with the CNN+STL implementation.

The CNN hyperparameters optimized are:

- *Batch size* is number of samples per gradient update, within the range [16] with step 1.
- *Sample size* is the number of observations which are used as independent variable. ranging from 52 to 100 with step 2.
- The number of *epochs* for training, ranging from 20 to 140 with a step of 20.
- Size of the kernels in the convolutional blocks: ranging from 3 to 13 for the first block, from 3 to 9 for the second one, and from 3 to 5 for the third one, with a step of 1.

– The number of kernels in the convolutional blocks: ranging from 8 to 32 for the first block, from 16 to 64 for the second one, and from 32 to 128 for third one, with steps for 8, 8 and 16 respectively.

The STL decomposition parameters optimized are:

– The most significant period of the time series for STL decomposition, *period*, within the range [30, 61] with step 1.
– The fraction of data used in fitting lowess regression, *lo_faction*, within the range [0, 1] with step 0.1. The parameters *lo_faction* and *lo_ delta* are converted into integer before the Gray coding.
– The distance within which to use linear-interpolation instead of weighted regression, *lo_ delta*, within the range [0, 0.2] with step 0.01.

2.3 Seasonal Decomposition of Time Series by Loess

Seasonal decomposition of time series by loess (STL) is a method of decomposing time series [7]. The whole time series is decomposed into three components. One of the components is the trend (T_t), related to the long-term increase or decrease of the original time series. Another component is seasonal component (S_t), related to the periodicity of the data. Finally, the third component is the remainder or random component (R_t). In the case of additive decomposition, the sum of these components results in the original time series (Y_t), $Y_t = T_t + S_t + R_t$. In our approach, additive STL decomposition is employed using the STLDecompose library [20].

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are specialized Neural Networks with special emphasis in image processing [12,14], although nowadays they are also employed in time series analysis and forecasting [9,17,18,23].

The CNN consists of a sequence of convolutional layers, the output of which is connected only to local regions in the input. These layers alternate convolutional, non-linear and pooling-based layers which allow extracting the relevant features of the class of objects, independently of their placement in the data example. The CNN allows the model to learn filters that are able to recognize specific patterns in the time series, and therefore they can capture richer information from the series. It also embodies three features which provide advantages over the multilayer perceptron: sparse interactions, parameter sharing and equivariance to translation [12].

Although CNN are frequently associated to image or audio classification -2D grid examples- or video sequence -3D grid examples-, it can also be applied to time series analysis -1D grid examples-. When processing time series, instead of a set of images, the series has to be divided in overlapping contiguous time windows. These windows constitute the examples, where the CNN aims at finding patterns. At the same time, the application to time series modelling requires the application of 1D convolutional operators, whose weights are optimized during the training process.

CNN architecture is composed of three branches handling the three components arisen from the STL decomposition. Each branch is composed of three convolutional blocks with `relu`—Rectified Linear Unit—as activation function. Ending this, the three intermediated representations are concatenated and then they pass through two dense layers of 64, and 16 neurons and `relu` as activation function; and a output layer with `linear` activation function. Among other hyperparameters, the number of filters in the convolutional blocks and their size are optimized through the PBIL algorithms. The loss function is the MSE and the optimizer is `Adam`.

For the implementation of the current approach, Python3 and Keras library have been used for the implementation of the CNN [6].

2.5 Statistics

In order to ascertain if the proposed forecasting methods applied to the test set improve the prediction, two different types of tests can be applied: parametric and non-parametric. The difference between both relies on the assumption that data are normally distributed for parametric tests, whereas non explicit conditions are assumed in non-parametric tests. For this reason, the latter is recommended when the statistical model of data is unknown [10,11].

The Kruskal-Wallis test is a non-parametric test used to compare three or more groups of sample data. For this test, the null hypothesis assumes that the samples are from identical populations. The procedure when using multiple comparison to test whether the null hypothesis is rejected implies the use of a post-hoc test to determine which sample makes the difference. The most typical post-hoc test is the Wilcoxon signed-rank test.

The Wilcoxon signed-rank test belongs to the non-parametric category. For this test, the null hypothesis assumes that the samples are from identical populations, whereas the alternative hypothesis states that the samples come from different populations. It is a pairwise test that aims to detect significant differences between two sample means.

3 PBIL for Optimizing Hyperparameters of Convolutional Neural Networks and Parameters of STL Decomposition

The algorithm used in this work is outlined in the flowchart of Fig. 2.

1. Initially, with a probability 0.5, v_i Gray-coded individuals are randomly created, each one with length n, being n the minimum number of bits for representing all the hyperparameters and parameters to optimize. In the tests, population is composed of 25 individuals.

2. Each individual is split for obtaining the hyparameters and parameters, and the parts are translated from Gray-coded to decimal-coded representation.
3. The individuals are evaluated through the CNN+STL architecture (Fig. 3). The test error on a single run of the implementation is used as individual fitness.
4. The best and worse individuals are identified for creating a new probability vector. Each component of the probability vector follows a binomial distribution. Best individuals modify p of previous generation so that it is more probable for the individuals in the next generation to obtain positive features, whereas the worse individuals act in opposite direction over p.
5. The process is repeated until the end criterion —number of generations—is achieved.

4 Experimental Results and Analysis

For evaluating the overall performance of the proposed approach, performance comparisons with Multiplayer Perceptron, Convolutional Neural Network, and—Long Short-Term Memory—Bidirection Recurrent Neural Network are made. In those comparisons, MSE is employed as figure of merit. In Table 2, the values of the MSE after 15 independent executions are presented. As it can be appreciated, the proposed approach, CNN+STL, produces the lowest error among the architectures evaluated. CNN+STL clearly outperforms Multilayer Perceptron (MLP), and previous efforts in CNN; and it is competitive with BRNN which has as excellent performance in time series forecasting for a diversity of observations, from Radon [5] to air pollutants concentration [4].

The application of the Kruskal-Wallis test to the MSE indicates that the differences between the medians are significant for a confidence level of 95% (p-value under 0.05), p-value $= 5 \cdot 10^{-9}$, which means that the differences are unlikely to have occurred by chance with a probability of 95%. Furthermore, the application of the Wilcoxon signed-rank test to the values of the MSE of CNN+STL and BRNN approaches points that the differences between the median of the MSE are not significant for a confidence level of 95% (p-value under 0.05), p-value = 0.08.

In Fig. 4, the evolution of the MSE for the PBIL population versus the generation is shown. In this case, the final MSE is 77.83 with an initial MSE of 90.86. This initial MSE is much lower that the final MSE of other runs. This could indicate that the initial search space is too wide, and the initial random population plays a critical role in the performance of the run, even if PBIL has the capability to produce improvements in all the runs.

In-detail insight on the final values of CNN hyperparameters and STL decomposition parameters, for both excellent-performance and poor-performance runs, offers a valuable information about outperforming configurations. For instance, performance critically degrades when the period of decomposition is out of the narrow range [50, 52], while the two other parameters of STL decomposition do not play a relevant role in the final performance.

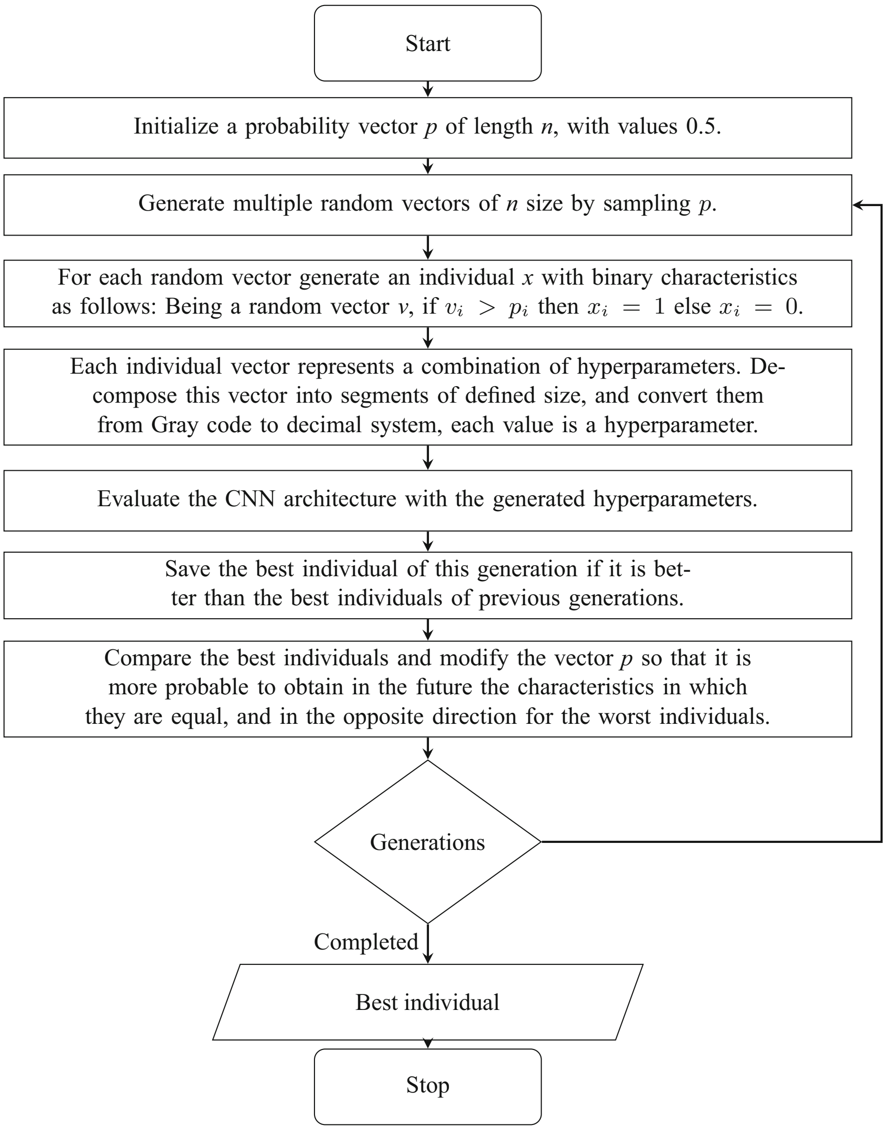

Start
Initialize a probability vector p of length n, with values 0.5.
Generate multiple random vectors of n size by sampling p.
For each random vector generate an individual x with binary characteristics as follows: Being a random vector v, if $v_i > p_i$ then $x_i = 1$ else $x_i = 0$.
Each individual vector represents a combination of hyperparameters. Decompose this vector into segments of defined size, and convert them from Gray code to decimal system, each value is a hyperparameter.
Evaluate the CNN architecture with the generated hyperparameters.
Save the best individual of this generation if it is better than the best individuals of previous generations.
Compare the best individuals and modify the vector p so that it is more probable to obtain in the future the characteristics in which they are equal, and in the opposite direction for the worst individuals.
Generations
Completed
Best individual
Stop

Fig. 2. Algorithm proposed for the implementation of PBIL in the optimization of hyperparameters of CNN and parameters of STL, in which given a specific number of generations the algorithm is able to obtain a combination of hyperparameters and parameters that approach the global optimum, improving the results with each generation.

With regards to the hyperparameters, the sample size seems not be as critical as the period of the STL decomposition for the performance of the proposed approach. A wide range of best MSE is achieved within the range of this hyper-

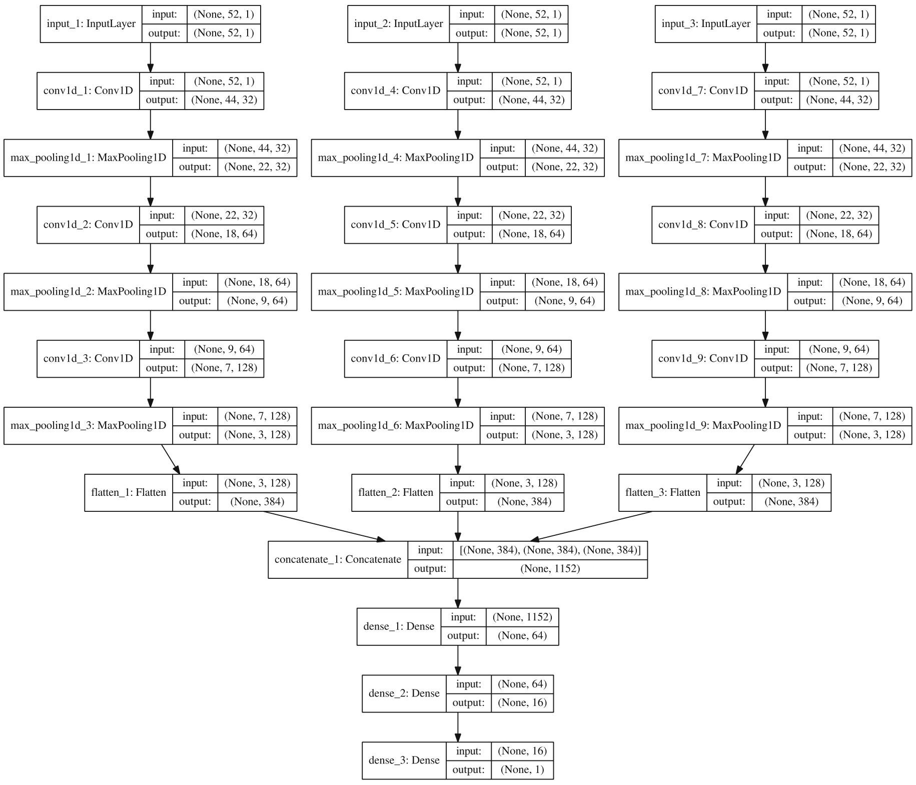


Fig. 3. CNN architecture. It is composed of three branches for handling the three components arising from the STL decomposition.

Table 2. Mean and standard deviation of MSE of test set (30% of the dataset) for 15 independent runs.

Architecture	MSE
MLP	113 ± 10
CNN	199 ± 22
BRNN	101 ± 2
CNN+STL	97 ± 9

parameter [52, 100]. A similar behaviour is observed for the number of epochs and the batch size.

The CNN architecture of the best case is composed of the following number of kernels: 13, 48, and 32; with sizes: 13, 4, and 5, with a sample size of 58. Two other high-performance cases appear for number of kernels (8, 40, 32), with kernels sizes (8, 9, 5) and (12, 3, 5), and sample sizes 96 and 100, respectively. Oppositely, low-performance cases appear when the number of kernels are much

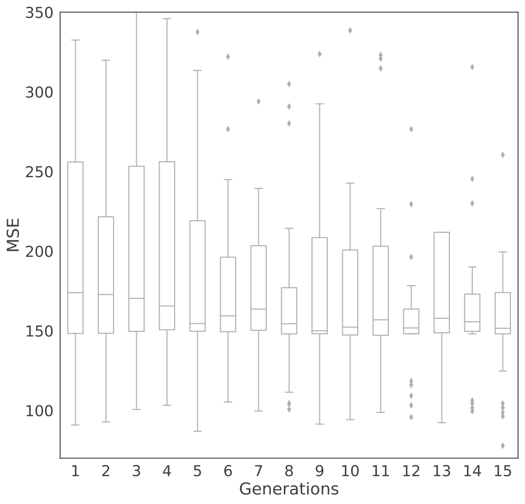


Fig. 4. Boxplot with the evolution of MSE per generation for the best case of 15 independent runs.

larger, specially in the last convolutional blocks, for instance (16, 48, 96), (32, 64, 80) and (24, 64, 128) with samples sizes: 70, 54, and 58, respectively.

5 Conclusions

In this paper, an approach based on PBIL for optimizing the hyperparameters of a CNN architecture and the parameters of a STL decomposition applied to the time series forecasting of ^{222}Rn concentration at Canfranc Underground Laboratory has been proposed. In the previous efforts, the performance of the time series forecasting with CNN architecture and STL decomposition was slightly penalized by the lack of a optimized parameters and hyperparameters set. The promising results obtained in the past deserve of an appropriated exploration of the parameter space. For this purpose, PBIL has been applied for finding a high-quality sub-optimal parameters and hyperparameters set. The use of an evolutionary algorithm for optimizing the hyperparameters of CNN allows replacing human-expertise-based values by optimized values.

The results and the statistical analysis state that the proposed architecture improves the previous performance of CNN and MLP algorithms being competitive with other well-recognized as suitable for time series forecasting as LSTM-Bidirectional Recurrent Neural Networks. Furthermore, the implementation can be used for exploring and evaluating not classical configuration of CNN architecture for time series forecasting.

To evaluate the proposed approach in other data sets is proposed as Future Work.

Acknowledgment. MCM is funded by the Spanish Ministry of Economy and Competitiveness (MINECO) for funding support through the grant "Unidad de Excelencia María de Maeztu": CIEMAT - FÍSICA DE PARTÍCULAS through the grant MDM-2015-0509.

References

1. Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical report, CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, January 1994
2. Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, 9–12 July 1995, pp. 38–46 (1995)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
4. Cárdenas-Montes, M.: Forecast daily air-pollution time series with deep learning. In: Pérez García, H., Sánchez González, L., Castejón Limas, M., Quintián Pardo, H., Corchado Rodríguez, E. (eds.) HAIS 2019. LNCS (LNAI), vol. 11734, pp. 431–443. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29859-3_37
5. Cárdenas-Montes, M., Méndez-Jiménez, I.: Ensemble deep learning for forecasting ^{222}rn radiation level at canfranc underground laboratory. In: Martínez Álvarez, F., Troncoso Lora, A., Sáez Muñoz, J.A., Quintián, H., Corchado, E. (eds.) SOCO 2019. AISC, vol. 950, pp. 157–167. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-20055-8_15
6. Chollet, F., et al.: Keras (2015). https://github.com/fchollet/keras
7. Cleveland, R.B., Cleveland, W.S., McRae, J., Terpenning, I.: STL: a seasonal-trend decomposition procedure based on loess. J. Official Stat. 3–73 (1990)
8. 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), 9 (2017)
9. Gamboa, J.C.B.: Deep learning for time-series analysis. CoRR abs/1701.01887 (2017). http://arxiv.org/abs/1701.01887
10. García, S., Fernández, A., Luengo, J., Herrera, F.: A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput. **13**(10), 959–977 (2009)
11. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J. Heuristics **15**(6), 617–644 (2009)
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
13. Hazan, E., Klivans, A.R., Yuan, Y.: Hyperparameter optimization: a spectral approach. CoRR abs/1706.00764 (2017). http://arxiv.org/abs/1706.00764
14. LeCun, Y.: Generalization and network design strategies. University of Toronto, Technical report (1989)
15. Li, A., et al.: A generalized framework for population based training. CoRR abs/1902.01894 (2019)

16. Maclaurin, D., Duvenaud, D.K., Adams, R.P.: Gradient-based hyperparameter optimization through reversible learning. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, pp. 2113–2122 (2015)
17. Méndez-Jiménez, I., Cárdenas-Montes, M.: Modelling and forecasting of the ^{222}Rn radiation level time series at the Canfranc Underground Laboratory. In: de Cos Juez, F., et al. (eds.) Hybrid Artificial Intelligent Systems - 13th International Conference, HAIS 2018, Proceedings. Lecture Notes in Computer Science, vol. 10870, pp. 158–170. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-319-92639-1_14
18. Méndez-Jiménez, I., Cárdenas-Montes, M.: Time series decomposition for improving the forecasting performance of convolutional neural networks. In: Herrera, F., et al. (eds.) Time series decomposition for improving the forecasting performance of convolutional neural networks. LNCS (LNAI), vol. 11160, pp. 87–97. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00374-6_9
19. Miikkulainen, R., et al.: Evolving deep neural networks. CoRR abs/1703.00548 (2017)
20. Montague, J.: STLDecompose (2017). https://github.com/jrmontag/STLDecompose
21. Rastegar, R., Hariri, A.: The population-based incremental learning algorithm converges to local optima. Neurocomputing **69**(13–15), 1772–1775 (2006)
22. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held 3–6 December 2012, Lake Tahoe, Nevada, United States, pp. 2960–2968 (2012). http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms
23. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: a strong baseline. CoRR abs/1611.06455 (2016). http://arxiv.org/abs/1611.06455