Extreme Learning Machine Based Model Improved with Adaptive Activation Functions

Chapter · January 2021

DOI: 10.1007/978-3-030-68133-3_12

2 authors, including:

Leonardo Goliatt
Federal University of Juiz de Fora
115 PUBLICATIONS **436** CITATIONS

SEE PROFILE

Extreme Learning Machine Based Model Improved with Adaptive Activation Functions

T. L. Fonseca[1][0000−0002−5923−1933] and L. Goliatt[1][0000−0002−2844−9470]

Computational Modeling Program, Federal University of Juiz de Fora, Juiz de Fora, Brazil
tales.lima@ice.ufjf.br, leonardo.goliatt@ufjf.br

Abstract. This paper proposes an approach combining an Extreme Learning Machine based model with adaptive activation functions for regression problems. Research related to machine learning seeks to solve a given problem in the best possible way, generating increasingly accurate generalist predictive models. On the other hand, these models become more complex, requiring massive computational power to train and make inferences. The use of adaptive activation functions aims at increasing the models' predictive capacity allowing generating better models without increasing their complexity. We evaluated the models' performance in a benchmark of ten regression problems. The results showed that Extreme Learning Machine with adaptive activation functions, in general, outperforms standard functions with the same architecture.

Keywords: Adaptive Activation Functions · Extreme Learning Machine · Regression Problems.

1 Introduction

In the beginning of the research on Artificial Neural Network (ANN) the works commonly used the activation functions such as Binary, Linear, Sigmoid and Hyperbolic Tangents [9]. These functions, separately or in combination through different layers, were and still are capable of solving different machine learning problems with satisfactory quality. Besides, the use of the appropriate activation function makes a total difference in the process of creating and learning an ANN [1], and even studies on which activation function or set of functions are best suited to the problem [16, 22].

However, many ANN's may require a large number of layers and neurons to perform the function/problem approximation, which leads to increasingly larger and computationally slower models, both in their training and prediction stages [3]. Besides, these models end up impacting, during their training and prediction, the environment [25, 26], in addition to being unfeasible in straight edge computing. Thus, in the opposite direction to increase the number of neurons and the number of layers, work has emerged related to neurons' learning power.

In order to bring greater predictive power to an artificial neural network, some studies have started to study activation functions with the ability to adapt its shape during the training stage [19]. The first work to explore this property sought to adapt a polynomial activation function, obtaining better results than traditional functions [21]. After this study, new works related to the adaptive activation function emerged as Hyperbolic Tangent [4], Splines [2, 30], PReLU [10], and Soft Exponential [7]. All of these works proved, through numerical experiments, to overcome functions that cannot adapt and presented a fast convergence.

With the confirmation of the gains of using adaptive activation functions, different applied studies have appeared, such as classification of electrocardiographic arrhythmias [28], inference of gene expression [18], classification of images generated by computed tomography [32] and approximation of smooth and discontinuous functions, as well as solutions of differential equations linear and non-linear partials [15].

All adjustable parameters of an ANN can be learned through an optimizer. The most common, based on the descending gradient, is Backpropagation [23]. Despite being the most common, it can present a slower convergence, needing in some cases to study other ways of finding the optimal parameters, as L-BFGS [31] and Evolution Strategies [27]. In another direction, to obtain useful models with fast learning, Extreme Learning Machines (ELM) networks emerged [12]. A combination of optimization methods is also common, such as the combination of Backpropagation with ELM [33], finding better parameters than each one individually.

This paper proposes a study of Adaptive Activation Functions in an ELM based model, trained the output layer using Back Propagation, for a set of 10 benchmark datasets. The main positive contributions of the study can be summarized as follows: (1) in the best the authors' knowledge, this is the first contribution implementing ELM with Adaptive Activation Functions; (2) this paper highlights two ways to use Adaptive Activation Functions, Adaptive Layer and Adaptive Neuron.

The results show that the adaptive approach of the functions, combined with ELM, produces results superior to those of a standard function with the same architecture. This work has the main contribution in using ELM with adaptive activation functions, something that has not been exploited until the present moment, according to the authors' knowledge.

2 Material and Methods

2.1 Datasets

In this work, to compare the proposed techniques, ten datasets related to regression problems were used. We obtained the datasets through PMLB [20], a broad benchmark suite for machine learning evaluation and comparison. We randomly extracted 20% of the complete database for testing. Table 1 presents

a summary of the databases used in the numerical experiments, containing their name, quantity used in training and testing and, the number of features.

Table 1. Summarization of the databases used in numerical experiments.

Dataset	Problem	Train Instances	Test Instances	Features
1	1027_ESL	390	98	4
2	1028_SWD	800	200	10
3	1029_LEV	800	200	4
4	1030_ERA	800	200	4
5	1089_USCrime	37	10	13
6	1096_FacultySalaries	40	10	4
7	294_satellite_image	5148	1287	36
8	4544_GeographicalOriginalofMusic	847	212	117
9	503_wind	5259	1315	14
10	573_cpu_act	6553	1639	21

2.2 Extreme Learning Machine (ELM)

Extreme Learning Machine (ELM) [13] is a particular case of feedforward artificial neural network, where the vast majority is composed of only one hidden layer. Compared with other types of prediction models, including ANN, the ELM has advantages if it is fast learning, good generalization ability, and convenience in terms of modeling [8].

In ELM, all hidden parameters are assigned randomly, including those of the activation functions, and the output parameters are obtained using the generalized inverse of the output matrix of the last hidden layer [14]. The ELM output function, with only one hidden layer, used in this article is given by [24]:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{w}_i, b_i, \mathbf{x}) \tag{1}$$

where $\hat{y}$ is the ELM prediction associated to the input vector $\mathbf{x}$, $\mathbf{w}_i$ is the weight vector that multiplies the input vector $\mathbf{x}$ to generate the input value of the ith-neuron in the hidden layer, b_i is the bias of the ith-neuron in the hidden layer, β_i are output weights that multiply the output of the ith-neuron in the hidden layer, $G(\cdot)$ is the nonlinear activation function, and L is the number of neurons in the hidden layer. The parameters $(\mathbf{w}, b)$ are randomly generated (normally distributed with zero mean and standard deviation equals to one), and weights β_i of the output layer are determined analytically.

In a traditional ELM, the output weight vector $[\beta_1, ..., \beta_L]$ can be determined by minimizing [11]

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^L} (\|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\| + C||\boldsymbol{\beta}||^2) \tag{2}$$

where $\mathbf{y}$ is the output data vector, $\mathbf{H}$ is the hidden layer output matrix

$$\mathbf{H} = \begin{bmatrix} G_1(\mathbf{w}_1, b_1, \mathbf{x}_1) & \cdots & G_L(\mathbf{w}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G_1(\mathbf{w}_1, b_1, \mathbf{x}_N) & \cdots & G_L(\mathbf{w}_L, b_L, \mathbf{x}_N) \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

is the output data vector with N the number of data points. The optimal solution is given by

$$\boldsymbol{\beta} = (\mathbf{H^T H})^{-1} \mathbf{H^T y} = \mathbf{H}^{\dagger} \mathbf{y}$$

where $\mathbf{H}^{\dagger}$ is the pseudoinverse of $\mathbf{H}$.

In this study, to adapt the parameters present in the activation functions, we modified the process of obtaining the parameters $\boldsymbol{\beta}$. Instead of calculating the pseudoinverse of $\mathbf{H}$, we use the traditional Back Propagation. This proposal generates a hybrid between ANN and ELM, where the hidden parameters are still fixed and randomly generated, and the final parameters are obtained through the Back Propagation.

2.3 Adaptive Activation Functions

In a traditional artificial neural network, we have the standard activation functions, which do not change their shape during the training phase. Despite this, we have the weights and biases that adapt during training to minimize/maximize a specific performance metric. This process of adapting the parameters allows the models, whether neural networks or not, to learn an approximate function of the training data, thus allowing the model to generalize to new data.

Adaptive activation functions emerged to increase the learning capacity of a neural network, allowing the learning of some parameters, such as weights and biases, during the training phase. In this way, an activation function can adapt to shapes that make it possible to find better minima/maximums of the objective function than standard activation functions.

In this work we study the following four activation functions and their adaptive forms:

$$\text{PReLU}(x, \alpha = 10^{-2}) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{otherwise} \end{cases}$$

$$\text{Quadratic}(x, \alpha = 0.1, \beta = 10^{-6}, \gamma = 10^{-6}) = \alpha x^2 + \beta x + \gamma$$

$$\text{Sigmoid}(x, \alpha = 1, \beta = 1) = \frac{\alpha}{1 + exp(-\beta x)}$$

$$\text{Swish}(x, \alpha = 1, \beta = 1) = \frac{\alpha x}{1 + exp(-\beta x)}$$

We fix the α, β, and γ values for the standard activation function at the values shown above. For adaptive cases, these values are the initial values and may converge to different ones during the training stage.

We propose two ways to build adaptive architecture, one called Adaptive Layer and the other one, Adaptive Neuron. In the Adaptive Layer, neurons in the same layer shared all the activation functions' adaptive parameters. On the other hand, in the Adaptive Neuron, each internal neuron of a layer has its adaptive parameters, without sharing it between them.

Using the Adaptive Layer strategy, the number of parameters to be adapted is less than or equal when compared with the Adaptive Neuronio strategy. There is a trade-off between the number of parameters and complexity to find all expected values.

3 Computational Experiments and Discussion

The same number of neurons, to each dataset, was used to create the architectures in order to verify the capacity of the adaptive activation functions. This decision ensures that when comparing the adaptive types of functions, the architecture is the same for each problem in terms of numbers of neurons, varying only the function parameters' existence. Furthermore, we created ELM with only one hidden layer, containing several neurons equal to the floor of the number of features divided by two.

For each database, each activation function and each adaptive type (Normal, Layer and Neuron), we trained the network using Adam optimizer [17] for 500 epochs with a batch size of 32 and a learning rate of 10^{-4} that was adjusted every ten epochs using a multiplicative factor of 0.99. Furthermore, we carry out 30 independent training and evaluation runs, calculating for each of them the Root Mean Squared Error (RMSE). Table 2 presents a summary of the results, showing the mean and standard deviation of the RMSE metric of the independent executions. It is possible to observe that, in general, the adaptive functions have a lower mean than the standard function, indicating that they can better minimize the RMSE. On the other hand, it is possible to observe that the results are similar among the adaptive functions, perhaps indicating the architecture's limit.

In order to provide a deeper comparative analysis of the results, we performed a One-way Analysis of variance (ANOVA) [6] with a confidence interval of 95%. The null hypothesis that the means between each adaptive type are statistically significantly different from each other. The results showed that, in some cases, the test rejected the null hypothesis. With a 95% confidence interval, we used the Tukey test [29] to check where the difference between the means existed. Figure 1 shows a count, with a maximum value equal to the number of databases, of which adaptive types of function showed a significant mean difference according to the Tukey test.

The Adaptive Layer and the Adaptive Neuron, of three of the four functions, presented only one problem in which they had a significant difference, which

Table 2. Mean and standard deviation of the RMSE metric for the different types of function and adaptive types. Values in bold represent the best result for the metric evaluated.

Dataset	Activation Function	Normal	Layer	Neuron
1	PReLU	1.225 ± 0.154	1.137 ± 0.174	1.137 ± 0.174
	Quadratic	1.281 ± 0.070	$\mathbf{1.020} \pm 0.200$	$\mathbf{1.020} \pm 0.200$
	Sigmoid	1.251 ± 0.104	1.181 ± 0.147	1.181 ± 0.147
	Swish	1.217 ± 0.155	1.129 ± 0.175	1.129 ± 0.175
2	PReLU	0.773 ± 0.029	0.764 ± 0.037	0.763 ± 0.037
	Quadratic	0.805 ± 0.005	0.757 ± 0.038	$\mathbf{0.752} \pm 0.039$
	Sigmoid	0.796 ± 0.018	0.786 ± 0.033	0.781 ± 0.033
	Swish	0.765 ± 0.031	0.759 ± 0.035	0.760 ± 0.033
3	PReLU	0.920 ± 0.042	0.908 ± 0.059	0.908 ± 0.059
	Quadratic	0.937 ± 0.008	$\mathbf{0.896} \pm 0.067$	$\mathbf{0.896} \pm 0.067$
	Sigmoid	0.935 ± 0.026	0.923 ± 0.043	0.923 ± 0.043
	Swish	0.914 ± 0.047	0.907 ± 0.060	0.907 ± 0.060
4	PReLU	2.014 ± 0.053	1.991 ± 0.086	1.991 ± 0.086
	Quadratic	2.039 ± 0.018	$\mathbf{1.953} \pm 0.110$	$\mathbf{1.953} \pm 0.110$
	Sigmoid	2.030 ± 0.042	2.014 ± 0.064	2.014 ± 0.064
	Swish	2.004 ± 0.064	1.988 ± 0.085	1.988 ± 0.085
5	PReLU	37.748 ± 4.831	37.290 ± 4.602	37.163 ± 4.661
	Quadratic	36.850 ± 2.205	36.270 ± 1.881	$\mathbf{36.073} \pm 2.003$
	Sigmoid	37.909 ± 3.094	37.509 ± 2.685	37.452 ± 2.667
	Swish	37.032 ± 4.564	36.597 ± 4.321	36.546 ± 4.360
6	PReLU	4.876 ± 0.735	4.699 ± 0.629	4.699 ± 0.629
	Quadratic	4.809 ± 0.560	$\mathbf{4.543} \pm 0.367$	$\mathbf{4.543} \pm 0.367$
	Sigmoid	4.787 ± 0.736	4.609 ± 0.575	4.609 ± 0.575
	Swish	4.838 ± 0.694	4.651 ± 0.571	4.651 ± 0.571
7	PReLU	1.246 ± 0.123	1.167 ± 0.065	$\mathbf{1.083} \pm 0.042$
	Quadratic	1.872 ± 0.093	1.172 ± 0.039	1.092 ± 0.035
	Sigmoid	1.310 ± 0.055	1.151 ± 0.040	1.148 ± 0.037
	Swish	1.205 ± 0.085	1.170 ± 0.060	1.098 ± 0.042
8	PReLU	0.647 ± 0.026	$\mathbf{0.593} \pm 0.028$	0.608 ± 0.023
	Quadratic	0.858 ± 0.015	0.607 ± 0.035	0.594 ± 0.027
	Sigmoid	0.704 ± 0.030	0.633 ± 0.026	0.642 ± 0.025
	Swish	0.626 ± 0.025	0.600 ± 0.029	0.606 ± 0.027
9	PReLU	4.947 ± 0.545	4.421 ± 0.569	4.419 ± 0.567
	Quadratic	6.221 ± 0.285	4.419 ± 0.572	$\mathbf{4.404} \pm 0.571$
	Sigmoid	4.778 ± 0.660	4.499 ± 0.541	4.478 ± 0.553
	Swish	4.711 ± 0.500	4.420 ± 0.570	4.414 ± 0.568
10	PReLU	13.752 ± 1.262	12.803 ± 1.115	12.387 ± 1.133
	Quadratic	16.337 ± 0.378	12.729 ± 1.160	$\mathbf{12.299} \pm 1.146$
	Sigmoid	13.383 ± 1.012	12.497 ± 1.028	12.439 ± 1.021
	Swish	13.589 ± 1.177	12.847 ± 1.193	12.300 ± 1.091

indicates that in general, for these problems, the Adaptive Layer and the Adaptive Neuron approaches produced similar results. Notwithstanding this,

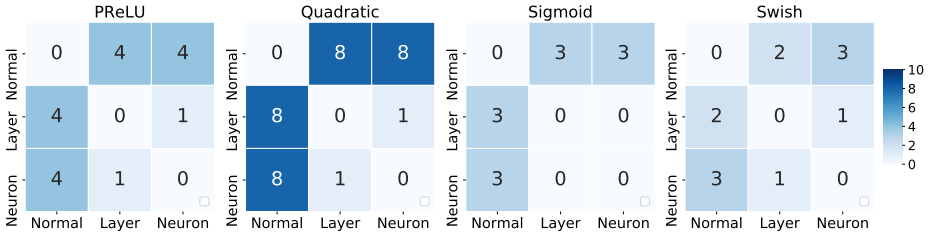

Fig. 1. Counting of adaptive function types that showed a significant mean difference according to the Tukey test.

it is possible to observe that both strategies present a considerable number of problems with a significant difference in the result, fostering the gain in using adaptive activation functions.

Finally, we use Performance Profiles [5] to facilitate the visualization and understanding of the results obtained in the experiments. Performance profiles are cumulative distributions of a performance metric. In this paper, RMSE is used to evaluate the performance of models. In general, a comparison is made between different algorithms to determine the one with the best performance on a specific set of problems. In general terms, the algorithm with the most significant area under the curve is considered the best among the analyzed group, considering the set of problems that were solved by them.

Figure 2 and Table 3 show the performance profiles' results for the different activation functions, different adaptive types, and the ten databases studied. It is possible to observe that the adaptive activation function approach is better for these databases than with the standard function. Besides, the Adaptive Neuron approach can perform better than the Adaptive Layer approach, despite presenting close results.

Table 3. Normalized area under performance profile curves for RMSE metric. Values in bold represent the best result for the metric evaluated.

Activation Function	Normal	Layer	Neuron
PReLU	65.94%	92.98%	**98.95%**
Quadratic	76.13%	98.28%	**100.00%**
Sigmoid	70.63%	99.04%	**99.09%**
Swish	62.91%	91.45%	**99.23%**

Although the research's preliminary results are encouraging, it is necessary to evaluate the proposed approach in a larger number of data sets. In this work, the analysis was limited to 10 data sets. In addition, the proposed strategy should also be tested on classification problems to assess better the contributions of the adaptive activation functions to solve these problems.

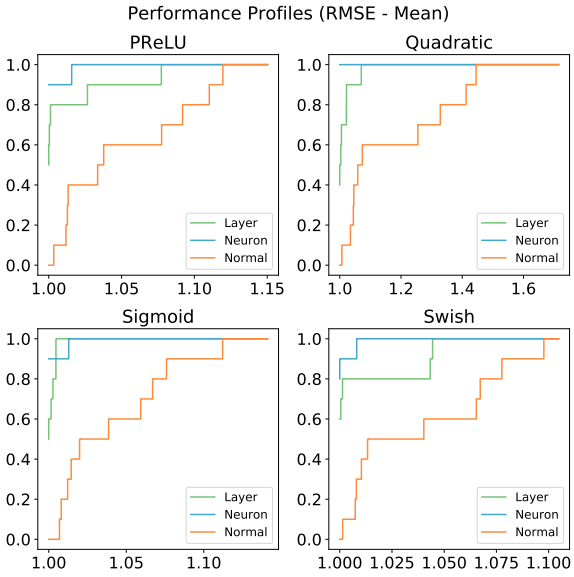

Fig. 2. Performance profile curves for RMSE Metric.

4 Conclusion

This paper investigates Adaptive Activation Functions in a modified version of the Extreme Learning Machine with Backpropagation to solve a regression benchmark's ten problems. The study showed that adaptive Layer and adaptive Neuron performed better than the standard function without increasing the architecture's size. This indicates that the search for an activation function's parameters increases its predictive power, making the need to search for increasingly larger architectures less priority. Regarding the activation functions, the Quadratic function was the one that obtained, in general, the best results for the analyzed databases and the one that presented a more significant performance gain in performing the learning of the parameters. This indicates that, with the analyzes carried out in this work, this adaptive activation function can result in better results in other problems. This study suggests limitations regarding the standard activation function and an alternative to increasing its predictive power. Further research involves searching for optimized architectures working together the adaptive activation functions to produce increasingly accurate solutions with a compact architecture size, reducing the model's complexity while improving its performance.

Acknowledgment

The authors acknowledge the financial support of CNPq (429639/2016-3), FAPEMIG (APQ-00334/18), and CAPES - Finance Code 001. The authors

would like to thank Itaú Unibanco for hours released to its collaborator to develop this work.

References

1. Bishop, C.M., et al.: Neural networks for pattern recognition. Oxford university press (1995)
2. Campolucci, P., Capperelli, F., Guarnieri, S., Piazza, F., Uncini, A.: Neural networks with adaptive spline activation function. In: Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications (MELECON 96). vol. 3, pp. 1442–1445. IEEE (1996)
3. Canziani, A., Paszke, A., Culurciello, E.: An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678 (2016)
4. Chen, C.T., Chang, W.D.: A feedforward neural network with function shape autotuning. Neural networks **9**(4), 627–641 (1996)
5. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical programming **91**(2), 201–213 (2002)
6. Fisher, R.A.: Xv.—the correlation between relatives on the supposition of mendelian inheritance. Earth and Environmental Science Transactions of the Royal Society of Edinburgh **52**(2), 399–433 (1919)
7. Godfrey, L.B., Gashler, M.S.: A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks. In: 2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K). vol. 1, pp. 481–486. IEEE (2015)
8. Guo, P., Cheng, W., Wang, Y.: Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems. Expert Systems with Applications **71**, 57 – 68 (2017)
9. Haykin, S.: Neural Networks - A Comprehensive Foundation, Second Edition. Prentice Hall, 2 edn. (1998)
10. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
11. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: A review. Neural Networks **61**(Supplement C), 32 – 48 (2015)
12. Huang, G.B., Wang, D.H., Lan, Y.: Extreme learning machines: a survey. International journal of machine learning and cybernetics **2**(2), 107–122 (2011)
13. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on. vol. 2, pp. 985–990. IEEE (2004)
14. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1-3), 489–501 (2006)
15. Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E.: Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Journal of Computational Physics **404**, 109136 (2020)
16. Karlik, B., Olgac, A.V.: Performance analysis of various activation functions in generalized mlp architectures of neural networks. International Journal of Artificial Intelligence and Expert Systems **1**(4), 111–122 (2011)

17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Kunc, V., Kléma, J.: On transformative adaptive activation functions in neural networks for gene expression inference. bioRxiv p. 587287 (2019)
19. Lau, M.M., Lim, K.H.: Review of adaptive activation function in deep neural network. In: 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). pp. 686–690. IEEE (2018)
20. Olson, R.S., La Cava, W., Orzechowski, P., Urbanowicz, R.J., Moore, J.H.: PMLB: a large benchmark suite for machine learning evaluation and comparison. BioData Mining **10**(1), 36 (Dec 2017)
21. Piazza, F., Uncini, A., Zenobi, M.: Artificial neural networks with adaptive polynomial activation function (1992)
22. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)
23. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)
24. Saporetti, C.M., Duarte, G.R., Fonseca, T.L., da Fonseca, L.G., Pereira, E.: Extreme learning machine combined with a differential evolution algorithm for lithology identification. RITA **25**(4), 43–56 (2018)
25. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green ai. arXiv preprint arXiv:1907.10597 (2019)
26. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243 (2019)
27. Sulistiyo, M.D., Dayawati, R.N., et al.: Evolution strategies for weight optimization of artificial neural network in time series prediction. In: 2013 International Conference on Robotics, Biomimetics, Intelligent Computational Systems. pp. 143–147. IEEE (2013)
28. Tezel, G., Özbay, Y.: A new neural network with adaptive activation function for classification of ecg arrhythmias. In: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. pp. 1–8. Springer (2007)
29. Tukey, J.W.: Exploratory data analysis, vol. 2. Reading, MA (1977)
30. Vecci, L., Campolucci, P., Piazza, F., Uncini, A.: Approximation capabilities of adaptive spline neural networks. In: Proceedings of International Conference on Neural Networks (ICNN'97). vol. 1, pp. 260–265. IEEE (1997)
31. Wu, R., Huang, H., Qian, X., Huang, T.: A L-BFGS based learning algorithm for complex-valued feedforward neural networks. Neural Processing Letters **47**(3), 1271–1284 (2018)
32. ZahediNasab, R., Mohseni, H.: Neuroevolutionary based convolutional neural network with adaptive activation functions. Neurocomputing **381**, 306–313 (2020)
33. Zou, W., Yao, F., Zhang, B., Guan, Z.: Back propagation convex extreme learning machine. In: Proceedings of ELM-2016, pp. 259–272. Springer (2018)