POLITECNICO DI TORINO
Repository ISTITUZIONALE

Heuristic optimization applied to ANN training for predicting renewable energy sources production

Original
Heuristic optimization applied to ANN training for predicting renewable energy sources production / Lorenti, G., Mariuzzo, I., Moraglio, F., Repetto, M.. - In: COMPEL. - ISSN 0332-1649. - ELETTRONICO. - (2022). [10.1108/COMPEL-11-2021-0420]

Availability:
This version is available at: 11583/2958900 since: 2022-03-28T09:31:14Z

Publisher:
Emerald Publishing Limited

Published
DOI:10.1108/COMPEL-11-2021-0420

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Emerald postprint/Author's Accepted Manuscript, con licenza CC BY NC (articoli e capitoli libri)

This Author Accepted Manuscript is deposited under a Creative Commons Attribution Non-commercial 4.0 International (CC BY-NC) licence. This means that anyone may distribute, adapt, and build upon the work for non-commercial purposes, subject to full attribution. If you wish to use this manuscript for commercial purposes, please contact permissions@emerald.com.

(Article begins on next page)

20 June 2026

Heuristic Optimization Applied to ANN Training for Predicting Renewable Energy Sources Production

structured abstract

Design/methodology/approach (limit 100 words)

This contribute shows the application of heuristic optimization algorithms to the training phase of ANN whose aim is to predict renewable power production as function of environmental variables like solar irradiance and temperature. The training problem is cast as the minimisation of a cost function whose degrees of freedom are the parameters of the neural network. A Differential Evolution algorithm is substituted to the more usual gradient-based minimization procedure and the comparison of their performances is presented.

Purpose (limit 100 words)

Compare stochastic gradient method used for Neural Network training with global optimiser without use of gradient information, in particular Differential Evolution.

Findings (limit 100 words)

The two procedures based on stochastic gradient and Differential Evolution, reach the same results being the gradient based moderately quicker in convergence but with a lower value of reliability as a significant number of runs does not reach convergence.

Research limitations/implications (limit 100 words)

The approach has been applied to two pre problems and, even if results are encouraging, the need for extend the approach to other problems is needed

Practical implications (limit 100 words)

The new approach could open the training of Neural Network to more stable and general methods, exploiting the potentialities of parallel computing

Social Implications (limit 100 words)

no social implications are foreseen

Originality/value (limit 100 words)

The research presented is fully original for the part regarding the Neural Network training with Differential Evolution

Plain Language Summary

The work presented proposes a new method to increase the performance of neural network when applied to prediction problems that, for instance, could largely improve the penetration of renewable energy sources in energy grids.

Abstract. Predicting complex systems like Renewable Energy Sources depending nonlinearly on several physical parameters can be approached by a phenomenological method based on Artificial Neural Networks. This contribute shows the application of heuristic optimization algorithms to the training phase of ANN whose aim is to predict renewable power production as function of environmental variables like solar irradiance and temperature. The training problem is cast as the minimisation of a cost function whose degrees of freedom are the parameters of the neural network. A Differential Evolution algorithm is compared with the more usual gradient-based minimization procedure and the comparison of their performances is presented. In all the cases tested, Differential Evolution training process is showing results in line with those of the differential-based ones, with comparable convergence speed, always showing a more reliable behaviour, avoiding problems of premature convergence and overfitting affecting in some applications 20% of gradient-based procedures.

Keywords: neural networks, optimization, gradient descent, differential evolution.

INTRODUCTION

The use of Machine Learning (ML) techniques applied to predict power production from renewable sources or electric load is becoming more and more widespread in electrical engineering. Different techniques are used going from regression methods [1], [2] to Artificial Neural Networks (ANN) [3], [4], [5]. The core of most of these techniques is based on the training process of a computational structure that is tuned to follow a nonlinear mapping between inputs and outputs. In neural prediction schemes, the quantity to be predicted y is computed as a function of regressors x, and no predefined model of this relation is set. Prediction is thus based on the minimization of the error between actual response values y_i and model output $\hat{y}_i$, where i denotes a sample belonging to the dataset.

When ANN are used, the training process is implemented through an optimization problem, that is looking for the minimum of an error function and it is usually performed by means of a gradient-based algorithm. This choice is due to the peculiar feature of ANN making the evaluation of the gradient of the error function computationally efficient using back-propagation scheme [6]. As the optimization problem is non-convex, the use of a deterministic gradient technique can lead to poor results as the "*error-surface may contain local minima so that gradient descent is not guaranteed to find a global minimum*" [6]. To overcome this problem, several stochastic, gradient-based methods have been developed, as for instance in [7]. A complete review of this class of minimization algorithms can be found in [8]. ADAM (ADAptive Moment estimator) method belongs to this class of algorithms and its use regarding Neural Networks optimization has become more and more widespread.

In this work, a different approach is proposed which uses a global optimizer based only on the value of error of the prediction, avoiding the computation of its gradient. The well-known Differential Evolution (DE) optimizer [9] is in fact adopted for this purpose and an extensive study of the performance of an ANN trained by ADAM and DE is performed.

In the following, a brief presentation of the ANN structure is given and its training by means of ADAM and DE is described. After that, the problem of predicting PhotoVoltaic (PV) power production is defined and the analysis of the accuracy and computational efficiency of ADAM- and DE-based training methods is discussed. Moreover, a similar comparison is performed using the same algorithms in the context of predicting Solar Thermal (ST) power production. For both case studies, various ANN architectures are tested using both optimisation methods. The architectures, however, are kept the same between case studies.

Finally, a discussion about the obtained results is presented.

ARTIFICIAL NEURAL NETWORK FOR PREDICTION

Artificial Neural Network architecture

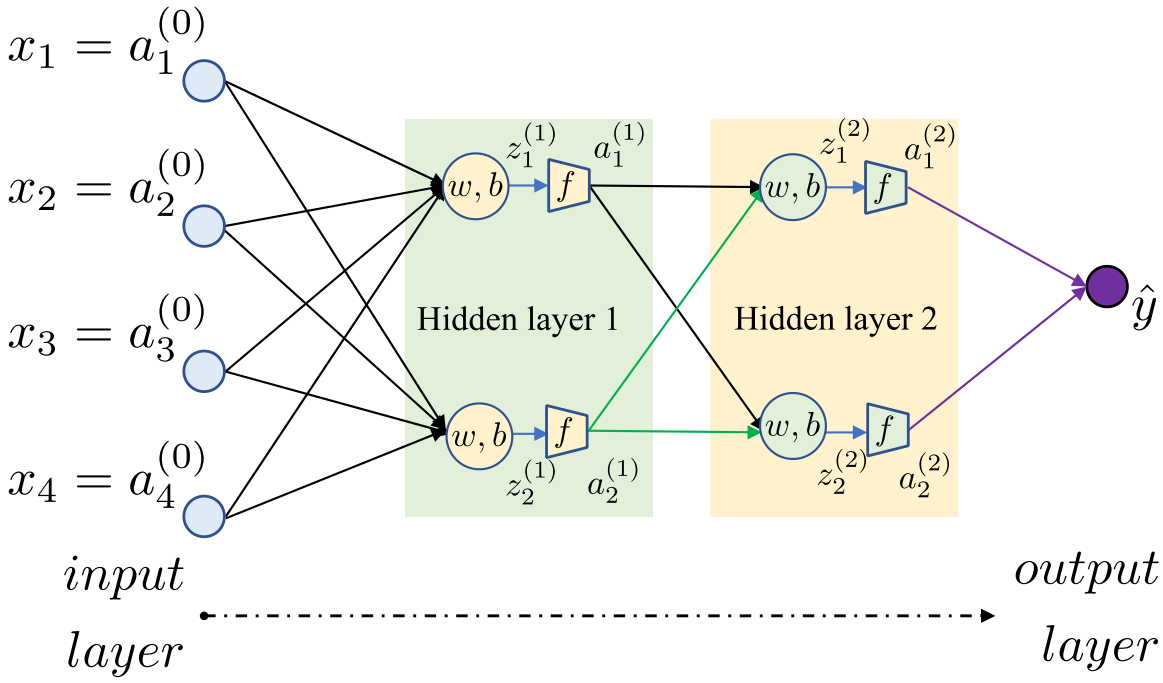

Figure 1: Artificial Neural Network architecture with two layers and two neurons for each layer.

The fundamental element of a neural network structure is the artificial neuron, a statistical model of its biological counterpart. In a similar fashion, simplest ANN architectures can be obtained by grouping a certain number of neurons in a series of layers, whose notation can be seen in Figure 1. This kind of ANN is one of the earliest neural computation models, and it is named Multi-Layer Perceptron (MLP) [13]. An MLP is a feedforward neural network, in that neurons in each layer have as their inputs the output signals of the preceding layer only. In other words, no feedback mechanism is considered.

In MLPs and other layered Neural Networks it is customary to distinguish between

1. *Input layer*: it is not accounted as a "real" layer belonging to the ANN, because its number of units is equal to the number of features in the training set, for instance (x_1, x_2, x_3, x_4) in Figure 1;
2. *Hidden layer(s)*: they are a series of fully connected layers (i.e. each neuron has connections/weights with each neuron of previous layer) representing the "black box" structure connecting input and output layers;
3. *Output layer*: this is the last one and its output is the global prediction of the ANN ($\hat{y}_i$ in figure 1).

Weight and bias values are the parameters which will be optimized while the neural network is being trained, where the former can be interpreted as synapses in a human brain while activation function mimics the mechanism of excitation of a neuron (firing). Starting with the input layer, the output of each layer is computed and is used as input to feed the next layer. In conclusion, to compute a layer output, two steps are needed:

- *Affine calculation*: is a linear evaluation by multiplying the weight matrix by the input X (i.e. output by previous layer) and by adding a biasing term.
- *Activation function*: it can be a non-linear evaluation through a proper function fed by previous linear calculus to obtain layer's output.

In all the analysis presented in this paper each layer is equipped with the Rectified Linear Unit (ReLU) activation function, with exception of the last one where the so-called identity activation function is applied transferring the output outside. For each hidden layer in the network, the affine calculations can be evaluated as

$$Z^{[l]} = W^{[l]} * A^{[l-1]} + b^{[l]}, \qquad l = 1, \dots, L,$$

where l is the index of the layer, $A^{[l-1]}$ is the activation output by previous one ($A^{[0]}$ is equal to the input values X for the first layer) and $b^{[l]}$ is the $l\text{-}th$ bias term. Then, by using ReLU activation function, the output is evaluated as

$$A^{[l]} = ReLU(Z^{[l]}) = \max\left(0, Z^{[l]}\right), l = 1, \dots, L - 1.$$

For the last layer, it is evaluated by using identity activation function, $A^{[L]} = Z^{[L]}$. Weight matrices and biases shapes depend on the number of neurons of the current and past layer. In general, the weight matrix of layer l, W_l, is has shape (N_l, N_{l-1}), where N_l is the number of neurons of the current layer and N_{l-1} is the number of units of the past one. Instead, the biasing term is a column vector of size N_l. By summing the number of weights and biases of each layer it is possible to estimate how many parameters are involved in the optimization:

$$\#parameters = \sum_{l=1}^{L} \left(N^{[l]} * N^{[l-1]}\right) + b^{[l]}.$$

First-order stochastic optimization algorithm

First-order optimizers can minimize or maximize an objective function by means of sequential evaluation of its derivatives with respect to the optimization variables, hence are well suited for applications involving continuous, differentiable and convex objective functions. In many problems, however, the non-convexity of the objective function may cause such methods to be trapped in local minima (or maxima) while looking for the global one in the variable search space. Graphically it can be represented as a crooked path going from initial (random) to optimal conditions with many spikes and oscillations. Several strategies have been investigated to address this point [7] by updating parameters not with past gradients directly, but using first or second order momentum of the past gradients. Stochastic Gradient Descent (SGD) with Momentum evaluates the exponential moving average of the gradients and uses it to update the parameters according to the learning rate; RMSProp performs similar calculations by using the square of the gradients instead. ADAM, which has been used in this paper, combines both first- and second-order moment estimates to update the parameters of the ANN and the path towards global minimum results to be much straighter and smoother than that obtained using simple gradient-based optimization. In this paper, the stochastic nature of the procedure is only due to the random initialization of both weights and biases. Since no batching is performed, such assumption makes each run different from the others and results must be averaged over multiple runs of the algorithm. The particular implementation of the ADAM algorithm used in this work can be found in [12]. The ADAM procedure can be run by specifying two parameters that are under user control, namely the (exponential) decay parameters involved in computing the moving averages of first- and second-order moment estimates. All parameters are used with default values but the following two:

- Learning rate: that is the update ratio of the gradient vector in new iterations, here set at the value of 1E-4.
- Stopping criteria: that is the number of iterations without further improvement of the objective function after which the procedure is stopped even if convergence has not been reached, Early Stopping, set at 10 epochs.

DIFFERENTIAL EVOLUTION GLOBAL OPTIMIZER

Short description of Differential Evolution technique

Differential Evolution (DE) refers to a class of population-based global optimizers [10]. Such algorithms are stochastic in nature and only require the evaluation of the objective function in a certain number of points for aiming at the optimal solution. The points where the objective function is evaluated are the individuals of the population, which are usually represented as vectors in a n-dimensional space. The population is randomly initialised within the feasibility domain and it is then iteratively updated at each new generation through a sequence of operations which consists in mutation, crossover, selection. The algorithm requires a small number of control parameters to be set: the population's size, NP, the mutation factor, F, and the crossover probability, Cr. The mutation process is the characteristic feature of all DE algorithms [10]. In the classical version, mutation consists in creating a new individual (mutation vector) from three randomly selected individuals (v_1, v_2, v_3), each one different from the others: v_3 provides a base vector, to which a differential term is added, which is evaluated as the difference between v_1 and v_2 scaled by the mutation factor. A trial vector, v_{trial}, is then created performing a crossover between the mutant and a fourth vector, v_x, chosen from the population. The crossover phase consists in randomly selecting elements from v_x or the mutant, basing on the crossover probability, thus mitigating the effects of the mutation search strategy [10]. The classical version of the algorithm also forces at least one element to be taken from the mutant, so that at least a different value from those of v_x is considered. Finally, v_{trial} and v_x are compared in terms of objective function, keeping the best one as a new individual for the next generation's population. This process is performed on all individuals at each generation, for a specified number of generations or until convergence (with respect to a given stopping criterion) is reached.

Differential Evolution applied to Artificial Neural Network

In this study, DE strategy has been applied to the training phase of an ANN. The individuals in the population are vectors of Neural Network parameters values (weights and biases), hence each individual corresponds to a different ANN. A variation to the classical DE strategy has been performed, as the best individual in each generation is chosen as the base vector for the mutation strategy, rather than randomly selecting it from the population. The cost function is evaluated as the Mean Squared Error (MSE) of the ANN output on the training set. The hyper-parameters have been set up as:

- $NP = 50$;
- $F = 0.3$;
- $Cr = 0.6$.

The optimisation procedure is carried out for 400 generations. However, an early stopping may occur from the 10-th generation on, in case there is no significant variation in the objective function for ten consecutive generations.

CASE STUDIES

Photovoltaic plant prediction

The first case study in this paper involves the optimization of neural network parameters for predicting time series data extracted from a real PhotoVoltaic (PV) plant located in northern Italy. Input dataset contains both environmental variables or features and a target variable (in this case, photovoltaic power), monitored quarter-hourly from 23/7/2018 to 01/01/2021, for a total of 85812 samples. The features used here are

- global irradiance [W/m^2];
- module temperature [°C].

No data referring to external temperature were available in the dataset. Then, dataset has been divided in training, validation and test set. The difference between the last two subsets is that the former is used to check model performances over unseen data during its training at each epoch, while the latter is used to perform predictions after the ANN has been trained. To boost convergence, features have been scaled with respect to the relative maximum value, then 70% of the dataset is used to train the ANN while the remaining is equally separated into validation and test set. No batching is applied, so the whole training set is processed once optimizing.

Solar Thermal plant prediction

The second case study refers to a Solar Thermal (ST) plant power prediction located in EU. Input data in this case are represented by a year monitoring period going from 01/05/2018 to 01/05/2019 on an hourly base (i.e. 8760 values for each environment variable and target variable). In this second case, involved features to predict the target (solar thermal power) are

- global irradiance [W/m^2];
- external temperature [°C];
- water flow rate [m^3/s];
- supply temperature [°C];
- return temperature [°C];
- supply-return temperature difference [°C].

Even if the latter feature is directly connected to the solar power delivery, it has been discarded since it is evaluated as difference of two already monitored temperatures and hence to avoid data redundancy. Also in this case study, the data have been divided according to the following procedure: 70% in training set and 15% for both validation and test sets and, no batch size is applied and same scaling procedure has been adopted.

NUMERICAL TESTS

To evaluate how both optimization strategies are effective, their ability to tune weights and biases has been tested over the two case studies described above. Several neural network architectures have been investigated by progressively increasing the number of hidden layers from 1 to 5, keeping constant and equal to 5 the number of neurons for each layer. Hence, the total number of trainable parameters (W and b) changes according to the equation presented previously, setting the number of neurons equal to 5 for all layers but the input and output

ones. The Mean Squared Error (MSE) has been used as the loss function to be minimized and the Normalized Mean Absolute Error (NMAE) is the metric used to present the two models' output. In the latter, for both scenarios (Photovoltaic and Solar Thermal) the Mean Absolute Error (MAE) has been normalized with respect to the maximum value of power in the dataset. Due to the stochastic nature involved in both algorithms, their performances have been evaluated on 50 runs and then comparison has been performed by taking into account the median value of both NMAE values over the validation set and the CPU time required by the simulations on a state-of-the-art laptop PC. Distributions of all 50 runs results are anyway stored and used for plot output. The tests have been conducted on a laptop whose processor is an Intel ® Core™ i7-10510U CPU with clock speed 2.30 GHz and memory size of 16 GB.

ADAM and Differential Evolution comparison for Photovoltaic

The comparison is made firstly by looking at the violin plots reporting the distribution of results over the runs. In all violin plots presented, the yellow line represents the median value while the green triangle is the arithmetic average of the results over 50 runs.

Main results obtained are reported in figures from 2 to 6 for PV case study. As it can be seen from the analysis of NMAE, both methods reach a value of NMAE which is around 0.015 in all ANN configurations. A comparison of median NMAE values between two methods is reported in Fig. 4 as function of the number of hidden layers while Fig. 5 reports the computational time versus the number of degrees of freedom: by fitting it by means of a polynomial curve it can be seen the CPU time of ADAM is increasing (almost) linearly while the DE optimizer shows a non-negligible quadratic term. In Fig. 6 the actual and predicted curves of PV power production are shown with the DE-trained network. It must be remarked that, due to the very similar NMAE values, the two reconstructions are practically coincident. By the analysis of the curves, it can be appreciated that most of the error is located close to region of sharp change in production, where the measured values of irradiance can be affected by uncertainties due to quick changes created by meteorological phenomena.

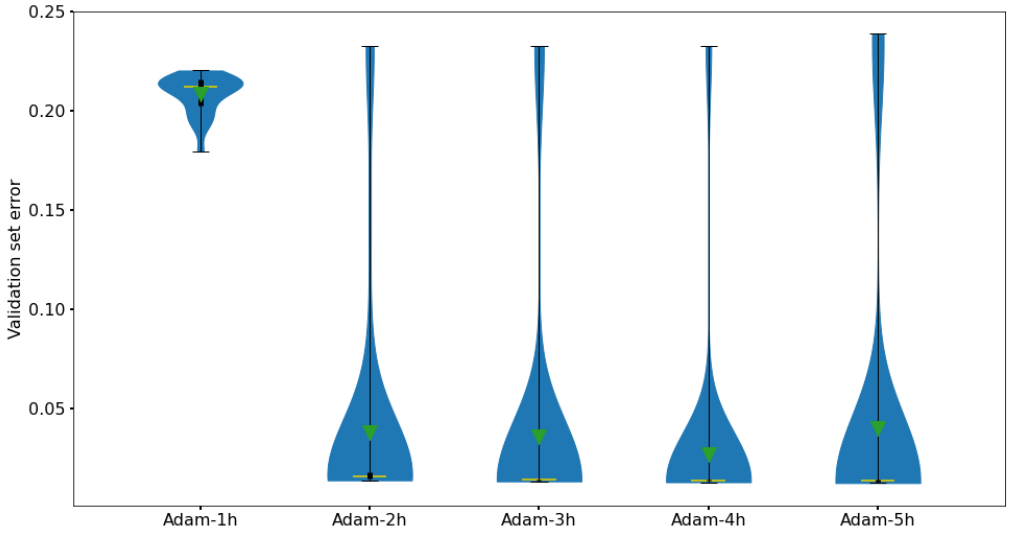

Figure 2: PV validation set NMAE for ADAM

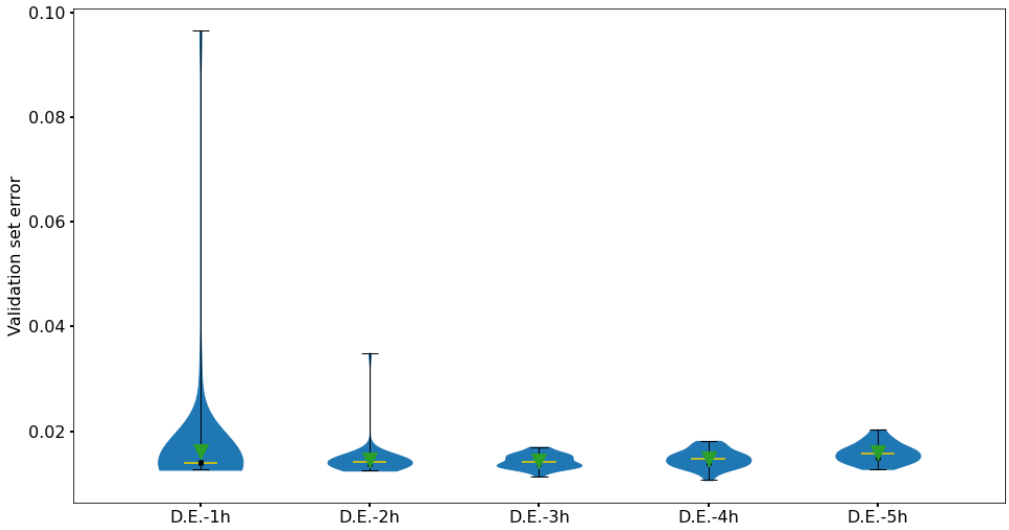

Figure 3: PV validation set NMAE for DE

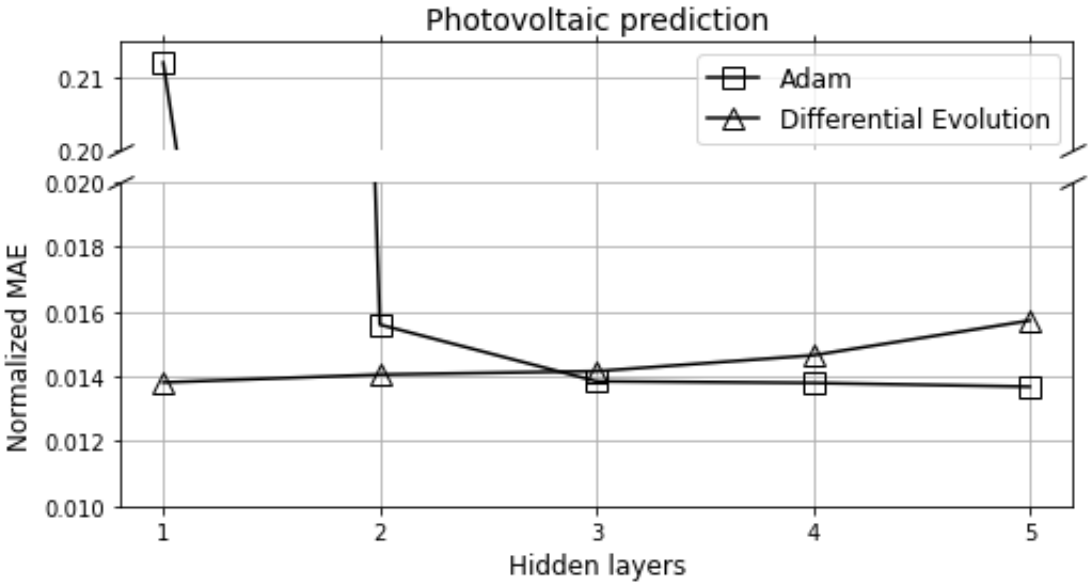

Figure 4: ADAM and DE NMAE median values comparison

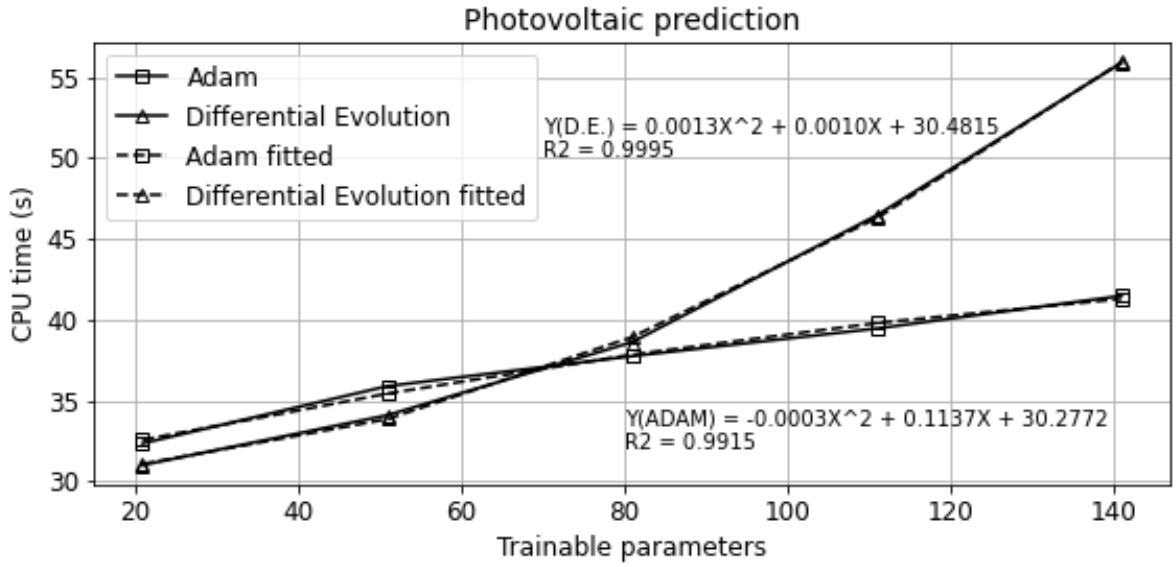

Figure 5: ADAM and DE computational time

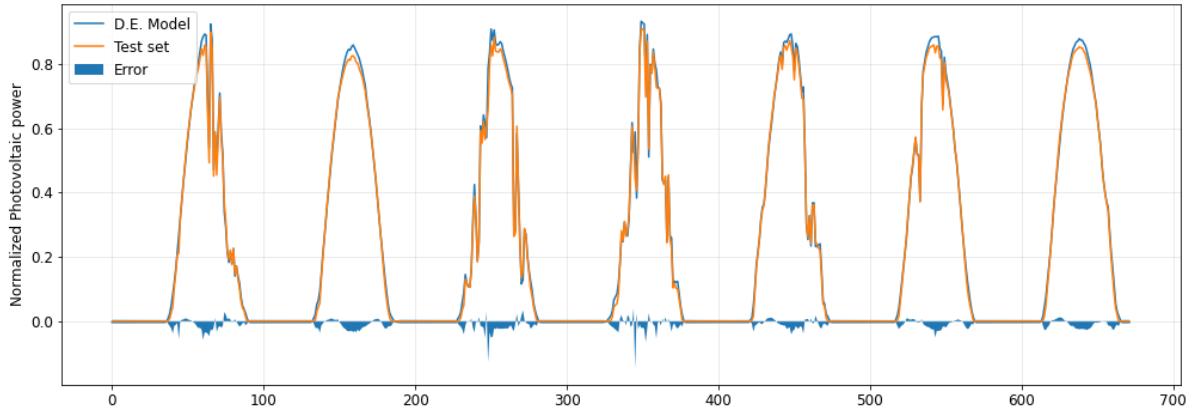

Figure 6: Photovoltaic prediction over test set. First week

ADAM and Differential Evolution comparison for Solar Thermal

Similar comparisons can be performed for the solar thermal test case. This time the problem to be estimated is more complex because the power production is not mainly depending on the solar irradiation but also the external temperature is playing an important role. The role of complexity can be seen in the behaviour of the NMAE values versus the number of hidden layers (fig. 7 and 8). Both methods reach error values larger with 1 hidden layer and their response improves for 2 and 3 hidden layers. ADAM trained networks seem to be steadily improving their performance with respect to the number of hidden layers, while DE error slightly increases for 4 and 5 hidden layers (fig. 9). Also in this case computational time of ADAM networks is increasing mostly in a linear way while DE ones seem to increase with the size of the network, after a minimum is attained at 3 hidden nodes (fig. 10). Power is again well reconstructed and, also in this case, the largest part of the error is located close to sharp changes in output (fig.11).

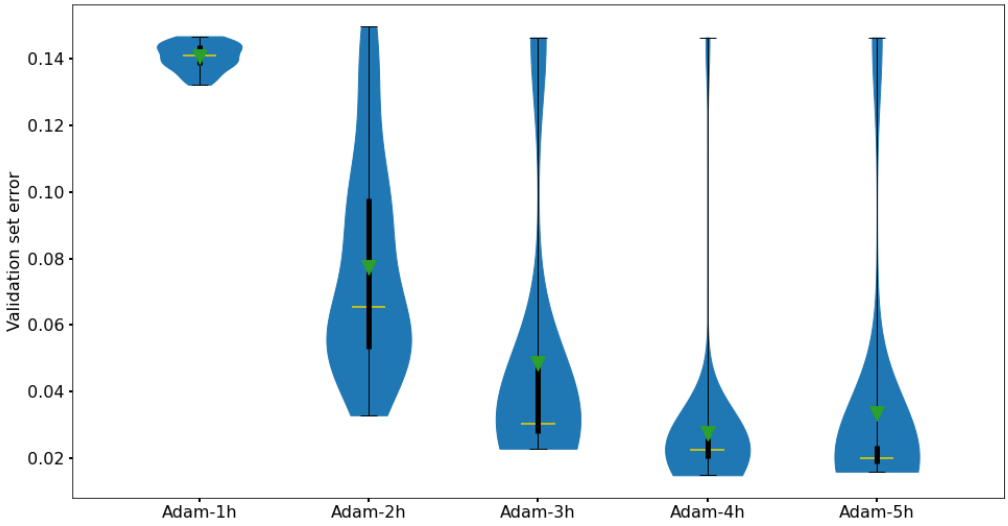

Figure 7: Solar thermal validation set NMAE for ADAM

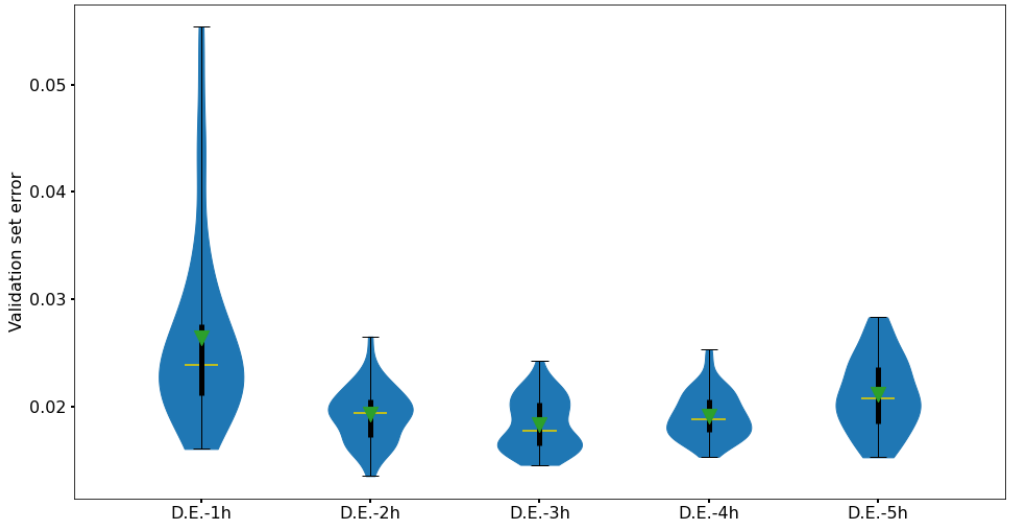

Figure 8: Solar thermal validation set NMAE for DE

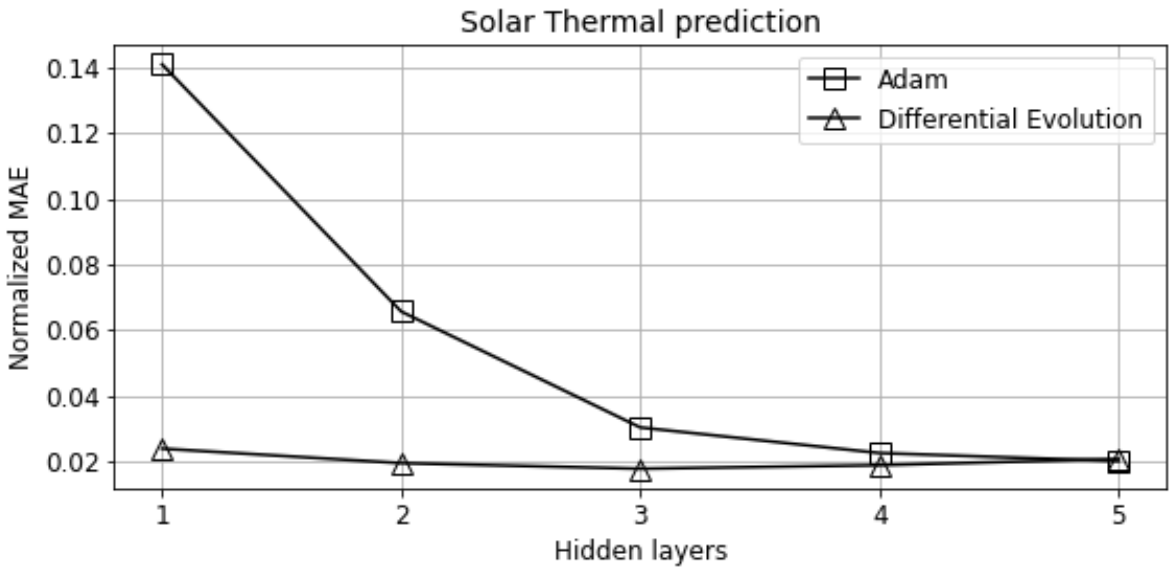

Figure 9: ADAM and DE. NMAE median values comparison.

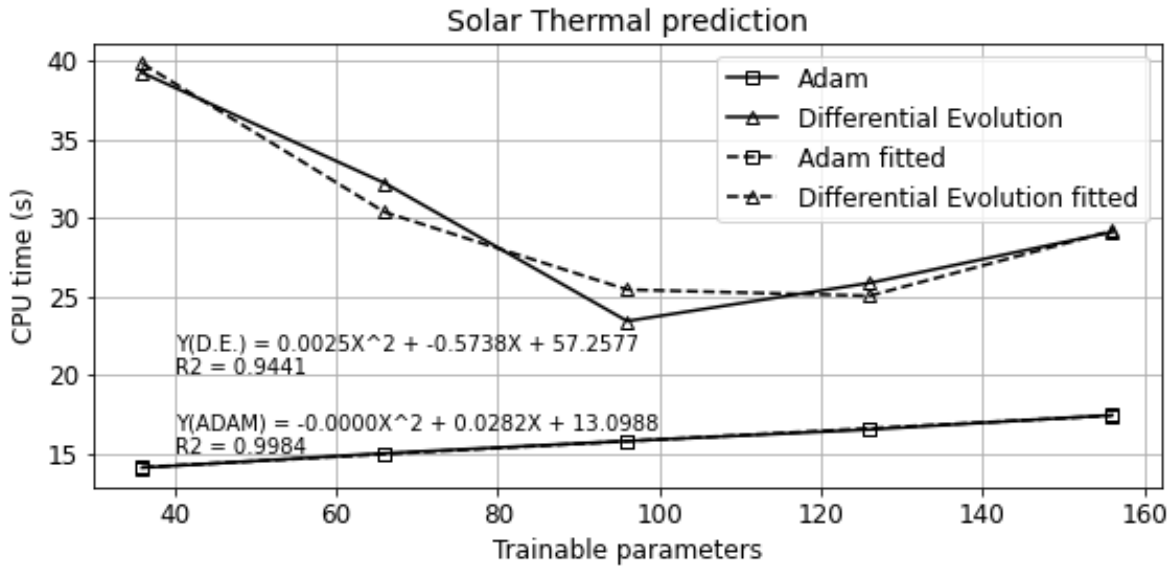

Figure 10: ADAM and DE. Computational time.

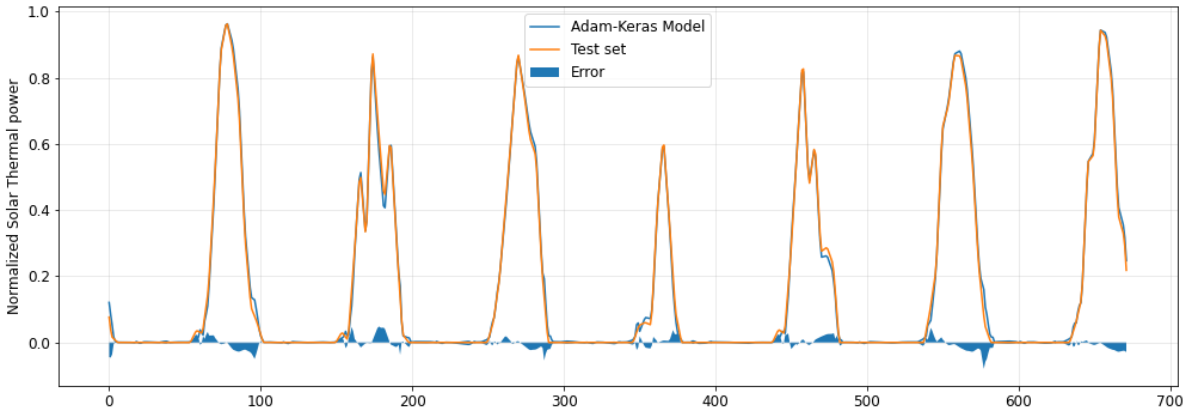

Figure 11: Solar thermal prediction over test set first week

By analyzing the behavior of the two methods in the 50 runs some differences between them can be outlined. In fact, in both test cases ADAM median and average values of NMAE are quite different (figs. 2 and 7). The existence of this pattern clearly depends on the presence of several outlying values among predictions (and hence in errors). This can be interpreted as a symptom of overfitting, which greatly increases variance of the model.

From the violin plots, it can be observed that, especially when employing two or more hidden layers, error distributions exhibit strong bimodality.

This is not true in the case of DE-optimized neural networks, whose errors tend to follow a unimodal, bell-shaped distribution. These results suggest that such metaheuristics-based methods are more robust to overfitting even if they require less fine tuning of training parameters. In other words, training procedures like the one based on DE

allow researchers and technicians to further automate the development of neural prediction models, thus reducing the amount of time required.

In order to quantify this phenomenon, an outlier detection search has been performed. For the sake of interpretability, we employed the classical criteria of inter-quartile range (IQR), despite it assumes a gaussian (and hence unimodal) distribution of observations. This method highlights values that are significantly out of the range between 75th (Q3) and 25th percentile (Q1) in particular, defining the inter-quartile range IQR= Q3 – Q1, outlier values are detected if they lie out of the range (Q1 – 1.5*IQR, Q3 + 1.5*IQR).

By using this criterion, it can be seen that ADAM results have a significant number of outliers in both cases (Figs. 12 and 13) reaching in some cases fractions close to 20%.

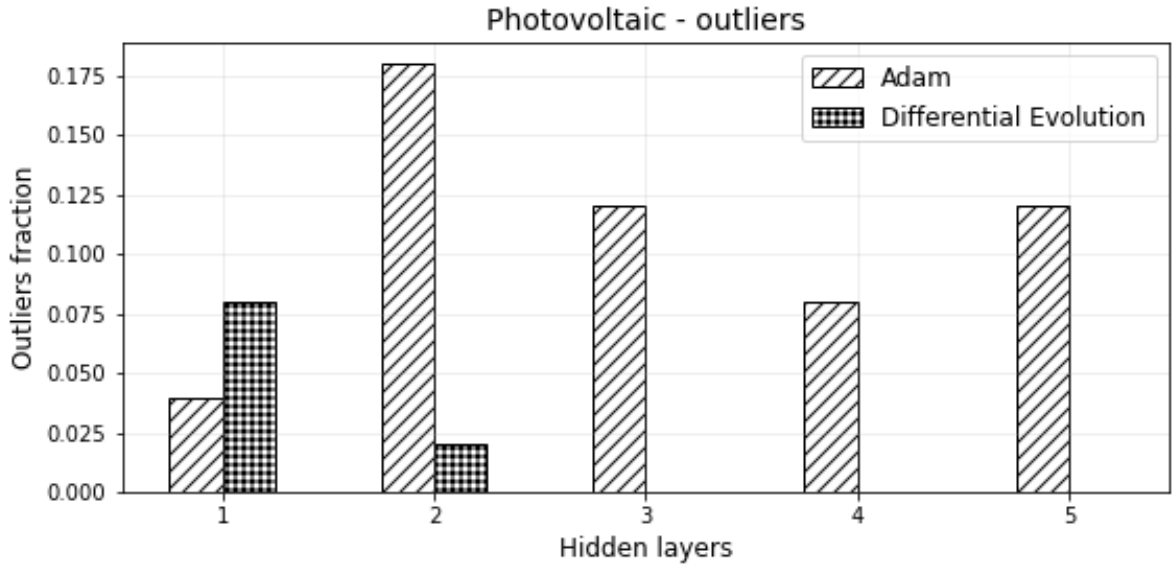

Figure 12: Adam and D.E. error outlier detection for photovoltaic over 50 runs

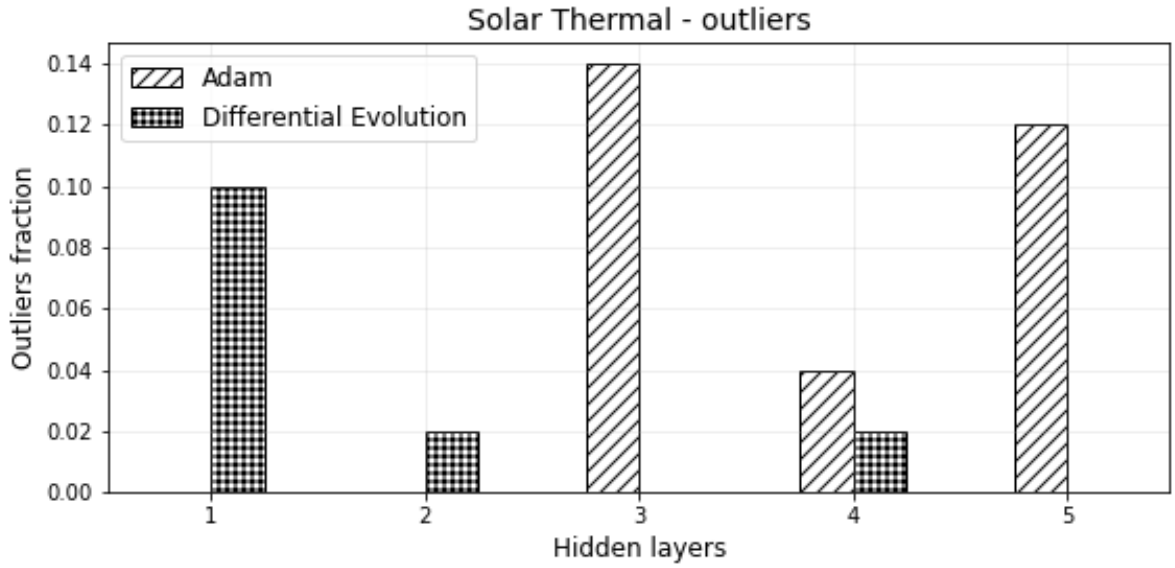

Figure 13: Adam and D.E. error outlier detection for solar thermal over 50 runs

DISCUSSION AND PRELIMINARY CONCLUSIONS

This work has allowed us to assess that a zero-th order optimization algorithm can reach performance values similar to those of more widespread training procedures based on first order minimization as ADAM. As the results obtained are relevant only to two cases, the analysis of results can only be preliminary, anyway some trends can be noted.

1. The two minimization algorithms have different behaviours: ADAM is faster and its computational burden seems to be only slightly increasing with respect to the size (number of degrees of freedom) of the problem. On the contrary, DE results appear to be more stable as it has been pointed out by an outlier analysis. These considerations are in line with those provided in [11] even if on completely different test cases.
2. Both problems are well tackled by a limited number of hidden layers and it seems that the error is not much decreasing with respect to the increase of the hidden layer numbers.
3. ADAM procedure computational costs are lower than those of DE but it must anyway be remarked that the present study compares one well established ADAM procedure with a research implementation of DE where some space for code efficiency increase is still foreseen.
4. Both problems have been analysed in terms of their outcomes but a future development will require an analytical definition of computational costs to point out possible trade-offs between accuracy and efficiency of the training procedures.

REFERENCES

[1] N. I. Sapankevych and R. Sankar, "Time Series Prediction Using Support Vector Machines: A Survey,", 2009, IEEE Computational Intelligence Magazine, vol. 4, no. 2, pp. 24-38, May 2009.

[2] F. Martınez-Alvarez, A. Troncoso, G. Asencio-Cortes 1, J. C.Riquelme, "A Survey on Data Mining Techniques Applied to Electricity-Related Time Series Forecasting", 2015, Energies 2015, 8, 13162–13193.

[3] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network,", IEEE Transactions on Smart Grid, vol. 10, no. 1, pp. 841-851, Jan. 2019

[4] L. Gigoni, A. Betti, E. Crisostomi, A. Franco, Ma. Tucci, F. Bizzarri, D. Mucci (2017), "Day-Ahead Hourly Forecasting of Power Generation From Photovoltaic Plants", 2018, IEEE Transactions on Sustainable Energy, Vol. 9, NO. 2, April 2018

[5] S. Leva, A. Dolara, F. Grimaccia, M. Mussetta, E. Ogliari, "Analysis and validation of 24 hours ahead neural network forecasting of photovoltaic output power", 2017, Mathematics and Computers in Simulation 131 (2017) 88–100

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back- Propagating Errors”, Nature, 323:533-536, 1986.

[7] E. Barnard, "Optimization for training neural nets," in IEEE Transactions on Neural Networks, vol. 3, no. 2, pp. 232-240, March 1992.

[8] S. Ruder, "An overview of gradient descent optimization algorithms", http://sebastianruder.com/, accessed on Sept. 29, 2021.

[9] R.Storn and K.Price, "Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces", Tech. Report, International Computer Science Institute (Berkeley), 1995.

[10] T. Eltaeib and A. Mahmood, “Differential Evolution: A Survey and Analysis”, 2018, Applied Science, vol. 8, no. 10: 1945.

[11] Baioletti, M.; Di Bari, G.; Milani, A.; Poggioni, V. Differential Evolution for Neural Networks Optimization. Mathematics 2020, 8, 69.

[12] F. Chollet, et al., "Keras", (2015), https://keras.io, accessed on October 13th, 2021.

[13] Graupe, D. “PRINCIPLES OF ARTIFICIAL NEURAL NETWORKS”, World Scientific Publishing, Singapore, 1997.