SOLVER PARAMETER INFLUENCE ON THE RESULTS OF MULTILAYER PERCEPTRON FOR ESTIMATING POWER OUTPUT OF A COMBINED CYCLE POWER PLANT

Prof. PhD. Prpić-Oršić Jasna[1], PhD. Mrzljak Vedran[1], PhD. Student Baressi Šegota Sandi[1], PhD. Student Lorencin Ivan[1]
[1]Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia
E-mail: jasna.prpic-orsic@riteh.hr , vedran.mrzljak@riteh.hr, sbaressisegota@riteh.hr, ilorencin@riteh.hr

Abstract: *Previous work has determined the ability of using the Multilayer Perceptron (MLP) type of Artificial Neural Network (ANN) to estimate the power output of a Combined Cycle Power Plant (CCPP) in which optimization did not focus on the solver parameter optimization. In previous work, the solvers used the default parameters. Possibility exists that optimizing solver parameters will net better results. Two solver algorithm's parameters are optimized: Stochastic Gradient Descent (SGD) and Adam, with 140 and 720 parameter combinations respectively. Solutions are estimated through the use of Root Mean Square Error (RMSE). Lowest RMSE achieved is 4.275 [MW] for SGD and 4.259 [MW] for Adam, achieved with parameters:* $\alpha = 0.05$, $\mu = 0.02$, *and nesterov=True for SGD and with parameters* $\alpha = 0.001, \beta_1 = 0.95, \beta_2 = 0.99$, *and amsgrad=False for Adam. Only a slight improvement is shown in comparison to previous results (RMSE=4.305 [MW]) which points towards the fact that solver parameter optimization with the goal of improving results does not justify the extra time taken for training.*

KEYWORDS: COMBINED CYCLE POWER PLANT, MULTILAYER PERCEPTRON, ARTIFICIAL NEURAL NETWORKS, STOCHASTIC GRADIENT DESCENT, ADAM, SOLVER ALGORITHM

1. Introduction

Gas Turbine (GT) power plants are common in such cases where power along with heat production is acceptable. GTs are characterized with a higher heat production, which is commonly released into the atmosphere [1]. To avoid such losses, this heat can be used for steam production, which is in turn used by a steam turbine (ST) [2]. These turbines are used in various applications such as power production and ship propulsion systems. Such systems, which combine the use of GT and ST are commonly referred to as Combined Cycle Power Plants (CCPP) [3]. Determining the total power production of such complex systems is extremely complex, leading to use of advanced techniques – such as artificial intelligence (AI) methods for their determination [4, 5]. Regressive AI methods, such as regression MLP ANN can be used to great effect in many fields such as maritime sciences [6, 7], robotics [8, 9] and medicine [10]. Previous work by Lorencin et al. in *"Genetic Algorithm Approach to Design of Multi-Layer Perceptron for Combined Cycle Power Plant Electrical Power Output Estimation"* shows the ability of using Evolutionary Computing (EC) methods for optimization of MLP ANN parameters. EC algorithms are another branch of AI algorithms widely used for optimization [11]. In paper in question, authors propose a solution that optimizes the parameters of the MLP through the use of a genetic algorithm. Energy analyses like the one performed here are extremely important in energetics with the goal of predicting the satisfaction of energy needs and possible failures [12-14]. The optimal architecture found by authors was one with five hidden layers with 80, 25, 65, 75 and 80 neurons in each, with Sigmoid, Tanh, ReLU, ReLU and ReLU activation function for each respective layer and 13 epoch executions, and it achieved a RMSE of 4.305 [MW]. When it comes to the solvers used, authors have only varied the algorithm used between SGD and Adam. As these algorithms have the ability of setting various parameters, this paper proposes use of grid search method in the attempt of finding a better solution, with a smaller error. RMSE will be used to estimate quality of solutions, with better solutions being ones with smaller error. Goal of the paper is to achieve a smaller error than previously achieved through adjusting the solver parameters.

2. Analyzed Power Plant and Dataset Description

This chapter describes the powerplant dataset used in the description, as well as the CCPP on which the measurements for dataset creation were made.

2.1. Analyzed Combined Cycle Power Plant

The CCPP used in this research consists of two GT and on ST. The motion energy of the described turbines is transmitted to power generators. The exhaust of GTs is connected to a heat recovery steam generator (HRSG), consisting of high pressure (HP) and low pressure (LP) ST. The schematic of the system is given on Figure 1.

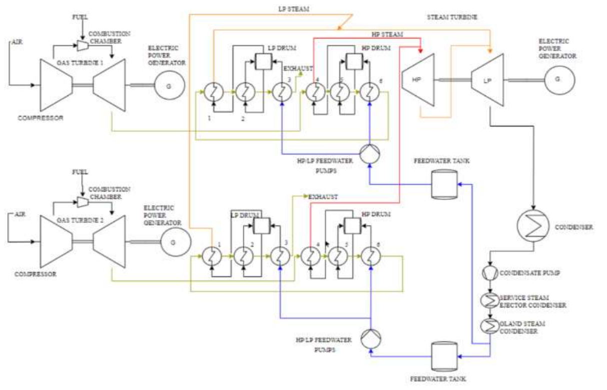

Fig. 1. *Schematic of the CCPP used in research [5]*
(1 – LP super heater; 2 – LP evaporator; 3 – LP economizer; 4 – HP super heater; 5 – HP evaporator; 6 – HP economizer)

System consists of two 160 [MW] ABB 13E2 GTs and one 160 [MW] ABB ST. Power plant used in this research has a nominal power generation capacity of 480 [MW].

2.2. Dataset Description

Dataset consists of 9568 total samples. Each sample consists of five parameters:

- temperature (T),
- ambient pressure (AP),
- relative humidity (RH),
- exhaust vacuum (V), and
- net hourly electrical energy output (EP).

First four parameters (T, AP, RH, V) are used as inputs, while the final (EP) is used as the output. T, AP and RH are measured as GT air intake parameters, and V is measured as ST exhaust vacuum. EP is measured at the electric power generators output. Ranges and units of the parameters are defined in Table 1.

Table 1. *Ranges of individual parameters*

Parameter	*Minimum*	*Maximum*	*Unit*
T	1.81	37.11	°C
AP	992.89	1033.3	mB
RH	25.56	100.16	%
V	25.36	81.56	cmHg
EP	420.26	495.76	MV

3. Multilayer Perceptron description

MLP is a feed-forward ANN consisting of a single input layer, a single output layer and one or more hidden layers. Each layer consists of neurons, that are connected to neurons in previous and following layers with connections. Input layer has a number of neurons equal to the inputs in the dataset (four in this case), while the output layer has a single neuron. Each connection connects the output of a neuron to the input of the other. When discussing MLP, usually a fully connected architecture is discussed. This denotes that each of the neurons in one layer is connected to every neuron in the following layer [15].

Each neuron has a value, with value of the neuron in the output layer being the output value of the ANN. Values of the input neurons are the input values of the dataset. Value of the output and hidden layer neurons is calculated as the weighted sum of inputs, passed to the activation function as shown in [15]:

$$f(X_t) = \mathcal{F}(X_t \cdot \Theta) = \mathcal{F}(x_1 \cdot \theta_1 + x_2 \cdot \theta_2 + \cdots + x_n \cdot \theta_n) \quad (1)$$

in which the activation function is a function which maps the input of the neuron to the output. Basic activation functions are commonly designed to keep the data within a wanted range (sigmoid, TanH), or to eliminate the unwanted elements (ReLU); X_k is the input vector – vector containing each output value of neurons connected to the neuron, with $x_1$to x_n being individual output values; and Θ is a vector of connection weights, with θ_1 to θ_n being the connection weight values corresponding to each of the input values $x_1$to x_n [16].

ANNs are multi-stage algorithms, consisting of a training and testing stage. First data is split into training and testing sets. In this research 7500 data points were used in the training set, with the remaining 2068 points being used in the testing set. Each iteration of the training stage is split into two parts – so called forward and backward propagation. First the initial weights of the MLP are set randomly. During the forward propagation input data from each data point is set as input to the neural network and propagated through the hidden layers. Once the output is reached the real, expected value (known from the dataset) is compared to the received value from the neural network. The difference between the expected and received value is used as the cost function (also known as Loss), defined as [15, 17]:

$$J(\Theta) = MSE = \frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{2n}, \quad (2)$$

where cost function $J(\Theta)$ is defined as a mean square error in which y_i is the expected value of the ANN output, $\hat{y}_i$ is the predicted value, in other words the value calculated by the ANN, and n being the total number of data points in the set.

We aim to lower this value to zero, and to achieve this the weights need to be adjusted. This process is called backward propagation, in which the error of the neural network is propagated from the output to the input and each connection weight is adjusted. The amount of adjustment depends on the size of the error – with larger errors causing a larger weight adjustment. To determine the adjustment necessary the gradient is used, calculated as the partial derivative of the cost function by the current weight:

$$\Theta' = \frac{\partial J(\Theta)}{\partial \Theta}. \quad (3)$$

The connection weight is being updated using:

$$\Theta_{new} = \Theta_{old} - \frac{\alpha}{n} \cdot \Theta', \quad (4)$$

with the new weight value being denoted by Θ_{new}, old value by Θ_{old}; gradient Θ' is determined by equation (3), n is a total number of datapoints, and α is the learning rate. Learning rate is a very important hyperparameter of the neural networks. It determines the speed at which the connection weights are updated. If set too low it will cause the weights not to converge to a value necessary to lower the cost function value to zero. If set to a too high of a value the ANN will learn faster but this introduces the risk of the correct weight value being skipped due to the adjustment not being fine enough to converge to zero, which causes oscillations and divergence. The weight update strategy does not necessarily have to be the same as equation (4), but it is a commonly used description as most solver algorithms use it or a slightly modified version [16].

Once the entire training stage is finished, the trained neural network is used in the training stage. Training stage consists only of forward propagation. In this stage, the outputs of the dataset and the neural network are compared, but no adjustments are made. This stage determines the quality of the neural network – how correctly does it predict the values stored in the dataset. This is used to determine if the quality of the training was high, and if it was not hyperparameter adjustment is necessary [15].

4. Solvers Description

Solvers are one of the hyperparameters of the ANN. Solver is the algorithm used in the process of backpropagation to calculate the weights of the neural network. Because of them adjusting the weights of the ANN these algorithms play an important role in the final result of the ANN. If the neural network is trained badly, even with good architecture, it will not be capable of determining the correct weights needed to achieve low values of errors.

First solver used in performed research is SGD. SGD is an optimization algorithm designed for unconstrained optimization problems. Unlike some other solvers, SGD approximates the gradient of the neural networks for each individual training sample. The update is weight update is performed based on the equation [18]:

$$\Theta_{new} = \Theta_{old} - \alpha \frac{\partial J(\theta_i \cdot x_i)}{\partial \theta_i}. \quad (5)$$

Note the similarity to the equation (4), with the difference being that SGD updates its value based on each individual sample's cost function (error). This provides a better adjustment of individual weights. Two parameters can be adjusted: momentum, and whether the so called Nesterov momentum modification will be applied. Momentum is a value that further accelerates SGD in the relevant direction ($J(\Theta) \to 0$). This works by adjusting the learning rate when the goal condition of minimizing the value of cost function – when $J(\Theta)$ value is large the momentum parameter increases the value of learning rate and lowers it when value drops. This value helps with lowering oscillations and faster weight convergence. It's introduced into the equation (5) as an additional component v which adjusts the gradient update depending on the overall ANN loss value, as shown in [18]:

$$\Theta_{new} = \Theta_{old} - \alpha \frac{\partial J(\theta_i \cdot x_i)}{\partial \theta_i} + v_t. \quad (6)$$

Where v_t is determined by:

$$v_t = \mu \cdot v_{t-1} - \alpha \nabla J(\Theta), \quad (7)$$

with parameter μ being the momentum parameter controlling the influence of speed v. Starting speed, v_0 equals zero.

Nesterov momentum (also known as Nesterov Accelerated Gradient – NAG) is a technique that adjusts the based on the intermediate position. The intermediate position is calculated based on the current value of the parameter μ, and the current loss is calculated. The gradient is then further adjusted to achieve a lower error. This method, while slower in each step, provides a more accurate weight adjustment which can lead to a faster overall convergence [19].

Adam is a different optimization algorithm, optimized for stochastic objective functions. It adaptively adjusts estimates of lower-order momentums and is designed to be computationally efficient. It adjusts the gradient in equation (4) in the similar manner as SGD does – shown in equation (6), but it uses two momentums and adjusts them differently. The weight in Adam algorithm are updated through [20]:

$$\Theta_{new} = \Theta_{old} - \alpha \cdot \frac{m_t}{\sqrt{v_t} + \Theta'}. \tag{8}$$

The equation Adam uses to adjusts the momentums are given by:

$$m_t = \frac{m_{t-1}}{1-\beta_1^t}, \tag{9}$$

and

$$v_t = (1-\beta_2) \sum_{i=1}^{t} \beta_2^{t-i} \cdot \Theta'^2_t, \tag{10}$$

with t being the current epoch (training iteration) – this means that weights are adjusted using all historical gradient values. Parameters $\beta_1, \beta_2 \in [0,1\rangle$ are constants with determine the exponential decay of the momentum values, or – in simpler terms, determine the speed of change for momentums m, v respectively [20, 21].

AMSGrad is an expansion of the Adam algorithm, in a similar way to NAG for SGD. AMSGrad was designed to help with convergence problems Adam exhibited in some cases, for example when momentum v_t is larger than the gradient Θ'. The difference between standard Adam and AMSGrad is that it stores the maximum historical average of the second momentum v_t , up to the current step and uses that value to normalize the gradient. With this, the learning rate change is stopped from increasing which results in easier convergence [21].

Both algorithms have default parameters, which are commonly used in research. Default SGD parameters are [22]: $\alpha = 0.01$, $\mu = 0.0$, and NAG not used. The default parameters for Adam are [22]: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and AMSGrad not used.

5. *Parameter choice*

Hyperparameters of the neural network are those values that determine its architecture. In presented research previously determined parameters are shown in Table 2. Values for each hidden layer (HL) are combined and written that the left most value is the hidden layer nearest the input.

Table 2. *Predefined hyperparameters of the used ANN.*

Hyperparameter	Value
Number of HL	5
Neuron per HL	80,65,25.75.80
Activations per HL	Sigmoid, Tanh, ReLU, ReLU, ReLU
Epochs	13

These parameters are fixed and variations are introduced via a modification of solver parameters. Parameters varied for each solver, along with their possible values are presented in Table 3. for SGD and Adam solver.

Table 3. *Parameters used in the Grid Search for each of the optimized solvers*

Solver	Parameter	Possible Values	Count
Both	α	0.001, 0.005, 0.1, 0.01, 0.2, 0.02, 0.1, 0.5, 0.05, 0.5	10
SGD	μ	0.0, 0.02, 0.005, 0.1, 0.01, 0.5, 0.9	7
	NAG	True, False	2
Adam	β_1	0.9, 0.95, 0.85, 0.75, 0.8, 0.99	6
	β_2	0.99, 0.95, 0.9, 0.8, 0.75, 0.85	6
	AMSGrad	True, False	2

The technique used for testing out parameter combinations is grid search. In grid search all the possible combinations of values are created. For this paper, with the values listed in Table 3. the total number of combinations for SGD solver is 140 and 720 for the Adam solver.

The quality of the trained neural networks is determined through the use of root mean square error (RMSE), defined with:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y_i})}{n}}. \tag{11}$$

The solution with a lower RMSE is considered a better solution, due to having a lower regression error compared to the solution with a higher RMSE value. Each parameter combination is executed three times, to avoid poor results in a single run due to randomization of initial weights, and the best solutions are then executed 20 additional times to achieve statistical significance.

6. *Results*

SGD algorithm has achieved the best results with following parameters:

- $\alpha = 0.05$,
- $\mu = 0.02$, and
- NAG used.

Over the course of 20 runs the average RMSE for given parameters was 4.275 [MW] with standard deviation of 0.08369.

Adam algorithm has achieved the best results with the following parameters:

- $\alpha = 0.001$,
- $\beta_1 = 0.95$,
- $\beta_2 = 0.99$, and
- AMSGrad not used.

Over the course of 20 runs the average RMSE with given parameters was 4.259 with the standard deviation of: 0.046395.

The achieved results show the average improvement over the results from the previous research of 0.03 [MW] for SGD and 0.046 [MW] for Adam.

7. *Conclusion*

Obtained results show only the slight improvement over the results previously obtained. While there is improvement, it is questionable whether the solver parameter adjustment beyond the default values is strictly necessary, considering the added complexity and the time needed to test all the combinations. It is also apparent that the values of best parameters are close to the default values for both algorithms. Still, if higher precision is needed the results point towards the fact that solver parameter adjustment can provide that. Further tests on different datasets are necessary to test the effect of solver parameter changes. Future work could also include the comparison between various implementations of solver algorithms, as different implementations could demonstrate different results (Keras framework has been used in this research). Furthermore, the difference in training times with different solver algorithms is apparent and further research in optimizing the parameters with the goal of faster training could be performed.

8. *Acknowledgment*

This research has been supported by the Croatian Science Foundation under the project IP-2018-01-3739, CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), University of Rijeka scientific grant uniri-tehnic-18-275-1447, University of Rijeka scientific grant uniri-tehnic-18-18-1146 and University of Rijeka scientific grant uniri-tehnic-18-14.

Dataset used in research was obtained from the paper Tüfekci, P. (2014) "*Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods*" [23].

9. Nomenclature

Abbreviations	
CCPP	Combined Cycle Power Plant
GT	Gas Turbine
ST	Steam Turbine
LP	Low Pressure
HP	High Pressure
AI	Artificial Intelligence
MLP	Multilayer Perceptron
ANN	Artificial Neural Network
EC	Evolutionary Computing
SGD	Stochastic Gradient Descent
NAG	Nesterov Adaptive Gradient
RMSE	Root Mean Square Error
HL	Hidden Layer

Latin Symbols		m_t	Adam momentum 1
x	Individual input	v_t	Adam momentum 2
y	Expected ANN output	**Greek Symbols**	
$\hat{y}$	Actual ANN output	μ	SGD v parameter
X	Input Vector	β_1	m_t decay
t	Epoch	β_2	v_t decay
n	Number of samples	Θ	Weights vector
J	Cost Function/Loss	θ	Individual weight
$\mathcal{F}$	Activation Function	Θ'	Gradient
v	SGD momentum	α	Learning Rate

10. References

[1] Mrzljak, V., Poljak, I., Orović, J., & Prpić-Oršić, J. (2019, January). Numerical analysis of real open cycle gas turbine. In *INTERNATIONAL SCIENTIFIC CONFERENCE HIGH TECHNOLOGIES. BUSINESS. SOCIETY 2019*.

[2] Mrzljak, V., Anđelić, N., Lorencin, I., & Car, Z. (2019). Analysis of Gas Turbine Operation before and after Major Maintenance. *Pomorski zbornik*, *57*(1), 57-70.

[3] Glazar, V., Mrzljak, V., & Gubic, T., (2019) Thermodynamic Analysis of Combined Cycle Power Plant. *14th International Conference Heat Transfer, Fluid Mechanics and Thermodynamics, 355-341*

[4] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Genetic Algorithm Approach to Design of Multi-Layer Perceptron for Combined Cycle Power Plant Electrical Power Output Estimation. *Energies*, *12*(22), 4352.

[5] Lorencin I., Car Z., Kudlaček J., Mrzljak V., Anđelić N, & Blažević S. (2019). Estimation of Combined Cycle Power Plant Power Output Using Multilayer Perceptron Variations, *10th International Technical Conference - Technological Forum 2019 – Proceedings, 94-98*

[6] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Marine Objects Recognition Using Convolutional Neural Networks. *NAŠE MORE: znanstveno-stručni časopis za more i pomorstvo*, *66*(3), 112-119.

[7] Baressi Šegota, S., Anđelić, N., Kudláček, J., & Čep, R. (2019). Artificial neural network for predicting values of residuary resistance per unit weight of displacement. *Pomorski zbornik*, *57*(1), 9-22.

[8] Anđelić, N., Blažević, S., & Car, Z. (2018, January). Trajectroy Planning Using Genetic Algorithm For Three Joints Robot Manipulator. In *International Conference on Innovative Technologies, IN-TECH 2018*.

[9] Blažević, S., Anđelić, N., & Car, Z. (2017, January). Research of Unstable Behavior of Iterative Path Planning Algorithm for Robot Manipulator. In *IN-TECH 2017 International Conference on Innovative Technologies*.

[10] Lorencin, I., Anđelić, N., Španjol, J., & Car, Z. (2020). Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis. *Artificial Intelligence in Medicine*, *102*, 101746.

[11] Baressi Šegota, S., Lorencin, I., Ohkura, K., & Car, Z. (2019). On the Traveling Salesman Problem in Nautical Environments: an Evolutionary Computing Approach to Optimization of Tourist Route Paths in Medulin, Croatia. *Pomorski zbornik*, *57*(1), 71-87.

[12] Mrzljak, V., Blecich, P., Anđelić, N., & Lorencin, I. (2019). Energy and Exergy Analyses of Forced Draft Fan for Marine Steam Propulsion System during Load Change. *Journal of Marine Science and Engineering*, *7*(11), 381.

[13] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Exergy analysis of marine steam turbine labyrinth (gland) seals. *Pomorstvo*, *33*(1), 76-83.

[14] Mrzljak, V., Car, Z., Kudláček, J., Anđelić, N., Lorencin, I., & Blažević, S. (2019) Analysis of Two Methods For Steam Turbine Developed Power Calculation In Industry 4.0. *10th International Technical Conference - Technological Forum 2019 – Proceedings, 103-110*

[15] Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1, No. 10). New York: Springer series in statistics.

[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[17] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[18] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Physica-Verlag HD.

[19] Dozat, Timothy. Incorporating nesterov momentum into adam. (2016). In *Proceedings of 2016 International Conference of Learning Representations (pp. 1-4)*

[20] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[21] Reddi, S. J., Kale, S., & Kumar, S. (2019). On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

[22] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

[23] Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, *60*, 126-140.