Random Search Hyperparameter Optimization for BPNN to Forecasting Cattle Population

Bain Khusnul Khotimah[1*], *Fitri* Agustina[2], *Oktavia Rahayu* Puspitarini[3], *Husni*[1], *Devie Rosa* Anamisa[1], *Natasha* Prayugo[1] and *Aisyah Meta Sari* Putri[1]

[1]Department of Informatics Engineering, Faculty of Engineering, Universitas Trunojoyo Madura, Bangkalan, Indonesia
[2]Department of Industrial Engineering, Faculty of Engineering, Universitas Trunojoyo Madura, Bangkalan, Indonesia
[3]Departmen of Animal Husbandry, Faculty of Animal Husbandry, Universitas Islam Malang, Malang, Indonesia

Abstract. Backpropagation Neural Network (BPNN) is a suitable method for predicting the future. It has weaknesses, namely poor convergence speed and instability, requiring parameter tuning to overcome speed problems, and having a high bias. This research uses the Random Search hyperparameter technique to optimize BPNN to automatically select the number of hidden layers, learning rate, and momentum. The added accuracy of momentum will speed up the training process, produce predictions with better accuracy, and determine the best architectural model from a series of faster training processes with low bias. This research will predict the local Indonesian cattle population, which is widely developed by people in the eastern part, especially Madura, in 4 types of cattle: sono cattle, karapan cattle, mixed cattle, and breeder cattle. The results of BPNN hyperparameter measurements with the best model show that hyperparameter optimization did not experience overfitting and experienced an increase in accuracy of 2.5% compared to the Neural Network model without hyperparameter optimization. Based on the test results, the BPNN algorithm parameters with a data ratio of 70:30, the best architecture for backpropagation momentum is 6-6-1, with a learning rate of 0.002, momentum 0.3, which has an MSE during testing of 0.1176 on Karapan type Madurese cattle. Tests based on computing time measurements show that the BPNN hyperparameter algorithm stops at 490 iterations compared to regular BPNN. The research results show that the hidden layers, learning rate, and momentum if optimized simultaneously, have a significant influence in preventing overfitting, increasing accuracy, and having better execution times than without optimization.

Keywords: Hyperparameters, BPNN, Random Search, Prediction, Population Number, Madurese Cattle .

1 Introduction

Madurese cattle are one of Indonesia's local cattle breeds. This cow has several characteristics, such as a light brown to dark brown body color, black hooves, snout, fine hair around the mouth, and relatively long legs. Apart from that, it is more resistant to the hot climate in the Madura area, is susceptible to parasite attacks, and has good meat quality [1]. As local Indonesian cattle, Madurese cattle are one of Indonesia's genetic resources [2]. The Madurese people select Madurese cattle into four types based on their culture: karapan cattle, local cattle, sonok cattle, and broiler cattle. Karapan cattle are bulls that can run fast, are agile, have hard-working skeletal muscles, and are emotional. Sonok cattle are female cattle that are docile, obedient, and beautiful, while cattle and beef cattle are slaughtered for their meat and sold to consumers. Madura cattle have several advantages, namely high adaptability to heat and disease resistance, utilizing low-quality feed, having better reproductive performance compared to cross-breed cattle, and consumers much prefer their meat [3]. So, the government's role is significant for the regions to continue to implement and improve outreach programs coaching for farmers regarding issues of feed technology for ruminant livestock and beef cattle trade systems to accelerate economic development, especially in Madurese cattle farming. The development of the cattle population is used as a supporting role in improving the quality of human resources as a provider of nutritious food sources to create a healthy and productive society, as envisioned by the vision and mission of livestock development [4].

Machine learning methods are the most common methods for predicting things. The machine learning method uses forecasting data samples and looks at the data movement patterns of the development of the time series data using an algorithm that can learn ways. Machine learning has various types of methods, for example, intelligent systems and artificial neural networks (ANN) [5]. Backpropagation modeling falls into the Multilayer perceptron category. The

* Corresponding author: bain@trunojoyo.ac.id

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

development of the backpropagation algorithm is by adding momentum. Momentum in a neural network is a weight change based on the gradient direction of the last and previous patterns. Adding momentum parameters to the three artificial neural networks aims to speed up the learning process towards convergence [6]. Experimental results have also shown that this method can make the network converge quickly and stably. This situation is caused because momentum allows the network to adjust weights drastically if the results are changed in the same direction in several patterns, so training becomes faster [7].

Population forecasting using artificial neural network (ANN) models compared to traditional cohort component methods (CCM) for regions in Alabama, USA., which has diverse population and socio-economic characteristics. ANN has produced errors with the right level of precision for regional mapping recommendations [8]. This research is about forecasting population growth using a comparison of artificial neural networks and regression techniques, this shows that artificial neural networks are a good tool for predicting growth models [9].

Neural Network research has also been carried out by [10], where this research carried out parameter optimization in Neural Networks using Genetic Algorithms. This research shows that the combination of Neural Network-Genetic Algorithms is better than just using Neural Networks alone. The hyperparameter configuration that produces the best performance is known as hyperparameter optimization. Tuning hyperparameters can affect the classification algorithm's performance, making this step important [11]. Over the last few years, various algorithms have been developed to optimize hyperparameter values, including Random Search, Grid Search, and model-based approaches using Random Forest, and sequential model-based optimization [12]. Sequential model-based optimization is the best strategy for optimizing hyperparameters because it involves a probabilistic model of the data to determine the most promising points to evaluate. One of the sequential model-based optimization algorithms is Bayesian Optimization [13]. Research related to hyperparameter optimization using Bayesian Optimization was carried out on the Gradient Boosted Trees algorithm by [14]. This research showed that hyperparameter optimization using Random Search for machine learning models experienced an increase in accuracy compared to not using hyperparameter optimization. According to [15-16], some factors influence the performance of the Backpropagation algorithm, namely hidden layers, learning rate, and momentum, where these factors are hyperparameters in the neural network. Because these hyperparameters are an essential factor in backpropagation algorithm training, in building a classification model for Neural Networks in this research, these hyperparameters were optimized [17-18].

Hyperparameters with tuning techniques using the random search method have the advantage of this method being a straightforward approach [16]. However, this can be a drawback because experimenting with all combinations will take time when handling large-dimensional data. It would be very suitable to use this algorithm for data that develops population numbers over several years and does not have large dimensions. It is hoped that applying this method can reduce the model's error value on manageable data [19-20].

This research predicts the population of Madurese cattle using BPNN with the addition of momentum and hyperparameter tuning. To determine the level of accuracy of the artificial neural network system that has been created in predicting the number of developments of Madurese cattle in a specific year. In the first scenario of this research, different error targets will produce other numbers of iterations. The smaller the target error, the greater the number of iterations. Second, the smaller the target error, the accuracy value tends to be better (larger). Third, the number of iterations in the training process with added momentum is more diminutive than without. Based on previous research, the research has predicted the development of the Madurese cattle population using the backpropagation method with the addition of momentum. It is hoped that the prediction results will prepare for food security and prevent the extinction of Madurese cattle.

2 Methods

2.1 Algoritma momentum *backpropagation*

A neural Network (NN) is a network of a group of small processing units modeled based on human neural networks. NN is an adaptive system that can change its structure to solve problems based on external and internal information flowing through the web. In simple terms, NN is a non-linear statistical data modeling tool. NN can be used to model complex relationships between input and output to find patterns in data [11][21]. BPNN is a method developed by adding hidden for the backward process in NN. The Momentum Backpropagation algorithm has steps similar to the conventional one, but the difference lies in the reversal stage (backward propagation) [22-24]. The Momentum Backpropagation algorithm is an extension of the traditional Backpropagation algorithm, where in the learning process, it utilizes the concept of momentum, with a momentum constant value in the range from 0 to 1, to reduce conventional BPNN iterations [8]. BPNN steps are divided into two stages, namely, feedforward and backward propagation [25].

The feedforward step used for the initial direction produces the following initial output:

1. Start by setting an initial weight from 0 to 1.
2. Determine the maximum values for Epoch, Learning rate (α), Momentum, and Target Error.
3. Next, repeat steps 4 to 8 during the (Epoch < maximum epoch) or (MSE > Target Error) condition.
4. Execute steps 6 through 10 during the training process.
5. For the testing process, perform steps 6 to 7.

   Forward Propagation (feedforward)

6. Addition of weights in the hidden layer.

$$z_{-in(j)} = v_{0j} + \sum_{i=1}^{n} x_i . v_{ij} \quad (1)$$

With:
$z_{-in(j)}$ = Total input signal at unit layer j
v_{0j} = Input bias weights of unit 0 and unit layer j
x_i = Input value in unit i
v_{ij} = The weight between input unit i and layer unit j

7. Perform activation operations with weighted addition with the sigmoid activation function.

$$z_j = \frac{1}{1 + e^{-z\_in(j)}} \quad (2)$$

With:
z_j = Output at unit layer j
$z_{in(j)}$ = Total input signal at unit layer j

8. Calculates the output layer value

$$y_{\_ink(k)} = w_{0k} + \sum_{j=1}^{n} z_j . w_{jk} \quad (3)$$

With:
$y_{\_ink(k)}$= Total input signal at the output of k units
w_{0k} = Input bias weights unit 0 and unit layer k
z_j = Input value at unit layer j
w_{jk} = The weight between unit j's layer and unit k as output

9. Calculating the output signal with the activation function.

$$y_k = \frac{1}{1 + e^{-y\_in(k)}} \quad (4)$$

With:
y_k= Output at unit layer k
$y\_in(k)$= Total input signal at the output of k units

Backward propagation

1. Calculates the error value based on the error value and the target value.

$$\delta_k = (t_k - y_k) \quad (5)$$

With:
δ_k = error factor in the output of unit k
y_k = Output at unit layer k
t_k = Target

2. Calculate the weight correction value w_{jk}:

$$\Delta w_{jk}(t) = \alpha\delta_k z_j + \Delta w_{jk}(t-1)\mu \quad (6)$$

With:
$\Delta w_{jk}(t)$= Correct the weight of unit j and output k in iteration t
α = *Learning rate*
δ_k= error factor in the output of unit k
z_j = Output at unit layer j
$\Delta w_{jk}(t-1)$= Correction of weights in the previous iteration
μ = Momentum

3. Calculate the bias correction value w_{0k}:

$$\Delta w_{0k}(t) = \alpha\delta_k + \Delta w_{0k}(t-1)\mu \quad (7)$$

With:
$\Delta w_{0k}(t)$ = Correction of unit bias weight 0 and output k at iteration t
α = *Learning rate*
δ_k= error factor in the output of unit k
z_j= Output at unit layer j
$\Delta w_{0k}(t-1)$= Bias correction in the previous iteration
μ = Momentum

4. Addition of unit deltas for each hidden layer neuron

$$\delta_{in\,j} = \sum_{k=1}^{m} \delta_k\, w_{jk} \quad (8)$$

With:
$\delta_{in\,j}$= The total sum of weighted unit delta
δ_k = Error factor in the output of unit k
w_{jk} = The weight between unit j's layer and unit k's output

5. Calculating the error value:

$$\delta_j = \delta_{in\,j}(z_j)(1 - z_j) \quad (9)$$

With:
δ_j = Error value in unit layer j
$\delta_{in\,j}$ = Error factor in the output of unit k
z_j = Output at unit layer j

6. Calculate the weight correction value v_{ij}:

$$\Delta v_{ij}(t) = \alpha\delta_j x_i + \Delta v_{ij}(t-1)\mu \quad (10)$$

With:
$\Delta v_{ij}(t)$ = Correct the weight of unit i and output j in iteration t
α = *Learning rate*
δ_j= Error value in unit layer j
x_i= Input value in unit i
$\Delta v_{ij}(t-1)$ = Correction of weights in the previous iteration
μ = Momentum

7. Calculate the bias weight correction value v_{0j}:

$$\Delta \boldsymbol{v_{0j}(t)} = \boldsymbol{\alpha\delta_j} + \Delta \boldsymbol{v_{0j}(t-1)\mu} \tag{11}$$

With:
$\Delta v_{oj}(t)$ = Correction of unit bias weight 0 and output j at iteration t
α = *Learning rate*
δ_j = Error value in unit layer j
$\Delta v_{0j}(t-1)$ = Correction of bias weights in previous iterations
μ = Momentum

8. Improvement of weight values and bias values in the output layer.

$$\boldsymbol{w_{jk}(new)} = \boldsymbol{w_{jk}(old)} + \Delta \boldsymbol{w_{jk}} \tag{12}$$
$$\boldsymbol{w_{0k}(new)} = \boldsymbol{w_{0k}(old)} + \Delta \boldsymbol{w_{0k}} \tag{13}$$

With:
$w_{jk}(new)$ = Weight between unit j layer and unit k output (new)
$w_{jk}(old)$= Weight between unit j layer and unit k output (old)
Δw_{jk}= Correct the weight of unit j and output k in iteration t
$w_{0k}(new)$ = Input bias weights unit 0 and unit layer k (new)
$w_{0k}(old)$ = Unit input bias weight 0 and unit layer k (old)
Δw_{0k} = Correction of unit bias weight 0 and output k at iteration t

9. Improved weight values and bias values in the hidden layer.

$$\boldsymbol{v_{ij}(new)} = \boldsymbol{v_{ij}(old)} + \Delta \boldsymbol{v_{ij}} \tag{14}$$
$$\boldsymbol{v_{0j}(new)} = \boldsymbol{v_{0j}(old)} + \Delta \boldsymbol{v_{0j}} \tag{15}$$

With:
$v_{ij}(new)$ = Weight between input unit i and layer unit j (new)
$v_{ij}(old)$= Weight between input unit i and layer unit j (old)
Δv_{ij}= Correct the weight of unit i and output j in iteration t
$v_{0j}(new)$ = Unit input bias weight 0 and output j (new)
$v_{0j}(old)$ = Unit bias weight 0 and output j (old)
Δv_{0j} = Correction of unit bias weight 0 and output j

2.2 Random Search

Random Search (RS) is very suitable in high dimensions because the desired function has a low practical dimension and is more sensitive to changes in some dimensions than others [26]. In particular, if a function f of two variables can be approximated by another function of one variable (f(x1, x2) ≈g(x1)), it can be said that f has a low practical dimension [27]. Figure 1 illustrates how grid points and uniform collections of random points differ in dealing with standard adequate dimensions.

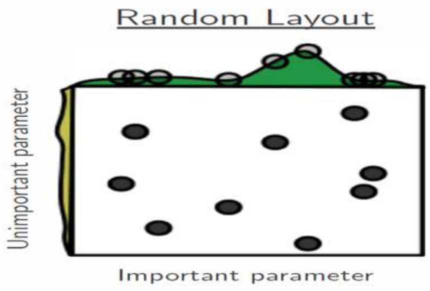

Figure 1. Illustration random search of n parametaers for optimizing [28].

Fig. 1. Show for using random search of n experiments to optimize the function f(x, y) = g(x) + h(y) ≈ g(x) with low adequate dimensions [29]. Above each grid, g(x) is shown in green, and on the left of each box, h(y) as the objective function and n as trials test g(x) in different places. With random search, experiments continuously explore different values grid until the appropriate parameter values are obtained [30-31]. The parameters used for tuning parameters in BPNN are shown in Table 1.

Table 1. The random search algorithm.

Parameter	Algorithm BPNN
Number of hidden layers	3,6,8,10,12,24
Number of Nodes	3,6,8,10,12
Learning Rate	0.001-0.1
Momentum	0.001-0.1

2.3 Data Preprocessing

Data preprocessing is carried out by imputation and normalization to produce data ready as classification input. This process is carried out to eliminate missing values with a value obtained from the dataset. Data Interpolation is an action that occurs in the pre-processing stage. This stage fills in empty data by using the average value of the data before and after the open data [8].

$$\boldsymbol{x_n} = \frac{\boldsymbol{x_{n-1}} + \boldsymbol{x_{n+1}}}{\boldsymbol{2}} \tag{16}$$

So, normalization is done for transformation process stabilizes the data by changing the data into a scale form with a value range from 0 to 1, often called scalar min-max. In the normalization flow, it is carried out after imputation of data to overcome missing values, so that complete data is obtained [10].

$$\boldsymbol{x'} = \frac{\boldsymbol{x} - \boldsymbol{x_{min}}}{\boldsymbol{x_{max}} - \boldsymbol{x_{min}}} \tag{17}$$

The normalization process uses the actual data minus the most minor data divided by the highest data minus the most minor data to produce data ranging from 0 – 1 [9][10].

3 Result and Discussion

3.1 Hyperparameter and architecture for BPNN performance

In this section, we apply GS hyperparameters to investigate the performance of BPNN by measuring the performance of MSE to assess the effectiveness of all models for forecasting local cattle populations spread across Indonesia's eastern part of the island of Madura. Hyper-parameters with GS by selecting the best parameter, according to Fig. 2. to determine how significant, the variation in prediction results.

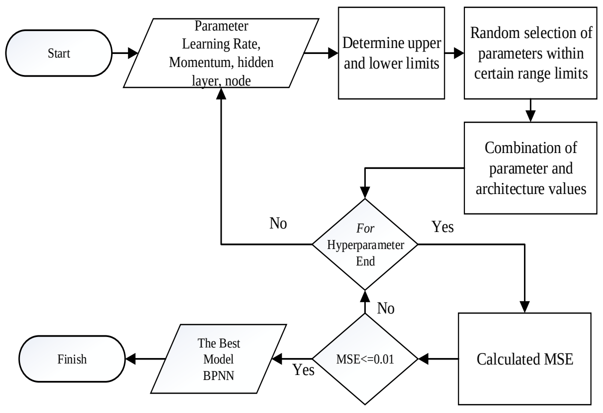
Start
Parameter Learning Rate, Momentum, hidden layer, node
Determine upper and lower limits
Random selection of parameters within certain range limits
Combination of parameter and architecture values
No
For Hyperparameter End
Yes
No
Finish
The Best Model BPNN
Yes
MSE<=0.01
Calculated MSE

Fig. 2. Proposed method for RS hyperparameters in BPNN

Table 2. The effect of learning rate and momentum on MSE in the 4-7-1 architecture.

Learning rate	**Momentum**	**MSE**			
		Sono cattle	**Kerapan cattle**	**Breeder cattle**	**Beef cattle**
0.2	0.6	0.1816	0.1754	0.1913	0.2110
0.02	0.6	0.1909	0.1766	0.1951	0.1839
0.002	0.6	0.1215	0.1285	0.1491	0.1527
0.2	0.3	0.1727	0.1763	0.1843	0.1923
0.02	0.3	0.1761	0.1711	0.1842	0.1836
0.002	0.3	0.1335	0.1392	0.1253	0.1403
0.2	0.9	0.1788	0.1784	0.1893	0.1912
0.02	0.9	0.2123	0.1777	0.2268	0.1867
0.002	0.9	0.1653	0.1707	0.1902	0.1807

RS approach to find the best hyperparameter values with respect to a given value. The Backpropagation Momentum process is carried out at the training stage with a data division of 70%:30% of 1,000 data. The existing data was checked for missing values, and the imputation process was carried out. After obtaining complete data, proceed with min–max normalization so that the data is in the range of 0 to 1. The training stages use the Backpropagation momentum algorithm with the first stage, namely initializing the weight, max epoch, learning rate, and target error, carried out randomly. When the epochs are less than the maximum epoch or the MSE is greater than the target error, it will go to the next stage, but if not, it will immediately produce model output. When the number of iterations is less than the number of datasets, it will proceed to the next stage, namely FeedForward, from the input layer to the hidden layer to the output layer. If the max epoch is 1000, while epoch < max epoch or MSE > target error, at the FeedForward stage from input to the hidden layer to output, adding up the weight values in the hidden layer. For the sigmoid activation function, the results will be obtained from the output signal from the hidden neuron unit. At each layer, the output will add up the signals from the layer in the hidden unit with weights and biases. The sigmoid activation function is performed to calculate the output signal from the unit at the output. Next, calculate the error value.

Table 3. The Effect of learning rate and momentum on MSE in the 6-6-1 architecture.

Learning rate	**Momentum**	**MSE**			
		Sono cattle	**Kerapan cattle**	**Breeder cattle**	**Beef cattle**
0.2	0.6	0.1781	0.1792	0.1877	0.1791
0.02	0.6	0.1781	0.1719	0.1975	0.1855
0.002	0.6	0.1176	0.1454	0.1511	0.1573
0.2	0.3	0.1732	0.1817	0.1781	0.1945
0.02	0.3	0.1629	0.1726	0.1719	0.1851
0.002	0.3	0.1298	0.1413	0.1212	0.1447
0.2	0.9	0.1586	0.1804	0.1669	0.1956
0.02	0.9	0.2085	0.1779	0.2190	0.1922
0.002	0.9	0.1714	0.1620	0.1741	0.1789

The first experiment was carried out on the 4-7-1 architecture with different combinations of learning rate (0.2, 0.02, 0.002) and momentum (0.6, 0.3, 0.9). In the experiment conducted on Sono cattle data, the best performance was in the model with learning rate = 0.002 and momentum = 0.6 and the model with learning rate = 0.002 with momentum = 0.3. Both models have the same performance in MSE = 0.09039. In kerapan cattle, the best performance lies in the model with a learning rate = 0.002 and momentum = 0.3. The model has a performance of MSE = 0.09104. The best broiler performance lies in the model with a learning rate = 0.002 with a momentum of 0.3. The model has a performance of MSE = 0.09421. In beef cattle, the best performance lies in the model with learning rate = 0.002 and momentum = 0.3. The model has a performance of MSE = 0.08875. From the trials carried out on four data, the best model for testing on the 4-7-1 architecture is the model with a learning rate = 0.002 and momentum = 0.3.

Table 4. The Effect of learning rate and momentum on MSE in the 8-13-1 architecture.

Learning rate	**Momentum**	**MSE**			
		Sono cattle	**Kerapan cattle**	**Breeder cattle**	**Beef cattle**
0.2	0.6	0.1884	0.1721	0.1963	0.2365
0.02	0.6	0.1860	0.1700	0.1952	0.1853
0.002	0.6	0.1288	0.1383	0.1337	0.1561
0.2	0.3	0.1778	0.1730	0.1897	0.1995
0.02	0.3	0.1780	0.1617	0.1844	0.1738
0.002	0.3	0.1183	0.1261	0.1363	0.1472
0.2	0.9	0.1991	0.1694	0.2069	0.2898
0.02	0.9	0.1998	0.1677	0.2073	0.1880
0.002	0.9	0.1824	0.1749	0.2016	0.1781

The second experiment was carried out on the 6-6-1 architecture with different combinations of learning rate (0.2, 0.02, 0.002) and momentum (0.6, 0.3, 0.9). The model has a performance of MSE = 0.1176 far sono cattle, the best performance lies in the model with learning rate = 0.002 and momentum = 0.6. The higher the momentum, the lower the MSE results. Momentum will make the backward weighting iterations increase but in general it can increase the performance of the algorithm.

The architecture BPNN on the 8-13-1 with different combinations of learning rate (0.2, 0.02, 0.002) and momentum (0.6, 0.3, 0.9). In experiments carried out, the best performance lies in the model with learning rate = 0.002 and momentum = 0.6. The model has a performance MSE = 0.1288 for sono cattle.

3.2 Analysis

This research has carried out non-momentum BPNN training and additional Momentum BPNN using three architectures, four learning rates, and three momentums on data on different types of livestock. It aims to analyze and observe performance architectures that consistently produce better-performing models. Apart from that, this research varies the learning rate, non-momentum, and momentum of each architecture to keep the effect of execution time on the best model. At this training stage, the author carried out 1,000 epochs on each of the best architectural models to determine BPNN performance. Hyperparameters with Random search are used to choose the most suitable architecture and calculate the execution time.

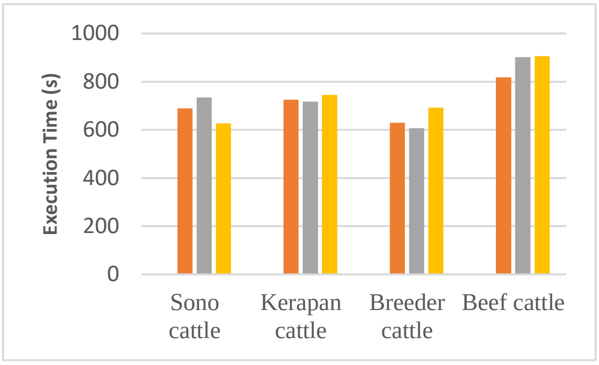

Fig. 3. Best execution time of BPNN in momentum with architecture 4-7-1.

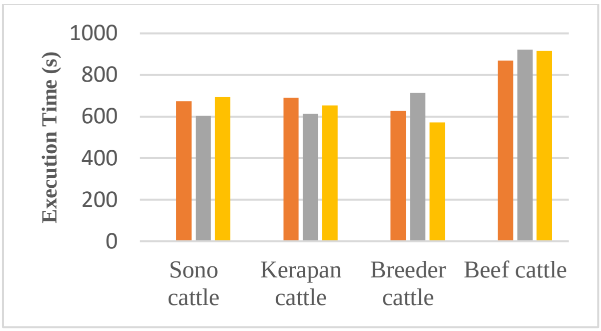

Fig. 4. Best execution time of BPNN in momentum with architecture (6-6-1).

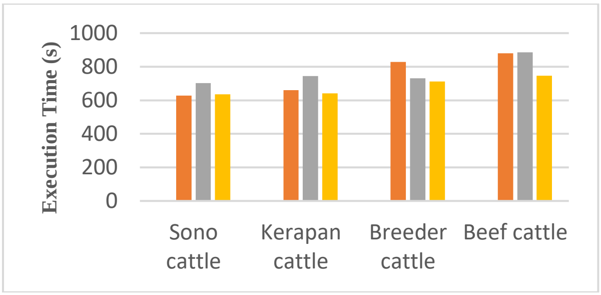

Fig. 5. Longest execution time of BPNN with architecture BPNN in momentum (8-13-1)

Backpropagation in momentum is tied to the previous iteration to calculate weight changes involving velocity. The testing with execution time on 6-6-1 architecture with momentum = 0.002 for 600 seconds. What affects this is that the model needs help to find patterns from various datasets, so other architectures also take a long time. The more hidden layers, the longer the execution time, and the backward phase does this in that it updates the weights without tying up the constructs from the previous iteration. The MSE obtained is slightly more significant than the Backpropagation momentum. The drilled model has difficulty reaching the minimum value, which causes the acceleration of convergence to be slow. When comparing training times Backpropagation momentum is longest when execution occurs on the 8-13-1 architecture, when learning rate = 0.2. because more complex architectures require more iterations.

The comparison of non-momentum BPNN training time for the best 4-7-1 architecture, with the effect of changing learning rate on training runtime. When the learning rate = 0.2, it has the fastest time, namely 469 s for Beef Cattle, whereas if the learning rate = 0.002, it has the longest time, namely 900 s for Breeder Cattle. The average time for all executions is 716s, meaning that BPNN has an uncertain time to get the optimum value, so its convergence is slower. The time will increase slightly faster. For example, the learning rate is reduced to 0.002, so the computational model is faster, and the computation takes longer.

Table 5. Execution time is based on learning rate BPNN in momentum.

Data	**Execution time is based on Learning Rate (s)**		
	0.002	0.02	0.2
Sono cattle	560	845	692
Kerapan cattle	634	617	507
Breeder cattle	900	625	730
Beef cattle	880	752	680

Furthermore, testing based on execution time in Backpropagation adds momentum. The higher the momentum value produces longer iterations because of the role of momentum in calculating weight updates. Apart from that, saving the previous correction weights in the weight update settings is one of the causes of the extended computing time in each iteration. Selection of the best time-based architecture, namely 6-6-1, produces good time-based testing with added momentum, according to Table 5.

Adding momentum is part of deep learning in BPNN to produce low bias. The formation of the best model in this research uses several variations on hidden neurons, activation functions, and automatic selection of parameters with random search. Random search chooses gradient descent with the best momentum and adaptive learning rate d at 0.2 for BPNN l, and the activation function used is sigmoid.

4 Conclusion

Hyperparameter Backpropagation Neural Network (BPNN) using random search to optimize the model architecture, learning rate, and momentum, with testing conducted epoch of 1,000. Tuning parameters based on learning rate and momentum on the performance of BPNN in momentum is in the range of learning rate 0.2, 0.02, 0.002 has produced the best learning performance in this model is 0.002. The best momentum parameter tuning with values of 0.3, 0.6, and 0.9 makes the best MSE value at the lowest momentum of 0.3.

The effect of the best architecture model on Backpropagation adding the best momentum to the 6-6-1 architecture, which has an MSE at the time of testing of 0.3020 in sono cattle, 0.0435 in race cattle, 0.3725 in Breeder cattle, and 0.0041 in the combination.

From the trials carried out during training, it is found that Backpropagation momentum is faster when compared to Backpropagation without momentum with a time ratio of 507 s. Backpropagation momentum can accelerate convergence to the target error.

The authors thank LPPM Trunojoyo University for facilitating and funding this research with the National Collaborative Research scheme with the main contract number 5869/UN46.4.1/PT.01.03/2023. The author would also like to thank the Faculty of Animal Husbandry of the Islamic University of Malang for collaborating with the Team by assisting with validation and data analysis in conducting the research until completion.

References

1. R. Benzer, Population dynamics forecasting using artificial neural networks, January 2015, Fresenius environmental bulletin, **24**, 2 (2015).
2. V. Riiman, A. Wilson, P. Pirkelbauer, Comparing Artificial Neural Network and Cohort-Component Models for Population Forecasts, Published in Population, Economics, Review, 22 October 2019
3. N Widyas, T. S. M. Widi, E. Baliarti, Predicting Madura cattle growth curve using non-linear model, IOP Conf. Ser.: Earth Environ. Sci. **142**, 012006, (2018).
4. U. Paputungan, M. J. Hendrik, W. Utiah, Predicting live weight of Indonesian Local-Bali cattle using body volume formula, Livestock Research for Rural Development. Volume 30, 2018.
5. P. Alkhairi, E. R. Batubara, R. Rosnelly, W. Wanayaumini, and H. S. Tambunan, Effect of Gradien Descent With Momentum *Backpropagation* Training Function in Detecting Alphabet Letters, *Sinkron :* Jurnal Penelitian Teknik Informatika, **8**, 574–583 (2023).
6. T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna, *A next-generation hyperparameter optimization framework,* in Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019).
7. F. R. Ramadani, Inggih Permana, M. Afdal, and Siti Monalisa, Model for Estimating Waste Generation in Pekanbaru Using Backpropagation Algorithm, J. INFORMATICS Telecommun. Eng., **7**, pp. 317–327, (2023).
8. M. G. M. Abdolrasol et al., Artificial neural networks based optimization techniques: A review, Electron, **10,** 21 (2021). https://doi.org/10.3390/electronics10212689
9. K. M. R. Alam, N. Siddique, and H. Adeli, A dynamic ensemble learning algorithmfor neural networks, Neural Comput. Appl., **32**, 2, pp. 8675–8690, (2020).

10. A. Zheng. Chapter 4: Hyperparameter tuning, In: Evaluating Machine Learning Models. USA: O'Reilly Media, Inc., (2015).
11. M. Feurer and F. Hutter, Hyperparameter optimization, pp. 3–33, (2019).
12. Y. A. Du, Research on the Route Pricing Optimization Model of the Car-Free Carrier Platform Based on the BP Neural Network Algorithm, Complexity, (2021)
13. K. Adamczyk, D. Zaborski, W. Grzesiak, J. Makulska, W. Jagusiak, Recognition of culling reasons in Polish dairy cows using data mining methods, Comput, Electron, Agric, 26-27 (2016)
14. Lee, D. H.; Lee, S.-H.; Cho, B.-K.; Wakholi, C.; Seo, Y. W.; Cho, S.-H.; Kang, T.-H.; Lee, W.-H. Estimation of carcass weight of Hanwoo (Korean Native Cattle) as a function of body measurements using statistical models and a neural network. Asian-Australas. J. Anim. Sci., **33**, (2020)
15. L. Krpálková, V. E. Cabrera, J. Kvapilík, J. Burdych, P. Crump, Associations between age at first calving, rearing average daily weight gain, herd milk yield and dairy herd production, reproduction, and profitability. J. Dairy Sci., **97** (2014)
16. J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization. J. Mach. Learn. Res., **13**, pp 281–305 (2012)
17. N Widyas, S Prastowo, T S M Widi and E Baliarti, Predicting Madura cattle growth curve using non-linear model, IOP Conf. Series: Earth and Environmental Science, **142** (2018)
18. J. Choi, D. Kim, M.Ko, D. Lee, K. Wi, H. Lee, Co mpressive strength prediction of ternary-blended concrete using deep neural network with tuned hyperparameters, Journal of Building Engineering **75**, 15 September 2023.
19. M. Jin, Q. Liao**,** S. Patil, A. Abdulraheem, D. Al-Shehri, G. Glatz, Hyperparameter Tuning of Artificial Neural Networks for Well Production Estimation Considering the Uncertainty in Initialized Parameters, *ACS Omega*, **7**, pp. 24145−24156, (2022)
20. M. Ahuja, D. P. Mishra, D. Mohanty, H. Agrawal, S. Roy, Development of Empirical and Artificial Neural Network Model for the Prediction of Sorption Time to Assess the Potential of CO2 Sequestration in Coal. *ACS Omega*, **8,** 34, pp. 31480-31492, (2023)
21. Z. S. Kadhim, H. S. Abdullah, K. I. Ghathwan, Artificial Neural Network Hyperparameters Optimization: A Survey, (iJOE), **18**, 15, pp. 59-87, (2022)
22. S. Bansal and A. Kumar, Automatic Deep Neural Network Hyper-Parameter Optimization for Maize Disease Detection, 2021, IOP Conf. Series: Materials Science and Engineering, **1022**, (2021) 012089.
23. L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, Neurocomputing, **415**, pp 295-316 (2020)
24. A. Esmaeili, Z. Ghorrati, E. T. Matson, Agent-Based Collaborative Random Search for Hyperparameter Tuning and Global Function Optimization. Systems, **11**, 228 (2023)
25. F. F. Firdaus, H. A. Nugroho and I. Soesanti, Deep Neural Network with Hyperparameter Tuning for Detection of Heart Disease, *2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, Bandung, Indonesia, pp. 59-65, 2021, doi: 10.1109/APWiMob51111.2021.9435250.
26. I. Jamaleddyn, R. El ayachi, M. Biniz, An improved approach to Arabic news classification based on hyperparameter tuning of machine learning algorithms, Journal of Engineering Research, 11, 2, (2023)
27. L. Wen, X. Ye, L. Gao, A new automatic machine learning based hyperparameter optimization for workpiece quality prediction, Measurement and Control, **53**, 7,), pp.1088–1098, (2020)
28. S. KARMAKAR, G. SHRIVASTAVA, M. K. Kowar, Impact of learning rate and momentum factor in the performance of back-propagation neural network to identify internal dynamics of chaotic motion, Kuwait Journal of Science (KJS), **41**, 2 (2014)
29. K. M. R. Alam, N. Siddique, and H. Adeli, "A dynamic ensemble learning algorithmfor neural networks," Neural Comput. Appl., **32**, 12, pp. 8675–8690 (2020).
30. J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, *Algorithms for hyper-parameter optimization, Adv. Neural Inf. Process*. Syst. 24–25th Annu. Conf. Neural Inf. Process. Syst., NIPS 2011, 1–9, (2011).
31. B. Raharjo, N. Farida, P. Subekti, R. H. S. Siburian, P. D. H. Ardana, and R. Rahim, Optimization Forecasting Using Back-Propagation Algorithm, J. Appl. Eng. Sci., **19**, 1083–1089, (2021).
32. G. I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, and H. Samulowitz, An effective algorithm for hyperparameter optimization of neural networks, IBM J. Res. Dev., **61**, 4, (2017)
33. P. Liashchynskyi and P. Liashchynskyi, Grid search, random search, genetic algorithm: A big comparison for NAS, 2017, pp. 1–11, 2019, [Online]. Available: http://arxiv.org/abs/1912.06059.