IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 14, No. 5, October 2025, pp. 3994~4005
ISSN: 2252-8938, DOI: 10.11591/ijai.v14.i5.pp3994-4005

Improving multilayer perceptron on rainfall data using modified genetics algorithm

Marji[1], Wayan Firdaus Mahmudi[1], Endang Wahyu Handamari[2], Edy Santoso[1], Maulana Muhamad Arifin[2]
[1]Department of Informatics Engineering, Faculty of Computer Science, Brawijaya University, Malang, Indonesia
[2]Department of Mathematics, Faculty of Mathematics and Natural Sciences, Brawijaya University, Malang, Indonesia

Article Info

Article history:

Received Dec 4, 2024
Revised Jul 22, 2025
Accepted Aug 6, 2025

Keywords:

Chromosome selection
Hyperparameter tuning
Modified genetics algorithm
Multilayer perceptron
Rainfall prediction

ABSTRACT

Rainfall prediction is essential for managing water resources, agriculture, and disaster response, particularly in regions affected by climate variability. This study introduces a modified genetic algorithm (MGA) to optimize hyperparameters of a multilayer perceptron (MLP) for rainfall forecasting. The MGA incorporates elitism to retain top-performing solutions and adaptive selection based on model accuracy. The proposed MGA–MLP model was tested on rainfall datasets from Australia and Indonesia (BMKG). Experimental results show that configurations with two hidden layers, rectified linear unit (ReLU) activation and limited-memory Broyden Fletcher Goldfarb Shannon (LBFGS) optimizer, a learning rate of 0.001 and 1000 epochs consistently delivered strong performance. The model achieved accuracies of 86.02% and 79.05%, respectively. These findings indicate that MGA significantly improves MLP performance and provides a reliable, generalizable method for rainfall prediction across diverse climatic conditions.

Corresponding Author:

Marji
Department of Informatics Engineering, Faculty of Computer Science, Brawijaya University
Bhirawa Street, Tegalweru, Dau, Malang, East Java, Indonesia
Email: marji@ub.ac.id

1. INTRODUCTION

Weather is a complex and dynamic atmospheric phenomenon governed by interdependent variables, including temperature, humidity, wind, air pressure, and precipitation. These parameters fluctuate over short time scales and exhibit nonlinear interactions, requiring advanced analytical methods for accurate prediction. As noted by Chen *et al.* [1], machine learning (ML) has emerged as a crucial tool in weather and climate data analysis due to its ability to efficiently process high-dimensional and nonlinear data. Rainfall prediction, in particular, remains a significant challenge due to its high spatiotemporal variability and the compounding effects of climate change [2]. Rainfall processes are deeply intertwined with broader climatic dynamics, including evaporation and condensation cycles, as well as regional atmospheric circulation patterns. The authors in [3], [4] highlighted the critical role of rainfall in sustaining groundwater recharge and supporting long-term hydrological management. In the Indonesian context, climate anomalies have amplified the frequency and severity of hydrological extremes. For example, Gradiyanto *et al.* [5] reported notable disruptions in the hydrological regime of the Kupang River, attributing the shifts to increased rainfall variability in the Greater Pekalongan area.

In response to these challenges, artificial intelligence (AI) has significantly advanced the modeling of precipitation systems. Doost *et al.* [6] demonstrated how K-means clustering can effectively pre-process spatial rainfall patterns to improve model accuracy. Kantharia *et al.* [7] employed an adaptive neuro-fuzzy inference system (ANFIS) that incorporated soil moisture parameters for improved runoff modeling. At the same time, Mahajan and Sharma [8] highlighted the comparative performance of classical ML algorithms in rainfall forecasting. Hybrid approaches are also gaining momentum. For instance, Wang *et al.* [9] compared back-propagation neural network (BPNN), group method of data handling (GMDH), and autoregressive integrated moving average (ARIMA) models coupled with wavelet decomposition for monthly rainfall prediction. Meanwhile, Liao *et al.* [10] introduced a ConvLSTM-SmaAT-UNet framework for short-term rainfall forecasting with promising results. Deep learning has further elevated rainfall modeling capabilities. Hu *et al.* [11] merged multisource precipitation data using ConvLSTM architectures, and Farooq *et al.* [12] integrated meteorological indices with ML models to refine deterministic predictions in Australia. Rahman *et al.* [13] utilized ML fusion to support innovative city rainfall systems, while Fang *et al.* [14] combined long-lead multi-model forecasts into a deep learning structure. Similarly, Li *et al.* [15] developed ensemble learners to improve the accuracy of seasonal rainfall predictions.

The refinement of neural network architectures has also seen significant developments. Necesito *et al.* [16] introduced a univariate deep learning framework for daily rainfall prediction in data-scarce regions. The authors in [17], [18] employed deep generative models for precipitation nowcasting, thereby improving both temporal resolution and spatial accuracy. Ling *et al.* [19] proposed a diffusion-based two-stage forecasting model, while Harris *et al.* [20] leveraged stochastic downscaling to increase the granularity of climate-scale rainfall projections. Feature selection and automation techniques continue to boost forecasting performance. Quintanar *et al.* [21] applied AutoML for predicting the standardized precipitation index, and Wen *et al.* [22] proposed AdaNAS, a neural architecture search method tailored for ensemble rainfall prediction. He *et al.* [23] explored multimodal deep learning frameworks integrating meteorological and environmental features, while authors in [24], [25] demonstrated the effectiveness of ensemble methods in projecting future precipitation scenarios and sub-seasonal forecasts.

Building upon these advancements, this study introduces a modified genetic algorithm (MGA) to optimize the hyperparameters of a multilayer perceptron (MLP) model. The proposed MGA incorporates elite chromosome preservation and adaptive gene selection to accelerate convergence and mitigate premature stagnation. The optimized MLP is evaluated using rainfall datasets from both Australia and Indonesia, allowing for a comparative performance analysis across distinct climatic regions and thereby enhancing the generalizability of rainfall prediction systems.

2. METHOD

2.1. Datasets

This study utilizes two real-world rainfall datasets to assess the performance and generalization capabilities of the proposed model. The first dataset is the Australian rainfall dataset, obtained from Kaggle, which consists of 145,460 daily records from 49 weather stations across Australia. After eliminating missing values, 56,564 valid samples were retained. The dataset includes 22 attributes, of which 17 are numerical. The second dataset was collected by the Meteorology, Climatology, and Geophysics Agency (BMKG) of Indonesia. It initially contained 589,265 daily samples and 12 attributes, 9 of which are numerical. After cleaning, 372,151 samples remained. These datasets represent distinct climatic regions: temperate (Australia) and tropical (Indonesia), providing a comprehensive basis for assessing both model robustness and cross-regional generalization performance [5], [12].

2.2. Data preparation

Both datasets underwent a structured and consistent preprocessing pipeline to prepare them for binary classification tasks. The process began with the removal of incomplete entries to ensure the quality and reliability of the data. Next, only specific weather stations deemed relevant to the study were retained. Following this, all numerical input features were normalized using min–max scaling to bring them onto a comparable scale. Finally, the target variable representing rainfall was encoded into binary labels, aligning with the classification framework employed in this study [2], [6], [8].

For the Australian dataset, sixteen meteorological attributes were selected as input features: MinTemp, MaxTemp, Rainfall, Evaporation, Sunshine, WindGustDir, WindGustSpeed, WindDir9am, WindDir3pm, WindSpeed9am, Pressure3pm, Cloud9am, Cloud3pm, Temp9am, Temp3pm, and RainToday. The output variable, RainTomorrow, was encoded as 1 for "rain" and -1 for "no rain". In the BMKG dataset, eight features were selected to capture the characteristics of tropical climate conditions. The output variable, RR (rainfall amount), was transformed into binary classes, where rainfall amounts exceeding 1 mm were labelled as 1 (rain) and those equal to or below 1 mm were labelled as –1 (no rain). The detailed descriptions of the selected BMKG features are presented in Table 1. All preprocessing steps were implemented using Python with the Pandas and scikit-learn libraries [13], [16].

Table 1. Descriptions of selected features from the BMKG dataset

Feature	Description
Tn	Minimum temperature usually occurs in the early morning
Tx	Maximum temperature, typically occurring in the afternoon.
Tavg	Average of Tn and Tx, indicating general temperature conditions.
RH_avg	Average relative humidity (%), showing moisture in the air.
RR	Rainfall amount (mm), representing total precipitation.
ss	Duration of sunshine (hours) during the observation period.
ff_x	Maximum wind speed (m/s) recorded during the day.
ddd_x	Direction of the maximum wind.
ff_avg	Average wind speed (m/s).

2.3. Data class distribution visualization

To visualize the separation between rainfall classes, the sixteen input features from the Australian dataset were transformed into a single discriminant axis using linear discriminant analysis (LDA). This dimensionality reduction technique preserved the class-separating characteristics while projecting the high-dimensional data onto a one-dimensional space. The transformed data were then plotted in a histogram to illustrate the distribution of the two classes: “rain” and “no rain”. A similar process was conducted for the BMKG dataset, applying LDA to project the selected features onto a one-dimensional axis. The resulting class distributions for both datasets are illustrated in Figure 1 for Australia and Figure 2 for Indonesia [16], [26].

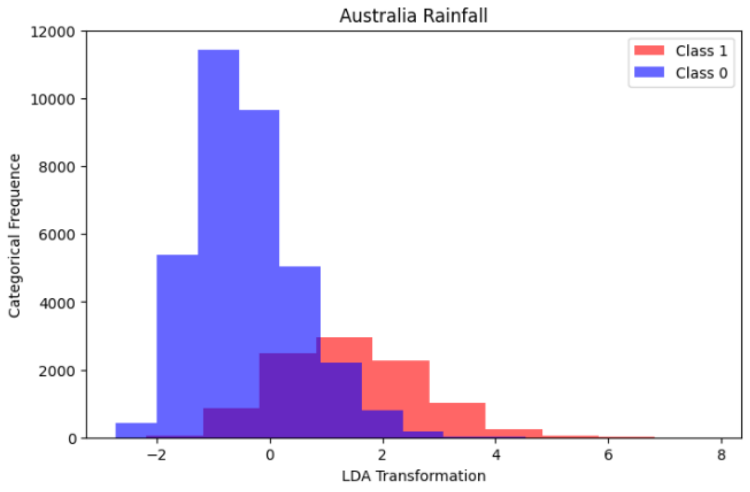

Figure 1. LDA-transformed class distribution for the Australian dataset

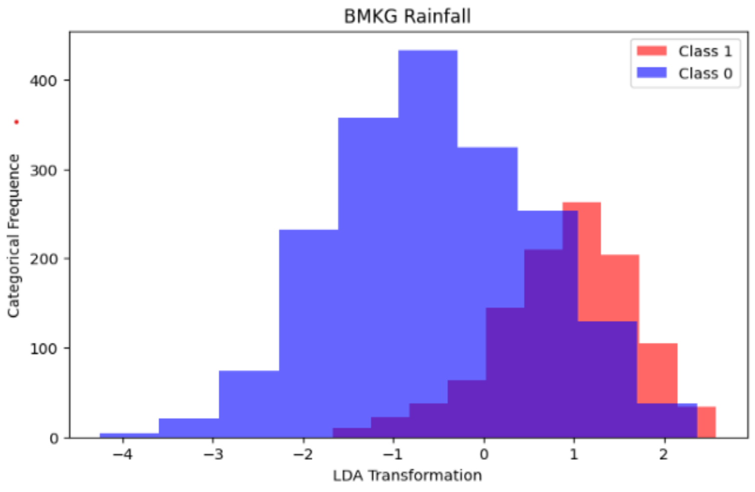

Figure 2. LDA-transformed class distribution for the BMKG dataset

The histograms demonstrate a degree of overlap between the two rainfall classes, particularly in regions near the decision boundary. This overlapping class distribution suggests the potential for misclassification, especially for models that fail to effectively capture nonlinear or high-order feature interactions. Therefore, robust classification strategies and adaptive hyperparameter optimization are crucial for enhancing predictive accuracy. All preprocessing and visualization steps in this phase were implemented using Python with the scikit-learn and Matplotlib libraries. The use of dimensionality reduction for class separability has parallels in recent rainfall distribution studies employing hybrid reduction-clustering techniques [26].

2.4. Schematic of the proposed method

The overall framework of the proposed rainfall prediction system is depicted in Figure 3. It integrates MGA with an MLP to optimize classification performance, which is consistent with recent works highlighting the effectiveness of evolutionary optimization for deep learning models in rainfall forecasting [6], [22]. The flowchart outlines the end-to-end process, beginning with data preprocessing, followed by population initialization and fitness evaluation, and culminating in model convergence through iterative optimization.

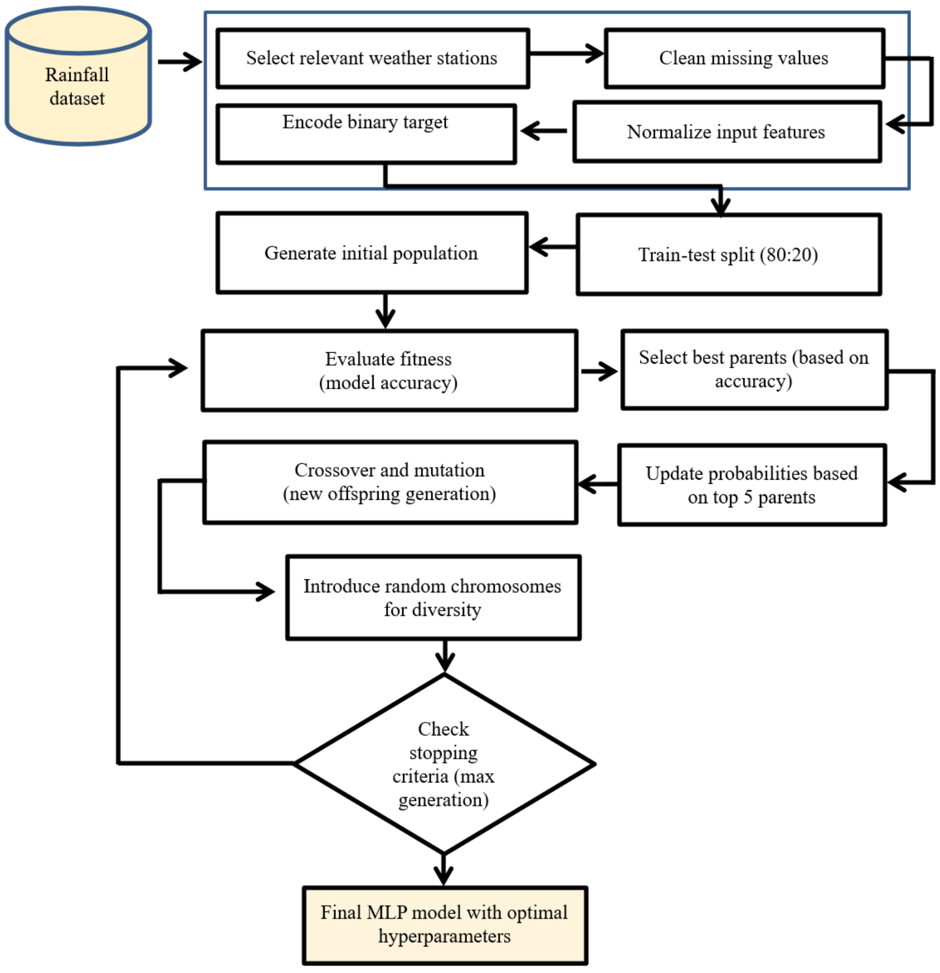
Rainfall dataset
Select relevant weather stations
Clean missing values
Encode binary target
Normalize input features
Generate initial population
Train-test split (80:20)
Evaluate fitness (model accuracy)
Select best parents (based on accuracy)
Crossover and mutation (new offspring generation)
Update probabilities based on top 5 parents
Introduce random chromosomes for diversity
Check stopping criteria (max generation)
Final MLP model with optimal hyperparameters

Figure 3. Workflow of the proposed MGA–MLP-based rainfall prediction framework

The system consists of three primary components that operate interactively throughout the optimization process. The first component is the MLP module, which is responsible for configuring, training, and evaluating the MLP. This module defines the neural network architecture, including the number of hidden layers, the number of neurons per layer, the activation functions, the optimizer type, the learning rate, and the number of training epochs. The model is trained using 80% of the data and tested on the remaining 20%, a common practice in rainfall prediction studies to ensure fair model evaluation [9], [14]. The resulting classification accuracy serves as the fitness value for each chromosome. This component operates during the fitness evaluation phase, corresponding to step 5.

The second component is the hyperparameter encoding module, which encodes each candidate solution (i.e., an MLP configuration) into a structured chromosome format. Each chromosome consists of a sequence of genes, with each gene representing a specific hyperparameter. This encoding enables efficient manipulation through genetic operators, such as crossover and mutation, which are widely applied in rainfall-related optimization frameworks [2], [13]. The encoding and decoding processes are carried out during population initialization (step 3) and fitness evaluation (step 5).

The third component is the MGA optimization module, which forms the core of the evolutionary search mechanism. Once chromosomes are ranked by fitness, the algorithm selects the top-performing individuals as parents. Genetic operators are then applied to produce new offspring. A notable feature of this module is the probability-based update mechanism, which constructs probabilistic distributions of hyperparameter values based on the top five chromosomes in each generation. These distributions guide the sampling of new candidate solutions, steering the search toward configurations that have historically been successful, a strategy that improves exploration–exploitation balance compared to classical GA [15], [21]. This process occurs between steps 7 and 9.

The interaction between these components enables the system to explore the hyperparameter space more efficiently than traditional grid search methods, thereby improving both computational efficiency and generalization performance across datasets with different climatic characteristics. The complete operational workflow is summarized in Table 2. All simulations were conducted using Python 3.10 with the TensorFlow and scikit-learn libraries, which are widely used in rainfall forecasting research for their robustness and scalability [8], [10]. All experiments were executed on a workstation equipped with an Intel Core i7 CPU and 16 GB of RAM.

Table 2. Summary of the MGA–MLP process

Process	Description
Data loading	Load rainfall datasets (Australian and BMKG) from CSV files into system memory.
Preprocessing	Select relevant weather stations, clean missing values, normalize input features, and encode the binary target (1= rain, –1= no rain).
Generate initial population	Randomly generate an initial population of 20 chromosomes, where each chromosome encodes a set of MLP hyperparameters.
Train-test split (80:20)	Split the preprocessed data into 80% training and 20% testing subsets using random sampling.
Evaluate fitness	Train an MLP model for each chromosome configuration and calculate classification accuracy as its fitness score.
Select best parents	Identify and retain top-performing chromosomes based on fitness scores.
Update probabilities	Adapt the distribution of hyperparameter values for the next generation based on traits of the best five parent chromosomes.
Crossover and mutation	Perform crossover and mutation to generate new offspring chromosomes from selected parents.
Introduce random chromosomes	Add randomly generated chromosomes into the population to maintain exploration and genetic diversity.
Check termination	Determine whether the maximum generation limit (100) has been reached. If not, return to training step for the next generation.
Final model output	Output the final MLP model using the best hyperparameter configuration obtained through the MGA process.

3. RESULTS AND DISCUSSION

3.1. Performance of MLP using modified genetic algorithm for hyperparameter tuning

This study optimized a MLP model using MGA for hyperparameter tuning. The implementation of the MLP model was carried out using the scikit-learn library in Python. The model training function is defined as:

```
Def MLP_Model(strukturHidden,aktivasi,optimizer,epoch,learning_rate,x,y,xVal,yVal):
   clf=MLPClassifier (hidden_layer_sizes=strukturHidden,  activation=aktivasi, solver=optimizer,
   learning_rate_init=learning_rate, max_iter=epoch, random_state=1, verbose=False, tol=0.000001)
   clf.fit(x,y);  ypred=clf.predict(xVal)
   accurate = accuracy_score(yVal, ypred)
   return round(accurate, 5)
```

This function receives several hyperparameters as inputs and returns the model's classification accuracy, which is computed based on predictions made on the validation dataset [9]. The MLP model is tuned using six key hyperparameters, as shown in Table 3.

The dataset was randomly split into 80% training and 20% validation subsets to evaluate the model's performance. The genetic algorithm (GA) iteratively updates the hyperparameters through successive generations to maximize the model's accuracy. In each generation, the MLP model is trained using the specified hyperparameters, and its classification accuracy is computed using the validation subset. The result is rounded to five decimal places to maintain consistency during comparison. This setup allows for an adaptive and systematic search over the hyperparameter space, leading to the discovery of optimal configurations that improve the MLP's ability to predict rainfall outcomes accurately across different datasets.

Table 3. Key hyperparameters used in MLP optimization

Hyperparameter	Description
Hidden layer size	Specifies the number of hidden layers and the number of neurons in each layer.
Activation function	Determines the non-linear transformation applied to the output of each neuron in the hidden layers. Standard options include 'rel', 'logistic', and 'tanh'.
Solver (optimizer)	Refers to the optimization algorithm that adjusts the weights during backpropagation. Standard solvers include 'adam', 'sgd', and 'lbfgs'.
Learning rate	Refers to controls the size of the weight updates during training. These hyperparameters affect the convergence speed and stability of the model.
Epochs	The number of complete passes over the training dataset, contributing to the training depth of the model.
Other hyperparameters	Any parameters not explicitly mentioned were retained at their default values as defined by the scikit-learn implementation.

3.1.1. Chromosome formation in the genetic algorithm

In the context of this study, each chromosome within the GA encodes a unique configuration of hyperparameters for the MLP model. The encoded parameters include the number of hidden layers, the activation function, the optimizer type, the number of training epochs, and the learning rate. All other hyperparameters not explicitly defined are retained at their default values as specified by the system's library. The chromosome structure is formally represented as follows:

Chromosome structure = {hidden layers, activation function, optimizer, epochs, learning rate}

The GA population consists of multiple chromosomes sharing the same structural format. Each chromosome is assigned an index from 1 to n. The model accuracy associated with each chromosome is calculated during the evolutionary process. Chromosomes with the highest accuracy are selected as parents for the next generation. New chromosomes are then generated through crossover and mutation operations applied to these selected parents, allowing exploration of potentially better hyperparameter configurations in the solution space.

3.1.2. Modified genetic algorithm

This study introduces MGA that incorporates two primary enhancements to improve performance and stability:

i) Preservation of high-performing chromosomes across generations: this strategy ensures that the best-performing chromosomes are retained throughout the evolutionary process, avoiding the loss of optimal solutions due to random variation.
ii) Adaptive selection probability based on chromosome fitness: chromosomes with higher accuracy are more likely to be selected for crossover and mutation. This selection accelerates convergence and increases the likelihood of discovering high-performing configurations earlier.

These modifications aim to address the limitations of conventional GAs, such as the risk of premature convergence and the instability caused by purely random selection mechanisms. As a result, the modified algorithm achieves improved model performance, attaining an accuracy of 86.016% on the Australian rainfall dataset and 79.05% on the BMKG dataset. These results surpass the baseline performance of standard GAs previously reported in the literature [12]. Furthermore, the adaptability of the proposed MGA makes it applicable beyond rainfall prediction, offering potential benefits for optimization tasks in domains such as financial forecasting and healthcare systems. This MGA significantly contributes to hyperparameter optimization in ML by improving stability, convergence speed, and generalization capability.

3.2. Result

3.2.1. Hidden layers analysis

The proposed method was applied to two rainfall datasets: the Australian rainfall dataset obtained from Kaggle.com and the Indonesian BMKG rainfall dataset. Both datasets were randomly divided into training and testing subsets in an 80:20 ratio. Through multiple generations of evolutionary optimization, optimal hyperparameter configurations were obtained, following similar approaches in recent rainfall prediction studies [6], [16], [26].

i) Australian data: the highest accuracy of 0.86016 was achieved using the following configuration:
– 'hidden_layers': [9, 7, 3, 3, 6, 3, 8, 10, 8]
– 'activation_function': 'tanh'
– 'optimizer': 'lbfgs'

– 'epoch': 1000
– 'learning_rate': 0.001

ii) BMKG Data: The highest accuracy of 0.7905 was achieved using the following configuration:
– 'hidden_layers': [10, 7]
– 'activation_function': 'tanh'
– 'optimizer': 'lbfgs'
– 'epoch': 1000
– 'learning_rate': 0.001

The model was executed for 100 generations, evaluating multiple hidden layer architectures. It was observed that two hidden layers consistently produced the highest accuracy across both datasets. Table 4 summarizes the various hidden layer configurations that led to firm performance.

Hidden layers in artificial neural networks play a crucial role in processing complex rainfall data. The dataset used in this study has moderate feature dimensions (16 features for the Australian data and 8 for the BMKG data) with overlapping class distributions. Based on the experimental results, two hidden layers were optimal for capturing non-linear patterns and improving the model's ability to better separate the data classes. The first layer captures complex feature interactions, while the second layer simplifies the data representation into high-level abstractions, enhancing the model's classification ability. Using two hidden layers, the model effectively captures data patterns without causing overfitting. Reducing the number of nodes between the first and second layers, such as in configurations like [10, 5] or [14, 7], is a natural form of regularization, improving the model's generalization capability.

In addition to using two hidden layers, activation functions such as rectified linear unit (ReLU) and tanh were employed to support the non-linear transformations required for processing rainfall data. ReLU is particularly effective because it filters out noise by mapping negative values to zero, while tanh provides competitive performance for smaller networks with normalized data, thanks to its symmetric output range of [-1, 1]. Combining these activation functions contributed to achieving high accuracy (86.016% for the Australian data and 79.05% for the BMKG data) while maintaining computational efficiency. Experiments with configurations involving more than two hidden layers showed the risk of overfitting without significantly improving accuracy. Conversely, using only one hidden layer proved insufficient to adequately capture the data's complexity. Therefore, two hidden layers offer an optimal balance between model capacity, generalization, and computational efficiency.

Table 4. Hidden layer structures associated with high classification accuracy

Hidden layer count	Generations	Unique hidden layer structures
1	10	[11], [7], [2], [14]
2	41	[10, 5], [12, 1], [14, 7], [10, 7], [5, 3], [6, 9], [11, 4], [12, 14]
3	5	[6, 8, 2], [13, 13, 7]
5	17	[12, 11, 4, 9, 2], [6, 14, 13, 5, 5], [14, 6, 10, 5, 12], [7, 3, 1, 11, 11], [12, 12, 6, 2, 11]
6	13	[5, 12, 1, 2, 11, 3], [9, 2, 8, 3, 8, 3], [4, 14, 4, 5, 10, 6]
7	2	[6, 5, 7, 13, 3, 14, 7]
8	3	[5, 8, 8, 8, 1, 10, 4, 11], [11, 4, 8, 8, 5, 10, 10, 6]
9	9	[13, 8, 10, 13, 14, 1, 4, 13, 14], [11, 11, 8, 2, 13, 13, 7, 8, 9], [13, 10, 6, 3, 7, 3, 2, 13, 8], [4, 2, 2, 14, 10, 6, 7, 1, 7], [6, 4, 3, 8, 8, 3, 10, 8, 5]

3.2.2. Node count optimization

The number of nodes in the hidden layers significantly influences the network's capacity to capture patterns within the data. In this study, the rainfall datasets comprising 16 features in the Australian dataset and 8 in the BMKG dataset required an architecture capable of modelling moderate complexity. Optimal configurations such as [10, 5] and [14, 7] indicate that the first layer captures complex feature interactions while the second layer condenses these interactions into abstract representations suitable for classification. From a model design perspective, the ideal number of nodes balances representational power and generalization. A progressive reduction in node count between hidden layers acts as an implicit regularize, mitigating the risk of overfitting, which aligns with prior findings on neural network architectures for rainfall prediction [9]. This behavior was reflected in the achieved accuracies: 86.016% for the Australian dataset and 79.05% for the BMKG dataset.

Further experiments revealed that using too few nodes (e.g., [5, 2]) led to underfitting, while an excessive number of nodes (e.g., [20, 15]) increased computational cost without substantial gains in accuracy. Therefore, configurations such as [10, 5] and [14, 7] were found to strike the optimal balance between model complexity, efficiency, and accuracy. In total, eight two-layer hidden structures consistently yielded high accuracy: [10, 5], [12, 1], [14, 7], [10, 7], [5, 3], [6, 9], [11, 4], and [12, 14].

3.2.3. Activation function selection

Activation functions are crucial components in artificial neural networks as they determine how data is processed at each layer. In this study, ReLU achieved the highest accuracy (86.016% for the Australian dataset and 79.05% for the BMKG dataset) due to its ability to handle complex non-linear patterns and efficiency in training large networks. The dataset used in this study has moderate yet complex feature dimensions, requiring an activation function capable of effectively capturing non-linear patterns. ReLU demonstrated superior performance on datasets with positive values because of its ability to filter out noise by mapping negative values to zero. On the other hand, tanh showed competitive results for smaller networks, as its symmetric output range of [-1, 1] is well-suited for normalized data. The average activation function that provided the best accuracy was 'relu', based on 15 runnings, and each running had 40 generations, as shown in Table 5.

According to Table 5, the activation functions achieving high accuracy were ReLU with 297 occurrences, tanh with 110, and logistic with 193. This result suggests that ReLU has the highest probability of delivering high accuracy. From the model perspective, ReLU helps address the vanishing gradient problem, enabling networks with two hidden layers to learn more complex patterns without losing efficiency, a property also noted in deep learning approaches for rainfall prediction [16]. So, it ensures that the model performs well on the training data and generalizes effectively to the testing data.

Table 5. The activation function

Activation function	Running														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ReLU	1	23	32	16	26	25	14	21	11	31	26	14	19	11	27
Tanh	1	3	3	7	5	10	16	7	2	6	5	2	20	11	12
logistic	38	14	5	17	9	5	10	12	27	3	9	24	1	18	1

3.2.4. Optimizer analysis

The optimizer determines how the model updates its parameters during training. In this study, three optimizers were compared: blogs, Adam, and stochastic gradient descent (SGD). Table 6 shows the frequency of high-accuracy outcomes across 15 experimental runs (40 generations each).

The research findings show that the limited-memory Broyden Fletcher Goldfarb Shannon (LBFGS) optimizer outperforms the others, with the highest frequency of achieving high accuracy. Several key factors can explain this result. First, LBFGS is a quasi-Newton algorithm designed to efficiently handle non-linear problems. MLP models use complex non-linear activation functions, making LBFGS ideal for achieving fast and stable convergence in this context. Using an approximated Hessian matrix approach, LBFGS takes more informed steps toward the function's minimum, improving efficiency. Second, compared to other optimizers such as SGD or Adam, LBFGS processes the entire dataset in each iteration (full-batch optimization). This approach minimizes noise in gradient estimates, resulting in a more stable optimization process. In this study, the dataset (rainfall data from Australia and BMKG) is of moderate size, so LBFGS performs optimally without suffering from memory limitations often associated with larger datasets. Third, the success of LBFGS can also be attributed to the combination of hyperparameters determined through the tuning process, particularly the learning rate of 0.001 and 1000 epochs. This combination provides a balanced exploration-exploitation tradeoff during optimization, allowing LBFGS to effectively search the parameter space and reach optimal convergence. Overall, the full-batch optimization, stability against noise, and efficiency in handling non-linear problems make LBFGS an excellent choice for this study. These findings are consistent with prior works that have highlighted the effectiveness of LBFGS in training neural networks with structured and moderately sized datasets [16].

Table 6. Frequency of best-performing optimizers

Optimizer	Running														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LBFGS	40	35	27	35	27	39	36	37	38	31	25	37	23	36	39
SGD	0	1	1	1	1	0	1	1	1	6	3	0	1	0	1
Adam	0	4	12	4	12	1	3	2	1	3	12	3	16	4	0

3.2.5. Epoch optimization

This study tested several epoch values: 100, 200, 300, 400, 500, 1000, 2000, and 3000. These values were chosen randomly, and each epoch was evaluated across 15 runs, with 40 generations per run. The results are summarized in Table 7. As shown in the table, an epoch value of 1000 most frequently produced the highest accuracy. Selecting the optimal number of epochs is crucial to ensure that the model learns data patterns effectively without causing overfitting. In this study, 1000 epochs were found to provide the best performance. Experiment results show that fewer epochs, such as 500 or 300, caused the model to fail in achieving full convergence (underfitting). In comparison, more epochs, such as 2000 or 3000, did not yield significant accuracy improvements and even increased the risk of overfitting. The rainfall datasets with moderate feature dimensions and nonlinear patterns required sufficient training time to learn complex patterns. One thousand epochs ensured that the model had enough time to converge, which was supported by its high performance on the test data (86.016% for the Australian dataset and 79.05% for the BMKG dataset). These findings are consistent with previous research indicating that an appropriate number of training epochs, matched to dataset complexity and optimizer characteristics, is essential for achieving high accuracy without overfitting [1]. Furthermore, this choice complements the LBFGS optimizer, which relies on stable and full-batch updates to efficiently find optimal solutions. With 1000 epochs, the model achieved a balance between adequate training and generalization capability, reflected in the consistency of accuracy across multiple runs.

Table 7. Frequency of best-performing epochs across 15 runs

Epoch	Number of running															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
100	5	9	0	0	4	0	0	1	3	0	9	2	0	0	0	33
200	1	0	11	2	2	4	10	13	0	17	0	15	1	1	16	93
300	0	1	0	1	4	0	0	1	0	1	4	4	4	3	8	31
400	3	7	2	8	4	11	0	1	9	0	3	8	2	13	2	73
500	3	1	4	0	6	1	2	11	0	8	14	8	16	3	4	81
1000	5	1	12	15	2	22	5	1	10	10	3	0	7	1	9	103
2000	18	2	7	13	15	2	5	2	6	1	7	2	0	11	0	91
3000	5	19	4	1	3	0	18	10	12	3	0	1	10	8	1	95

3.2.6. Learning rate optimization

The count of optimal learning rate selections is presented in Table 8. The table shows that the learning rate of 0.001 most frequently resulted in high accuracy. While a lower learning rate of 0.0001 was also tested, it led to a lower frequency of high accuracy, indicating that a lower learning rate does not necessarily yield better performance.

Table 8. Frequency of best-performing learning rates across 15 runs

Optimizer	Running															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
1	0	0	0	0	0	2	2	1	1	1	2	0	0	8	3	20
0.1	0	0	0	2	0	1	1	0	0	0	0	0	1	8	0	13
0.01	0	1	3	5	2	3	3	1	1	0	6	2	1	0	14	42
0.001	32	20	28	12	13	20	20	13	4	5	25	13	2	17	9	233
0.0001	8	19	9	21	25	14	14	25	34	34	7	25	36	7	14	292

As supported by prior research on neural network optimization [1], the selection of an appropriate learning rate is critical for achieving high accuracy and stable convergence. The optimal learning rate of 0.001 was the most effective in this study. This value strikes a balance between convergence speed and the stability of the optimization process. Higher values, such as 0.01, led to instability in training with fluctuating accuracy, while smaller values, such as 0.0001, resulted in overly slow convergence. When combined with the 'LBFGS' optimizer, a learning rate of 0.001 allows for precise parameter updates, ensuring efficient and consistent convergence. Additionally, the synergy between this learning rate and the selected epoch value (1000 epochs) enabled the model to capture complex patterns in the high-dimensional rainfall datasets, as evidenced by high accuracy on both the Australian (86.016%) and BMKG (79.05%) data. This selection also supported the model's ability to generalize effectively, evident from the strong test data performance.

Choosing the optimal learning rate is a key factor in training neural networks. In this study, 0.001 provided the best performance, ensuring stable and efficient parameter updates while enabling fast convergence without the risk of divergence or overtraining. The rainfall datasets here contain complex, non-linear patterns with overlapping class distributions. The 0.001 learning rate allowed the model to effectively learn these patterns, achieving high accuracy in test data. It demonstrated a more stable performance than other values, such as 0.01 or 0.0001. The 0.01 rate caused instability in training, while 0.0001 slowed convergence and failed to reach the optimal solution. Furthermore, 0.001 is highly compatible with the 'LBFGS' optimizer, which relies on stable gradients to accelerate convergence. This learning rate also helped the model avoid overfitting, as seen in its consistent performance across multiple experiments. Therefore, selecting a learning rate of 0.001 provides optimal accuracy, stability, and efficiency balance.

4. CONCLUSION

This study presents MGA to optimize hyperparameters of a MLP model for rainfall prediction. The MGA incorporates two primary enhancements: i) elitism preservation, which ensures that the best-performing chromosomes are retained across generations, and ii) adaptive selection probability based on chromosome fitness, which increases the likelihood of high-performing configurations being passed on. These improvements address common limitations in standard GA, such as random selection instability and premature convergence. The proposed method was evaluated using two real-world rainfall datasets: the Australian and Indonesian BMKG datasets. Experimental results showed that the MGA–MLP achieved classification accuracies of 86.016% and 79.05%, respectively, demonstrating superior performance compared to baseline GA-tuned models. Analysis of hidden layer structures revealed that two-layer configurations with decreasing node counts (e.g., [10, 5] or [14, 7]) provided optimal generalization. ReLU and tanh activation functions, LBFGS optimizer, 1000 epochs, and a learning rate of 0.001 were also identified as key contributors to performance. In conclusion, the proposed MGA offers a reliable, adaptive, and efficient approach to hyperparameter tuning in neural networks. Its ability to generalize across different climate datasets highlights its potential for broader applications, including finance, agriculture, and environmental modeling.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the leadership of the Faculty of Computer Science for the financial support and trust they provided, which contributed to the successful completion of this study.

FUNDING INFORMATION

This research was funded by the Research Fund of the Faculty of Computer Science, Universitas Brawijaya, based on the Dean's Decree Number 362/2024 and the Research Implementation Agreement Number 4567/UN10.F15/PN/2024.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Marji	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Wayan Firdaus Mahmudi	✓	✓				✓		✓	✓		✓	✓		✓
Endang Wahyu Handamari			✓	✓	✓		✓			✓	✓			
Edy Santoso		✓				✓		✓	✓		✓	✓		✓
Maulana Muhamad Arifi				✓	✓		✓		✓	✓		✓	✓	

C : **C**onceptualization	I : **I**nvestigation	Vi : **Vi**sualization
M : **M**ethodology	R : **R**esources	Su : **Su**pervision
So : **So**ftware	D : **D**ata Curation	P : **P**roject administration
Va : **Va**lidation	O : Writing - **O**riginal Draft	Fu : **Fu**nding acquisition
Fo : **Fo**rmal analysis	E : Writing - Review & **E**diting	

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

INFORMED CONSENT

This study does not involve individual personal data or identifiable human subjects. Therefore, informed consent was not required.

ETHICAL APPROVAL

This study did not involve human participants or animals and, therefore, did not require ethical approval. All procedures were conducted in accordance with relevant institutional and national guidelines.

DATA AVAILABILITY

The datasets utilized in this study are openly available from two sources: Kaggle at https://www.kaggle.com/datasets/trisha2094/weatheraus and the official Indonesian Meteorological, Climatological, and Geophysical Agency (BMKG) Data Online portal at https://dataonline.bmkg.go.id.

REFERENCES

[1] L. Chen, B. Han, X. Wang, J. Zhao, W. Yang, and Z. Yang, "Machine learning methods in weather and climate applications: a survey," *Applied Sciences*, vol. 13, no. 21, 2023, doi: 10.3390/app132112019.

[2] S. M. Gowtham, Y. S. Ganesh, and M. M. Ali, "Efficient rainfall prediction and analysis using machine learning techniques," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 6, pp. 3467–3474, 2021.

[3] A. Alelaimat, I. Yusoff, M. K. Nizar, T. F. Ng, and Y. A. Majali, "Groundwater management in the face of climate change: enhancing groundwater storage in the alluvium aquifer of Wadi Araba, Jordan, through GIS-based managed aquifer recharge and groundwater MODFLOW," *Water Supply*, vol. 23, no. 12, pp. 5136–5153, 2023, doi: 10.2166/ws.2023.316.

[4] H. Gebreslassie, G. Berhane, T. Gebreyohannes, M. Hagos, A. Hussien, and K. Walraevens, "Water harvesting and groundwater recharge: a comprehensive review and synthesis of current practices," *Water*, vol. 17, no. 7, 2025, doi: 10.3390/w17070976.

[5] F. Gradiyanto, P. N. Parmantoro, and Suharyanto, "Impact of climate change on Kupang River flow and hydrological extremes in Greater Pekalongan, Indonesia," *Water Science and Engineering*, vol. 18, no. 1, pp. 69–77, 2025, doi: 10.1016/j.wse.2024.03.005.

[6] Z. H. Doost, A. Alsuwaiyan, A. Abdulraheem, N. M. Al-Areeq, and Z. M. Yaseen, "Rainfall prediction using integrated machine learning models with k-means clustering: a representative case study of Harirud Murghab Basin-Afghanistan," *IEEE Access*, vol. 13, pp. 111628–111646, 2025, doi: 10.1109/ACCESS.2025.3581921.

[7] V. Kantharia, D. Mehta, V. Kumar, M. P. Shaikh, and S. Jha, "Rainfall–runoff modeling using an adaptive neuro-fuzzy inference system considering soil moisture for the damanganga basin," *Journal of Water and Climate Change*, vol. 15, no. 5, pp. 2518–2531, 2024, doi: 10.2166/wcc.2024.143.

[8] D. Mahajan and S. Sharma, "Prediction of rainfall using machine learning," in *2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, 2022, pp. 1–4, doi: 10.1109/ICERECT56837.2022.10059679.

[9] W. Wang, Y. Du, K. Chau, H. Chen, C. Liu, and Q. Ma, "A comparison of BPNN, GMDH, and ARIMA for monthly rainfall forecasting based on wavelet packet decomposition," *Water*, vol. 13, no. 20, 2021, doi: 10.3390/w13202871.

[10] Y. Liao, S. Lu, and G. Yin, "Short-term and imminent rainfall prediction model based on ConvLSTM and SmaAT-UNet," *Sensors*, vol. 24, no. 11, 2024, doi: 10.3390/s24113576.

[11] B. Hu *et al.*, "Multisource precipitation data merging using a dual-layer ConvLSTM model," *Remote Sensing*, vol. 17, no. 3, 2025, doi: 10.3390/rs17030546.

[12] R. Farooq, M. A. Imteaz, and F. Mekanik, "A pioneering approach to deterministic rainfall forecasting for wet period in the Northern Territory of Australia using machine learning," *Earth Science Informatics*, vol. 18, no. 2, 2025, doi: 10.1007/s12145-025-01724-0.

[13] A. Rahman *et al.*, "Rainfall prediction system using machine learning fusion for smart cities," *Sensors*, vol. 22, no. 9, 2022, doi: 10.3390/s22093504.

[14] W. Fang, H. Qin, Q. Lin, B. Jia, Y. Yang, and K. Shen, "Deep learning integration of multi-model forecast precipitation considering long lead times," *Remote Sensing*, vol. 16, no. 23, 2024, doi: 10.3390/rs16234489.

[15] X. Li, S. Rui, Y. Niu, and Y. Liu, "Precipitation prediction using an ensemble of lightweight learners," *arXiv-Physics*, pp. 1–7, 2023.

[16] I. V. Necesito, D. Kim, Y. H. Bae, K. Kim, S. Kim, and H. S. Kim, "Deep learning-based univariate prediction of daily rainfall: application to a flood-prone, data-deficient country," *Atmosphere*, vol. 14, no. 4, 2023, doi: 10.3390/atmos14040632.

[17] I. Price and S. Rasp, "Increasing the accuracy and resolution of precipitation forecasts using deep generative models," *Proceedings of Machine Learning Research*, vol. 151, pp. 10555–10571, 2022.

[18] S. Ravuri *et al.*, "Skilful precipitation nowcasting using deep generative models of radar," *Nature*, vol. 597, pp. 672–677, 2021, doi: 10.1038/s41586-021-03854-z.

[19] X. Ling, C. Li, F. Qin, L. Zhu, and Y. Huang, "Two-stage rainfall-forecasting diffusion model," *arXiv-Computer Science*, pp. 1–8, 2024.

[20] L. Harris, A. T. T. McRae, M. Chantry, P. D. Dueben, and T. N. Palmer, "A generative deep learning approach to stochastic downscaling of precipitation forecasts," *Journal of Advances in Modeling Earth Systems*, vol. 14, no. 10, 2022, doi: 10.1029/2022MS003120.

[21] R. M. -Quintanar, C. E. G. -Tejada, J. G. -Tejada, H. G. -Rosales, S. D. J. M. -Gallegos, and A. G. -Domínguez, "Auto-machine-learning models for standardized precipitation index prediction in North–Central Mexico," *Climate*, vol. 12, no. 7, 2024, doi: 10.3390/cli12070102.

[22] Y. Wen, W. Yu, F. Zheng, D. Huang, and N. Xiao, "AdaNAS: Adaptively postprocessing with self-supervised neural architecture search for ensemble rainfall forecasts," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–10, 2024, doi: 10.1109/TGRS.2024.3360636.

[23] S. He, H. Liang, Y. Zhang, and X. Yuan, "Skillful high-resolution ensemble precipitation forecasting with an integrated deep learning framework," *arXiv-Computer Science*, pp. 1–19, 2025.

[24] N. B. -Barrenetxea, R. M. -España, P. J. -Sáez, S. H. Faria, and J. S. -Aparicio, "Multi-model ensemble machine learning approaches to project climatic scenarios in a River Basin in the pyrenees," *Earth Systems and Environment*, vol. 8, pp. 1159–1177, 2024, doi: 10.1007/s41748-024-00408-x.

[25] E. Orlova, H. Liu, R. Rossellini, B. A. Cash, and R. Willett, "Beyond ensemble averages: leveraging climate model ensembles for subseasonal forecasting," *Artificial Intelligence for the Earth Systems*, vol. 3, no. 4, 2024, doi: 10.1175/AIES-D-23-0103.1.

[26] W. Lin *et al.*, "Study of the spatiotemporal distribution characteristics of rainfall using hybrid dimensionality reduction-clustering model: a case study of Kunming City, China," *Atmosphere*, vol. 15, no. 5, 2024, doi: 10.3390/atmos15050534.

BIOGRAPHIES OF AUTHORS

Marji has been a faculty member at Brawijaya University since 1995. He earned his Master's in Computer Engineering from Institut Teknologi Bandung (ITB) in 2000. He completed his Doctorate in Environmental Science from Universitas Brawijaya in 2023, with research focusing on rainfall status classification using machine learning methods. He teaches fundamental courses such as calculus, numerical methods, statistics, and probability theory. He can be contacted at email: marji@ub.ac.id.

Wayan Firdaus Mahmudi received his bachelor's degree in Mathematics from Brawijaya University in 1995, his master's degree from Institut Teknologi Sepuluh Nopember (ITS) in 1999, and his Ph.D. from the University of South Australia in 2013. Over the past five years, he has taught courses in evolutionary algorithms, intelligent systems, machine learning, object-oriented modeling, and decision support systems. His research interests include artificial intelligence and its applications, with multiple publications in international journals. He serves as Dean of the Faculty of Computer Science at Brawijaya University. He can be contacted at email: wayanfm@ub.ac.id.

Endang Wahyu Handamari has been a lecturer at the Department of Mathematics, Brawijaya University, since 1991. She obtained his undergraduate degree in Mathematics from Institut Teknologi Sepuluh Nopember in 1985 and his master's in Applied Mathematics from Institut Teknologi Bandung in 1996. Her academic interests include probability theory and stochastic processes, and she is actively involved in developing the Actuarial Science Program at Universitas Brawijaya. In 2024, she researched insurance claim risk classification using the multilayer perceptron method. She can be contacted at email: ewahyu-math@ub.ac.id.

Edy Santoso has served as a lecturer at Universitas Brawijaya since 2003. He completed his bachelor's degree in Mathematics with a specialization in Computer Science in 1998 and earned his master's degree in Computer Engineering from Institut Teknologi Sepuluh Nopember. He also holds a Professional Engineering certification from Brawijaya University. His research focuses on handwriting recognition and intelligent computing. He teaches artificial intelligence, calculus, linear algebra, and databases. He can be contacted at email: edy144@ub.ac.id.

Maulana Muhamad Arifin earned his bachelor's degree in Mathematics from Brawijaya University in 2018 with a thesis on bi-univalent functions and completed his master's degree in 2021 with research related to COVID-19 insurance modeling. His recent work focuses on machine learning applications, remarkably comparing multilayer perceptron and support vector machine methods for rainfall prediction with optimized parameter tuning. He can be contacted at email: maulana.m.a.x2@gmail.com.