# A COMPARISON OF ADAM AND NELDER–MEAD SIMPLEX OPTIMIZATION IN NEURAL NETWORK-BASED RELATIVE POVERTY CLASSIFICATION

Nabil Bintang Prayoga [1)]
Rose Dwi Aulia Amaradhani [2)]
Vito Raditya Pratama [3)]
Raihan Sabillah Adisecha [4)]
Fathiyya Mufida [5)]
Akbar Rizki [6)]
[1,2,3,4,5,6)] Study Program in Statistics and Data Science, IPB University
e-mail: nabilbintangprayoga@gmail.com

**ABSTRACT**

Statistical optimization plays an important role in improving the performance of neural network (NN) models by optimizing their weights and biases. This study compares the performance of the Adaptive Moment Estimation (Adam) and Nelder-Mead (NM) simplex optimization methods for neural network-based relative poverty classification in Indonesia. Secondary data from Statistics Indonesia (BPS) in 2023 covering 514 regencies/cities, nine explanatory variables, and one binary response variable representing relatively high-poverty and relatively low-poverty areas were analyzed. The analysis included data preprocessing, exploratory data analysis, NN model development, hyperparameter tuning, and performance evaluation using the area under the receiver operating characteristic curve (AUC), accuracy, precision, sensitivity, specificity, and F1-score. The Adam model consistently outperformed the NM simplex model across almost all evaluation metrics, achieving an AUC of 0.818 compared with 0.735. Adam also obtained higher accuracy (0.777 vs. 0.738), precision (0.771 vs. 0.692), specificity (0.869 vs. 0.803), and F1-score (0.701 vs. 0.667), while both models achieved the same sensitivity (0.643). These findings demonstrate that Adam provides a more effective and computationally efficient optimization approach than the NM simplex for neural network-based relative regional poverty classification in Indonesia.

Keywords: Adam, Nelder-Mead simplex, neural network, optimization, poverty.

## INTRODUCTION

Statistical optimization is the process of finding parameter values that minimize or maximize an objective function to obtain the best solution within a model. Optimization methods are generally divided into two categories, namely derivative-based (gradient-based) and derivative-free. Derivative-based methods use derivative information from the objective function to determine the direction of the solution search, while derivative-free methods search for solutions without using gradient information.

Gradient descent is one of the most commonly used derivative-based optimization methods. This method uses an iterative algorithm that updates parameters by moving in the negative direction of the loss function gradient Gradient (Goodfellow et al., 2016). Gradient descent has been developed in various variations, one of which is adaptive moment estimation (Adam) that combines the advantages of momentum methods and root mean square propagation (RMSprop). Adam adjusts the learning rate for each parameter adaptively using estimates of the first and second moments of the gradients (Kingma & Ba, 2015).

The second category of optimization is Simplex, which is one of the classical derivative-free methods in solving linear optimization problems (Apriza et al., 2025). Simplex works by forming a set of solution points in the parameter space and evaluating the objective function value at each point to determine a better search direction (Dantzig et al., 1955). This method is very effective for

linear problems but is less suitable for complex and nonlinear objective functions. A development of the simplex is the Nelder-Mead simplex (NM simplex), which can be used for derivative-free nonlinear function optimization, although it has limitations in high dimensions. This method utilizes reflection, expansion, contraction, and shrinking operations on the vertices of the simplex to find the local minimum of the objective function (Nelder & Mead, 1965).

Statistical optimization, both derivative-based and derivative-free, plays an important role in training deep learning models such as neural networks (NN). Inspired by biological neural networks, NN have been widely applied in classification, prediction, and pattern recognition tasks (Windarto, 2017). However, these methods may suffer from slow convergence, training instability, and susceptibility to local minima (Suhendra & Wardoyo, 2015). However, these methods often experience slow convergence, instability during training, and susceptibility to local minima (Suhendra & Saputra, 2020). As an alternative, derivative-free optimization methods such as the Nelder–Mead (NM) simplex optimize model parameters without gradient information and have demonstrated superior performance in NN-based age and gender classification (Ozaki et al., 2017).

Poverty remains a crucial problem in Indonesia, especially in rural areas and regions with uneven development. The complexity of poverty factors makes poverty classification a challenging task that requires accurate predictive model, such as NN. Previous studies have applied NN trained using backpropagation algorithms poverty classification at the regency/city level in Indonesia (Saputri, 2023). In addition, studies in other classification domains have reported the effectiveness of Adam optimization in improving model performance (Saragih & Huda, 2022). However, optimization strategies in NN training are generally dominated by derivative-based methods, while the exploration of derivative-free optimization methods, such as the Nelder–Mead (NM) simplex method, remains limited, particularly in regional poverty classification problems. It should be noted that the classification used in this study is based on aggregate poverty rates at the regency/city level and therefore represents relative regional poverty conditions rather than the official poverty status of individuals or households. The classification threshold is based on the empirical mean poverty rate across the observed regencies/cities and should not be interpreted as an official national poverty threshold. Therefore, this study contributes by providing an empirical comparison between derivative-based (Adam) and derivative-free (NM simplex) optimization approaches in NN training for poverty level classification across regencies/cities in Indonesia. The findings are expected to provide insights into the suitability of alternative optimization strategies for improving NN classification performance.

## METHOD

### Data

The data used in this study are secondary data with one response variable and nine explanatory variables. The data were sourced from the Statistics Indonesia (BPS) in 2023, consisting of 514 observations representing each regency/city in Indonesia. The outcome variable in this study represents a relative regional poverty classification rather than an official poor/non-poor classification. The unit of analysis is the regency/city, and the poverty indicator used is the percentage of the population living below the poverty line as reported for each of the 514 regencies/cities in Indonesia. To construct the binary classification target, an empirical benchmark was calculated as the unweighted arithmetic mean of the poverty rates across all 514 regencies/cities included in the study. Specifically, the cut-off was calculated as:

$$c = \frac{1}{N}\sum_{i=1}^{N} p_i = \frac{1}{514}\sum_{i=1}^{514} p_i = 12.27\%$$

Where:

$c$ : empirical cut-off value,

$p_i$ : poverty rate of the i-th regency/city

$N$ : total number of regencies/cities included in the study.

Regencies/cities with poverty rates above 12.27% were classified as relatively high-poverty regions and assigned a value of 1, whereas those with poverty rates equal to or below 12.27% were classified as relatively low-poverty regions and assigned a value of 0. This approach was applied to transform the continuous poverty rate variable into a binary classification target for the subsequent modelling process (Wulandari et al., 2022).

The 12.27% threshold should therefore not be interpreted as the official national poverty rate or as an official poverty line. Rather, it serves as a data-driven, cross-regional benchmark for distinguishing areas with poverty incidence above and below the average level observed among Indonesian regencies/cities in the study. This approach is consistent with the regional unit of analysis and with the objective of the study, which is to classify relative differences in regional poverty incidence rather than to determine the poverty status of individuals or households.

The explanatory variables in the model are determined based on previous research and the author's preferences. Development and economic disparities result in high poverty rates in Indonesia, especially in certain regencies or cities. This condition is influenced by various factors, such as the open unemployment rate ($X_7$) (Wulandari et al., 2022), average years of schooling ($X_1$), economic growth (GRDP) ($X_9$) (Faritz & Soejoto, 2020), as well as life expectancy ($X_4$) and expected years of schooling (Allo & Fevriera, 2025). The use of adjusted per capita expenditure variable ($X_2$) refers to the latest BPS method that applies calculations between per capita expenditure and purchasing power parity (PPP), based on 96 current commodities as an update from the previous method (BPS Kabupaten Enrekang, 2024). Table 1 shows the variables used in this analysis and research.

**Table 1.** The variables

| Code | Variable | Unit |
|---|---|---|
| $Y$ | Poverty classification | Category (0 = relatively low-poverty, 1 = relatively high-poverty) |
| $X_1$ | Average years of schooling | Years |
| $X_2$ | Adjusted per capita expenditure | Thousand rupiah/person/year |
| $X_3$ | Human Development Index (HDI) | Points |
| $X_4$ | Life expectancy | Years |
| $X_5$ | Access to proper sanitation | Percent |
| $X_6$ | Safe drinking water | Percent |
| $X_7$ | Open unemployment rate | Percent |
| $X_8$ | Labor force participation rate | Percent |
| $X_9$ | Gross regional domestic product (GRDP) | Rupiah |

## Neural Network

NN is a computational system consisting of a collection of processing units (neurons) for classification or regression with several layers, namely the input layer, hidden layer, and output layer. This concept mimics the human nervous system, which consists of nerve cells or neurons connected through structures like axons and dendrites, with junction points called synapses (Aggarwal, 2023). Each neuron in a given layer receives input from neurons in the previous layer, processes it through an activation function, and passes its output to neurons in the next layer

(Kurniati et al., 2023). Figure 1 shows a simple architecture of NN that has three neurons in the input layer, one hidden layer with two neurons, and one neuron in the output layer. Each inter-neuron connection represents a weight, while each neuron has a bias added before the activation function is applied.

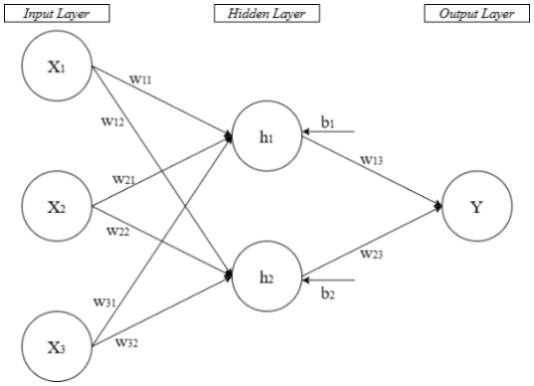



**Figure 1.** NN Architecture

NN classification works by processing $n$-dimensional input data through hidden layers with activation functions such as rectified linear unit (ReLU), and generating probability outputs using a sigmoid activation function. The process involves optimizing weights $(w)$ and biases $(b)$ whose number depends on the network architecture; for instance, a single hidden layer with $h$ neuronscontains $n \times h + h + h \times 1 + 1$ trainable parameters. Model's performance is evaluated using loss function that measures the deviation between predicted and actual values, forming a fundamental part of supervised learning in NN training (Goodfellow et al., 2016).

**Adaptive Moment Estimation**

Adaptive moment estimation (Adam) is a gradient-based optimization algorithm designed for training networks with large-scale and non-stationary data. This optimization method combines the advantages of the momentum approach, which retains the previous gradient direction to accelerate convergence, and RMSprop, which adjusts the learning rate based on the squared average of the gradients (Reddi et al., 2018). Adam calculates the first moment (mean of the gradients) and the second moment (variance of the gradients) for each parameter, and then adaptively adjusts the parameter updates based on equations 1 to 4 with the gradient $g_t = \nabla_\theta J(\theta_t)$ at time $t$ (Kingma & Ba, 2015). Calculating the first moment.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{1}$$

1. Calculating the second moment.

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{2}$$

2. Correcting bias.

$$\hat{m} = \frac{m_t}{1-\beta_1^t}, \quad \hat{v} = \frac{v_t}{1-\beta_2^t} \tag{3}$$

3. Updating parameters.

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \tag{4}$$

The learning rate in the formula is indicated by the symbol alpha, with $\epsilon$ as a small constant to avoid division by zero, and $\beta_1$ and $\beta_2$ as moment decay parameters. Adam can handle sparse gradients with convergence stability and adaptive learning rate settings for all parameters (Si et al., 2023). This optimization method relies on the backpropagation process to calculate the gradient of

the loss function with respect to the model parameters in the form of network weights using the chain rule. Backpropagation in the Adam optimization method has an advantage in generalization capability, meaning that a trained network can recognize patterns from training data and apply them to new data (Agustin & Prahasto, 2012).

**Nelder-Mead Simplex**

The Nelder–Mead (NM) simplex method is conceptually distinct from the classical simplex method used in linear programming, although both methods use the term “simplex”. The term simplex is used in different optimization contexts. The classical simplex method is designed to solve linear programming problems involving linear objective functions and constraints, whereas the NM simplex method is a derivative-free optimization algorithm for unconstrained nonlinear optimization (Dantzig et al., 1955; Nelder & Mead, 1965). Thus, the classical simplex method and the NM simplex method should not be interpreted as the same algorithm or as linear and nonlinear versions of a single optimization procedure. In this study, the term NM simplex refers specifically to the NM method. Unlike the classical simplex method, the NM simplex forms a simplex, a geometric shape consisting of $n + 1$ vertices in an $n$-dimensional parameter space, and iteratively updates its position through reflection, expansion, contraction, and shrinkage operations to approach a local optimum (Nelder & Mead, 1965).

Reflection reflects the worst point with respect to the centroid of the other points, and if successful, indicated by a decrease in the loss value, the point search will be further extended in that direction during the expansion process. Contraction is performed if the reflection step fails, by taking a step back toward the centroid. The simplex will be reduced to the current best point when all steps fail in the process called reduction. The shape of the simplex changes according to the pattern of the function surface being minimized during the expansion or contraction process. This method allows the simplex to adapt to the local conditions of the function, making the search for the minimum value more efficient and precise (Baudin, 2010).

The NM simplex is used to directly optimize weights and biases based on the results of the forward pass process, without going through backpropagation in the NN. Each point in the simplex represents a set of weights and biases that are evaluated through a loss function. The parameter update process is carried out based on the actual loss, not the gradient, making this method flexible for simple networks or cases with complex loss function structures (Ozaki et al., 2017). This method is indeed superior for low-dimensional problems and non-smooth functions, but its performance can decline in high dimensions, so it does not guarantee global convergence (Kolda et al., 2003). Since each trainable parameter corresponds to one dimension in the optimization space, the number of simplex vertices increases linearly with the number of network parameters ($n + 1$ vertices for $n$ parameters). Consequently, the computational cost and memory requirements of the NM simplex increase substantially as the neural network architecture becomes larger. Therefore, the NM simplex is generally more suitable for relatively low-dimensional neural network architectures, whereas gradient-based optimizers such as Adam remain computationally efficient for larger networks.

**Evaluation Metrics**

Evaluation is carried out using the area under the curve (AUC) value and a confusion matrix. AUC is the area under the receiver operating characteristic (ROC) curve. The AUC value summarizes the model's performance across all classification thresholds (Li, 2024). The confusion matrix represents classification results through four categories, namely true positive (positive data correctly predicted), true negative (negative data correctly predicted), false positive (negative data incorrectly classified as positive), and false negative (positive data incorrectly predicted as

negative) (Rainio et al., 2024). These components form the basis for calculating evaluation metrics such as accuracy, precision, sensitivity, specificity, and F1-score to assess the model's performance comprehensively.

Accuracy is the proportion of total correct predictions, both positive and negative predictions, to all observations. Precision, also known as the positive predictive value, assesses the accuracy of positive predictions made by the model. Sensitivity is an indicator for evaluating the model's ability to identify all actual positive cases. This metric is often used when failing to detect positive cases (FN) has a significant impact. Specificity measures how well the model identifies all actual negative cases, the opposite of sensitivity. The F1 score is an indicator that combines the metrics of precision and sensitivity through the harmonic mean. The value of this indicator will only improve if both metrics are good. A model is considered better if these metrics have higher values (Rainio et al., 2024). Nevertheless, all five need to be considered proportionally because the dominance of one metric at the expense of others does not always indicate good overall performance.

**Analysis Procedure**

Data analysis was conducted using Python software. The procedures include exploration and modeling with two different types of optimization in NN classification.

1. Conducting exploratory data analysis by evaluating the balance of the relative regional poverty classification response variable; identifying and addressing multicollinearity among the explanatory variables.
2. Splitting the data into training and testing sets with an 80:20 ratio based on previous research that produced the best model from various proportions in ANN (Sakti et al., 2024).
3. Normalizing the data using the min-max method based on Equation 5 (Allorerung et al., 2024),
$$x' = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (5)$$
4. Performing NN modeling using the Adam optimization method with gradient calculation through the backpropagation algorithm.
   a. Performing hyperparameter tuning using a combination based on Table 2 on training data with scenarios from the best parameter results in previous (Alkhairi & Windarto, 2023; Prasetyawan et al., 2018).

**Table 2.** Hyperparameter Combination

| Scenario | Epoch | Learning Rate | Batch Size |
|---|---|---|---|
| 1 | 100 | 0.1 | 64 |
| 2 | | | 128 |
| 3 | | 0.01 | 64 |
| 4 | | | 128 |
| 5 | | 0.001 | 64 |
| 6 | | | 128 |
| 7 | 500 | 0.1 | 64 |
| 8 | | | 128 |
| 9 | | 0.01 | 64 |
| 10 | | | 128 |
| 11 | | 0.001 | 64 |
| 12 | | | 128 |

   b. Building a model by applying the following architecture:
      1) One hidden layer containing 64 neurons with a ReLU activation function.

2) One dropout layer with a ratio of 0.2 to prevent overfitting.
3) One output layer with a sigmoid activation function (for binary classification).
4) Loss function: Binary crossentropy.

c. Choosing the best hyperparameter combination through cross-validation (5-fold cross validation) based on the highest average area under the curve (AUC) value.
d. Determining the best classification threshold value based on the probability prediction results on the validation data using the maximum Youden index value from the receiver operating characteristic (ROC) curve.
e. Applying the model with the best hyperparameters and threshold value to the test data.
f. Evaluating the model performance on the test data using AUC metrics and other classification metrics, such as accuracy, precision, specificity, sensitivity, and F1 score.

5. Performing NN modeling using the NM simplex optimization method.
   a. Defining the activation function used, namely ReLU for the hidden layer and sigmoid for the output layer in binary classification as shown by Equations 6 and 7 (Szandała, 2020).

$$\mathrm{ReLU(z)} = \max(0, \mathrm{z}) \tag{6}$$

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{7}$$

   b. Building the forward pass structure of NN with the following architecture:
      1) One hidden layer with $n_h$ neurons, where $n_h \in \{2,3, \dots ,20\}$ as a hyperparameter to be tuned.
      2) One output layer with a sigmoid activation function.
      3) The forward pass process is carried out with an activation function that can be adjusted according to application needs (Goodfellow et al., 2016). Equations 8 to 11 describe the steps of this forward pass calculation.

$$\mathrm{Z}_1 = X.W_1 + b_1 \tag{8}$$

$$a_1 = ReLU(Z_i) \tag{9}$$

$$Z_2 = a_1.W_2 + b_2 \tag{10}$$

$$y_{pred} = \sigma(Z_2) \tag{11}$$

      The initial weights and biases are randomly initialized from a normal distribution $N(0,0.1^2)$. The weights and biases are packed into a single parameter vector $\theta$ that will be optimized.
   c. The objective function is defined as the log loss function (binary cross-entropy) as in Equation 12 (Bishop, 2006).

$$\mathrm{LogLoss} = -\frac{1}{\mathrm{n}}\sum_{i=1}^{n}[y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y}_i)] \tag{12}$$

      Evaluation was conducted using 5-fold cross-validation and the final log loss value was calculated as the average of each fold for each $n_h$ configuration.
   d. Optimization was performed using the NM simplex method with a maximum of 300 iterations. Weights and biases were initialized, weight optimization was done using the NM simplex to minimize log loss through cross-validation, and the best weights were selected based on the lowest average log loss for each $n_h$ configuration.
   e. Tuning of the best hidden layer neuron count was performed within the range of 2 to 20 based on the lowest average log loss value from cross-validation.
   f. The best classification threshold was determined based on the maximum Youden index value from the ROC curve on validation data from the model with the best hidden layer neuron count. The Youden index is a method for determining the optimal cutoff point that combines sensitivity and specificity, as in Equation 13 (Bantis et al., 2019).

$$J = \text{sensitivity} + \text{specificity} - 1 \tag{13}$$

g. Apply the model with optimal parameters (forward pass with the 4 best weights) and the best number of hidden layer neurons on the test data.
h. Evaluate the model's performance on the test data using the AUC metric and other classification metrics, such as accuracy, precision, specificity, sensitivity, and F1 score.

6. Compare the evaluation metrics on the test data for both optimization methods.
   Interpret and draw conclusions.

## RESULTS AND DISCUSSION

### Data Explorations

The relative regional poverty classification resulted in 302 regencies/cities being categorized as relatively low-poverty regions (0), while the remaining 212 were categorized as relatively high-poverty regions (1), as presented in Figure 2. The classification was based on the empirical mean poverty rate of 12.27% across the 514 regencies/cities included in the study. This value serves only as a relative regional benchmark and should not be interpreted as the official national poverty threshold. The ratio between the two classes was approximately 4:5, indicating a reasonably balanced class distribution. Therefore, no additional data-balancing procedure was applied. These labels were subsequently used as the classification target in the modelling process.

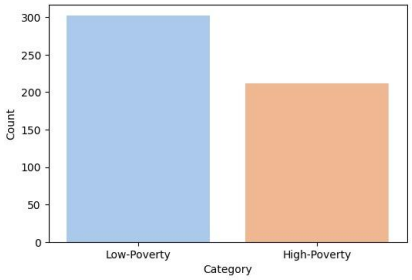



**Figure 2.** Distribution of the relative regional poverty classification

Feature redundancy among input variables was examined using the variance inflation factor (VIF). Although multicollinearity does not directly violate assumptions in neural network models as it does in regression models, highly correlated variables may provide redundant information and increase model complexity. Therefore, VIF analysis was applied as a preprocessing step to identify variables with excessive linear relationships among predictors. A VIF value greater than 10 was considered an indication of severe multicollinearity (Mohtasham et al., 2026).

**Table 3.** VIF test results for all variables

| Variable Code | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ | $X_9$ |
|---|---|---|---|---|---|---|---|---|---|
| VIF with $X_3$ | 9.488 | 8.581 | 36.070 | 4.491 | 2.278 | 1.637 | 2.159 | 1.951 | 1.578 |
| VIF without $X_3$ | 2.365 | 2.873 | - | 1.652 | 2.207 | 1.605 | 2.133 | 1.808 | 1.443 |

The VIF values in Table 3 show that the Human Development Index (HDI) ($X_3$) variable has a value greater than 10, indicating high redundancy with other socioeconomic indicators. Therefore, the HDI variable was excluded from the model. This decision was also supported by the formulation of HDI, which is calculated as a composite index consisting of the average years of schooling index ($I_{X_1}$), adjusted per capita expenditure index ($I_{X_2}$), and life expectancy index ($I_{X_4}$), as shown in Equation 14 (BPS Kabupaten Enrekang, 2024).

$$HDI = X_3 = \sqrt[3]{I_{X_1} \times I_{X_2} \times I_{X_4}} \times 100 \quad (14)$$

After variable selection, all remaining explanatory variables obtained VIF values below 10, indicating reduced redundancy among inputs. The selected variables were then divided into training and testing data with an 80:20 ratio. Subsequently, all input variables were standardized into a range of 0 to 1 using the min-max scaler method to ensure comparable variable scales during machine learning optimization (Prasetyo et al., 2022).

**NN Modeling with Adam Optimization**

The classification model is built with an architecture consisting of one hidden layer containing 64 neurons (ReLU), one dropout layer (rate 0.2), and one output layer (sigmoid). The training process is carried out using the backpropagation algorithm and the binary crossentropy loss function with the Adam optimization method for updating weights and biases. This process is applied to the model with the constructed architecture and algorithm on normalized training data. The model runs over several epochs with a certain batch size, with each iteration involving feedforward and backpropagation processes to minimize the loss value.

Weights and biases are determined through a gradient-based learning process carried out iteratively using the Adam optimization algorithm in classification using NN. Optimization is performed by calculating the derivatives (gradients) of the loss function with respect to each weight and bias through the backpropagation method. These weights and biases are then updated using a combination of the first and second moment averages of the gradients, which adaptively adjust the update step for each parameter (Kingma & Ba, 2015). Weights and biases are directly influenced by hyperparameters such as learning rate, batch size, and epoch. The learning rate determines the size of the weight update step in each iteration, while batch size affects the number of samples used to estimate the gradient. Epoch also influences how often the entire training data is used to update parameters (Goodfellow et al., 2016) The process of updating weights and biases, which is fully derivative-based in Adam optimization, makes the choice of activation function very important. ReLU, which is piecewise linear and non-saturating, allows the process of updating weights and biases to be more efficient and stable (Szandała, 2020). The details of hyperparameter tuning results are shown in Table 4.

**Table 4.** AUC values for each scenario

| Scenario | Epoch | Learning Rate | Batch Size | AUC values |
|---|---|---|---|---|
| 1 | 100 | 0.1 | 64 | 0.825 |
| 2 | | | 128 | **0.835** |
| 3 | | 0.01 | 64 | 0.826 |
| 4 | | | 128 | 0.823 |
| 5 | | 0.001 | 64 | 0.813 |
| 6 | | | 128 | 0.803 |
| 7 | 500 | 0.1 | 64 | 0.823 |
| 8 | | | 128 | 0.805 |
| 9 | | 0.01 | 64 | 0.833 |
| 10 | | | 128 | 0.824 |
| 11 | | 0.001 | 64 | 0.829 |
| 12 | | | 128 | 0.826 |

The best hyperparameters used consisted of a batch size of 128, a learning rate of 0.1, and a total of 100 epochs or in the second scenario, which resulted in the highest AUC value of 0.835. The model with this combination was retrained using all training data and produced an AUC value

of 0.855. The model was also applied to the test data to form a classification with an AUC result of 0.818. The relatively small difference between the two AUC values indicates that the model did not experience overfitting and has good generalization ability. The binary classification process was carried out using the optimal threshold through the ROC-Youden index method. The best threshold value obtained was 0.582, so observations that generated a probability higher than this limit would be classified as relatively high-poverty areas (1) and otherwise as relatively low-poverty areas (0).

**NN Modeling with NM Simplex**

The NM simplex algorithm is used to optimize weights and biases in NN classification without using any derivatives. The NN model is built using one hidden layer with a ReLU activation function and a single output neuron with a Sigmoid activation function. The objective function used is the average log-loss from 5-fold cross-validation. The NM simplex algorithm is executed for each hidden neuron configuration ranging from 2 to 20 neurons, with initial weights randomly initialized using a normal distribution $N(0,0.1)$. This optimization process is carried out iteratively with a maximum limit of 300 iterations per neuron configuration.

Unlike the Adam model, the number of hidden neurons in the NM simplex model was limited to 2–20. This limitation was adopted because the NM simplex directly optimizes all trainable weights and biases without gradient information, making the optimization dimension equal to the total number of trainable parameters. Increasing the number of hidden neurons substantially enlarges the simplex size and computational burden. Therefore, a relatively compact network architecture was used to ensure computational feasibility while maintaining representative classification performance. A dropout layer was not incorporated into the NM simplex model because the optimization algorithm evaluates deterministic objective function values at each simplex vertex. The stochastic behavior introduced by dropout would change the objective function between evaluations, potentially affecting the stability of the simplex search.

Weights and biases are treated as a single large parameter vector whose optimal values are sought through direct evaluation of the loss function. Unlike gradient-based approaches, the NM simplex does not require hyperparameters such as learning rate or batch size, because this algorithm works with the entire data directly to calculate the loss function value. The optimization process is carried out by changing the position of the parameters through reflection, expansion, contraction, and reduction steps to find the combination of weights and biases that produces the lowest global loss value. The presence of activation functions such as ReLU still affects the shape of the loss surface that must be explored by the algorithm, even though this method does not rely on gradients. This can impact the algorithm's ability to reach the global minimum, especially if the loss surface has many local minima.

The model configuration with two neurons in the hidden layer produced the lowest average log-loss of 0.653 based on cross-validation evaluation. The model was applied to the test data to evaluate predictive performance. Based on probabilistic results, the AUC value obtained on the test data was 0.735 and on the training data was 0.719. The small difference between the two indicates that the model does not experience overfitting and has good generalization ability. The classification threshold was determined through the ROC curve by maximizing the statistic value of Youden's index (Attwood et al., 2014). This approach resulted in an optimal threshold value of 0.402, so prediction probabilities exceeding this value will be classified as relatively high-poverty areas (1) and otherwise as relatively low-poverty areas (0). The NM simplex is relatively rarely applied in NN optimization, but this modeling was able to provide fairly good predictive performance results despite limited stability.

**Model Comparison**

The best NN model will be chosen based on the best evaluation scores. Evaluation is carried out on the performance of both models, the results of different optimization approaches on the test data, based on evaluation metrics such as AUC, accuracy, precision, sensitivity, specificity, and F1-score. The results in Table 5 explain that the model with Adam optimization consistently produces better performance across all evaluation metrics. The Adam model produces higher AUC and F1-score values. This indicates a better ability to classify regencies/cities as relatively high-poverty or relatively low-poverty areas while maintaining a balance between precision and sensitivity. The higher specificity in Adam indicates better capability in recognizing relatively low-poverty areas; however, the sensitivity value for identifying relatively high-poverty areas remains limited.

**Table 5**. Comparison of evaluation metrics

| Evaluation Metrics | Optimization Method | |
|---|---|---|
| | Adam | NM Simplex |
| AUC | 0.818 | 0.735 |
| Accuracy | 0.777 | 0.738 |
| Precision | 0.771 | 0.692 |
| Sensitivity | 0.643 | 0.643 |
| Specificity | 0.869 | 0.803 |
| F1-score | 0.701 | 0.667 |

The performance of the NM simplex model is lower on most metrics compared to Adam. The weakness of this optimization approach can be explained by its nature of not using derivative information, thus not leveraging the gradient structure of the loss function. This method is indeed suitable for use when the function's derivatives are difficult to compute, but its performance tends to be less efficient when applied to functions that are very complex in shape, especially if the function is non-convex or has many local minima (Cerdà et al., 2015). This method also results in slow and less stable convergence in optimizing large-scale NN parameters (Silva et al., 2018). Computation time can be much longer without guaranteeing better results. These results emphasize the importance of selecting an optimization method that matches the characteristics of the data and model structure. Derivative-based methods such as Adam provide more stable and efficient performance, making them more feasible to use in supporting data-driven policies. Therefore, NN models with Adam are a more viable choice to be implemented to support data-based social policy classification systems at the regional level.

**Best Model**

The best model in this study was obtained from the NN algorithm with the Adam optimization method and produced the best prediction threshold value of 0.582. The threshold value was determined based on the maximum Youden index from the ROC curve to obtain an optimal balance between sensitivity and specificity. This threshold selection provides a classification boundary that balances the identification of relatively high-poverty and relatively low-poverty areas. The test data AUC value of 0.818 indicates that the model has good overall classification ability, with optimal performance between the true positive rate and the false positive rate.

A fairly good performance was achieved in the evaluation of the test data model with an accuracy of 0.777, indicating that 77.7% of the test observations were correctly classified. The sensitivity value of 0.643 indicates that the model successfully identified 64.3% of actual relatively

high-poverty areas, while the specificity value of 0.869 demonstrates its ability to correctly identify relatively low-poverty areas. The F1-score of 0.701 indicates a balanced trade-off between precision and sensitivity in handling classification errors. The confusion matrix results show that out of 103 test observations, 53 relatively low-poverty areas were correctly classified (true negative) and 27 relatively high-poverty areas were correctly identified (true positive). Meanwhile, 8 relatively low-poverty areas were misclassified as relatively high-poverty areas (false positive), and 15 relatively high-poverty areas were misclassified as relatively low-poverty areas (false negative).

Despite achieving the best overall performance, the Adam model still has limitations in identifying relatively high-poverty areas, as indicated by the sensitivity value of 0.643. The false negative cases are important to consider because some relatively high-poverty areas may not be identified correctly and could receive less attention in targeted interventions. Therefore, model performance should not only be evaluated based on overall accuracy but also by considering the trade-off between sensitivity and specificity.

The threshold of 0.582 represents the statistically optimal trade-off between sensitivity and specificity based on the Youden index. However, in practical policy implementation, threshold selection may depend on the objective of the classification system. A lower threshold could be considered when the priority is to identify more potential relatively high-poverty areas, although this may increase false positive classifications by labeling some relatively low-poverty areas as relatively high-poverty areas.

## CONCLUSION

This study compared the performance of the Adam and NM simplex optimization methods for NN-based relative poverty classification across regencies/cities in Indonesia. The findings demonstrate that gradient-based optimization provides a more effective and computationally efficient training strategy than derivative-free optimization for this classification task. Although the NM simplex can optimize neural network parameters without gradient information, its performance is constrained by the high-dimensional optimization space, resulting in lower overall classification performance under feasible computational settings. These findings provide empirical evidence that the choice of optimization method substantially influences neural network performance and highlight the suitability of gradient-based optimization for regional relative poverty classification. This study also demonstrates that the comparison should be interpreted as a practical evaluation under computationally feasible settings rather than a strictly controlled comparison using identical network architectures. Future research may investigate multiclass relative poverty classification, evaluate scalable derivative-free optimization methods or deeper neural network architectures under comparable computational budgets, and compare Adam with other metaheuristic optimization methods, such as genetic algorithms and particle swarm optimization (PSO).

## REFERENCE

Aggarwal, S. (2023). LSTM based anomaly detection in time series for United States exports and imports. *Munich Personal RePEc Archive*, (117149).

Agustin, M., & Prahasto, T. (2012). Penggunaan jaringan syaraf tiruan backpropagation untuk seleksi penerimaan mahasiswa baru pada jurusan Teknik Komputer di Politeknik Negeri Sriwijaya. *Jurnal Sistem Informasi Bisnis*, *2*, 89–97. https://doi.org/10.21456/vol2iss2pp089-097

Alkhairi, P., & Windarto, A. P. (2023). Classification analysis of back propagation-optimized CNN performance in image processing. *Journal of Systems Engineering and Information Technology (JOSEIT)*, *2*(1), 8–15. https://doi.org/10.29207/joseit.v2i1.5015

Allo, R. F. A. R., & Fevriera, S. (2025). Efek kuantitas dan kualitas manusia terhadap kemiskinan di Kaltara. *Jurnal Ekonomika*, *16*(1), 1–17. https://doi.org/10.35334/sdr9y076

Allorerung, P. P., Erna, A., Bagussahrir, M., & Alam, S. (2024). Analisis performa normalisasi data untuk klasifikasi k-nearest neighbor pada dataset penyakit. *Jurnal Informatika Sunan Kalijaga*, *9*(3), 178–191. https://doi.org/10.14421/jiska.2024.9.3.178-191

Apriza, I., Izza, M. F., Sebastian, M. N., & Paduloh, P. (2025). Optimasi produksi konveksi IZZA Collection antara dua desain t-shirt menggunakan metode programa linear dan diselesaikan dengan metode simplex. *HUMANITIS: Jurnal Humaniora, Sosial, Dan Bisnis*, *3*(1), 194–200.

Attwood, K., Tian, L., & Xiong, C. (2014). Diagnostic thresholds with three ordinal groups. *Journal of Biopharmaceutical Statistics*, *24*(3), 608–633. https://doi.org/10.1080/10543406.2014.888437

Bantis, L. E., Nakas, C. T., & Reiser, B. (2019). Construction of confidence intervals for the maximum of the Youden index and the corresponding cutoff point of a continuous biomarker. *Biometrical Journal*, *61*, 138–156. https://doi.org/10.1002/bimj.201700107

Baudin, M. (2010). Nelder-mead user's manual. *Consortium Scilab – DIGITEO.*

Bishop, C. M. (2006). *Pattern recognition and machine learning* (M. Jordan, J. Kleinberg, & B. Scholkopf, Eds.). Springer.

BPS Kabupaten Enrekang. (2024). *Indeks pembangunan manusia tahun 2024 Kabupaten Enrekang* (Vol. 9). BPS Kabupaten Enrekang.

Cerdà, V., Cerdà, J. L., & Idris, A. M. (2015). Optimization using the gradient and simplex methods. *Talanta*, *148*, 641–648. https://doi.org/10.1016/j.talanta.2015.05.061

Dantzig, G. B., Orden, A., & Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, *5*(2), 183–195. https://doi.org/10.2140/pjm.1955.5.183

Faritz, M. N., & Soejoto, A. (2020). Pengaruh pertumbuhan ekonomi dan rata-rata lama sekolah terhadap kemiskinan di Provinsi Jawa Tengah. *JUPE (Jurnal Pendidikan Ekonomi)*, *8*(1), 16–21. https://doi.org/10.26740/jupe.v8n1.p15-21

Goodfellow, I., Bengio, Y., & Courvile Aaron. (2016). Deep learning. In *Healthcare Informatics Research* (Number 4). The MIT Press.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations*, 1–15. http://arxiv.org/abs/1412.6980

Kolda, T. G., Lewis, R. M., & Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *Society for Industrial and Applied Mathematics*, *45*(3), 385–482. https://doi.org/10.1137/S003614450242889

Kurniati, I., Winarno, H., & Yuliyanti, D. (2023). Pemanfaatan algoritma neural network untuk predictive analytic angka buta huruf di Indonesia. *Jurnal Elekto & Informatika Swadharma (JEIS)*, *3*(2), 94–104. https://doi.org/10.56486/jeis.vol3no2.364

Li, J. (2024). Area under the ROC Curve has the most consistent evaluation for binary classification. *Plos One*, 1–28. https://doi.org/10.1371/journal.pone.0316019

Mohtasham, F., Nazari, S. S. H., Pourhoseingholi, M. A., Kavousi, K., & Zali, M. R. (2026). Hybrid feature-selection and diversity-guided stacking framework for interpretable ensemble learning: Application to COVID-19 mortality prediction. *Plos One*, *21*(4). https://doi.org/10.1371/journal.pone.0341198

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*, 308–312. https://doi.org/10.1093/comjnl/7.4.308

Ozaki, Y., Yano, M., & Onishi, M. (2017). Effective hyperparameter optimization using Nelder-Mead method in deep learning. *IPSJ Transactions on Computer Vision and Applications*, *9*(20), 1–12. https://doi.org/10.1186/s41074-017-0030-7

Prasetyawan, P., Ahmad, I., Borman, R. I., Ardiansyah, A., Pahlvei, Y. A., & Kurniawan, D. E. (2018, October 4). Classification of the period undergraduate study using back-propagation neural network. *International Conference on Applied Engineering (ICAE)*.

Prasetyo, V. R., Mercifia, M., Averina, A., Sunyoto, L., & Budiarjo, B. (2022). Prediksi rating film pada website IMDB menggunakan metode neural network. *Network Engineering Research Operation (NERO)*, *7*(1), 1–8. http://repository.ubaya.ac.id/id/eprint/41805

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(6086). https://doi.org/10.1038/s41598-024-56706-x

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of ADAM and beyond. *International Conference on Learning Representations*, 1–23. https://doi.org/10.48550/arXiv.1904.09237

Sakti, A. I., Saputra, L., Suhendra, H., Halim, N., Alviari, I., Ilham, M. R. N., Putri, M. H. N., & Dalimunthe, D. Y. (2024). Implementasi artificial neural network (ANN) dalam memprediksi nilai tukar rupiah terhadap dolar Amerika. *Euler : Jurnal Ilmiah Matematika, Sains Dan Teknologi*, *12*(2), 124–130. https://doi.org/10.37905/euler.v12i2.26654

Saputri, A. A. (2023). Implementasi algoritma backpropagation, LVQ, dan multilayer perceptron untuk klasifikasi tingkat kemiskinan Kabupaten/Kota di Indonesia [Skripsi]. Universitas Hamzanwadi.

Saragih, T. H., & Huda, N. (2022). Jaringan syaraf tiruan backpropagation dengan adaptive moment estimation untuk klasifikasi penyakit Covid-19 di Kalimantan Selatan. *Epsilon: Jurnal Matematika Murni Dan Terapan*, *16*, 162–172.

Si, T., He, F., Li, P., & Gao, X. (2023). Tri-modality consistency optimization with heterogeneous augmented images for visible-infrared person re-identification. *Neurocomputing*, *523*, 170–181. https://doi.org/10.1016/j.neucom.2022.12.042

Silva, K. G.-E., Aloise, D., Souza, S. X.-D., & Mladenovic, N. (2018). Less is more: Simplified nelder-mead method for large unconstrained optimization. *Yugoslav Journal of Operations Research*, *28*(2), 153–169. https://doi.org/10.2298/YJOR180120014G

Suhendra, C. D., & Saputra, A. C. (2020). Penentuan parameter learning rate selama pembelajaran jaringan syaraf tiruan backpropagation menggunakan algoritma genetika. *Jurnal Teknologi Informasi*, *4*(2), 202–212. https://doi.org/10.47111/jti.v14i2.1141

Suhendra, C. D., & Wardoyo, R. (2015). Penentuan arsitektur jaringan syaraf tiruan backpropagation (bobot awal dan bias awal) menggunakan algoritma genetika. *Indonesian Journal of Computing and Cybernetics Systems*, *9*(1), 77–88.

Szandala, T. (2020). Review and comparison of commonly used activation functions for deep neural networks. *Bio-Inspired Neurocomputing*, 203–224. https://doi.org/10.1007/978-981-15-5495-7_11

Windarto, A. P. (2017). Implementasi JST dalam menentukan kelayakan nasabah pinjaman kur pada Bank Mandiri mikro serbelawan dengan metode backpropagation. *Jurnal Sains Komputer & Informatika (J-SAKTI*, *1*(1), 12–32. http://ejurnal.tunasbangsa.ac.id/index.php/jsakti

Wulandari, G., Febriyanti, N. A., Anwar, K., & Matematika, F. (2022). Pemodelan persentase penduduk miskin di Indonesia menggunakan regresi probit dan regresi logistik. *Prosiding Seminar Nasional Matematika, Statistika, Dan Aplikasinya*, 354–368.