

Optimizing Neural Network Weights and Biases Using Particle Swarm Optimization for Classification Task

Made Agus Dwiputra¹, I Gede Pasek Suta Wijaya², I Gde Wirarama Wedashwara³
agusdwiputram@gmail.com¹, gpsutawijaya@unram.ac.id², wirarama@unram.com³
Master of Information Technology, Universitas Mataram, Nusa Tenggara Barat, Indonesia

ABSTRACT

The current digital era greatly demands reliable automatic classification systems, especially to handle large and increasingly complex data volumes. One attractive alternative is the Particle Swarm Optimization (PSO) algorithm, which is recognized for its effective global search. Nevertheless, the performance of PSO for training artificial neural networks with complex or large-scale data is still uncertain. The primary purpose of this research is to create and assess a classification engine based on MLP, which uses a PSO algorithm to generate weights and biases. The assessment was made on three different types of data - dummy data, Iris data, and Sasak script images. For the dummy and Iris datasets, the model successfully achieved 100% accuracy, demonstrating the effectiveness of the PSO-MLP approach on simpler data. However, the results differed significantly for the more extensive and complex image dataset, where the model experienced a drastic decline in performance. In the image classification test with 6 classes, the model with one hidden layer achieved 71% accuracy, while the model with two hidden layers only reached 56%. For 12-class classification, accuracy dropped to 35% and 25%, respectively, and for 18 classes, the model achieved only 27% and 7%. These results indicate that while PSO is effective in optimizing perceptron weights and biases for smaller and simpler datasets, its ability to handle large-scale image classification with increasing complexity remains limited. Therefore, there is a need for optimization strategies to enhance the accuracy of optimization for more complex data.

Keywords: Particle Swarm Optimization (PSO); Classification Machine; Weight; Biases; Neural Network

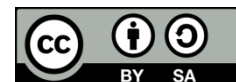
Article Info

Received : 12-02-2025

This is an open-access article under the [CC BY-SA](#) license.

Revised : 21-04-2025

Accepted : 30-06-2025



Correspondence Author:

Made Agus Dwiputra
Master of Information Technology,
Universitas Mataram,
Nusa Tenggara Barat, Indonesia.
Email: agusdwiputram@gmail.com

1. INTRODUCTION

The evolution of classification technology, especially in the areas of artificial intelligence and machine learning, is advancing exceptionally quickly today. The development of classification systems is one of the core elements of artificial intelligence and machine learning because it is fundamental to the requirements of pattern recognition, data classification and automating decision making processes. In the pursuit of creating efficient and precise. The increasing number of studies employing diverse datasets with varying characteristics has encouraged the exploration of adaptive classification models, as implemented in this research.

One increasingly popular approach involves the use of population-based optimization algorithms. To build an optimal classification model, experimentation with various types of data is essential. This study begins with simple data, namely dummy data, which in certain classification cases can produce high accuracy and serves as an initial step to evaluate the model's workflow and detect potential errors[1]. Subsequently, the Iris

dataset is utilized, consisting of 150 flower samples classified into three categories: Iris setosa, Iris versicolor, and Iris virginica, based on the length and width of petals and sepals[2].

The third dataset consists of images of Sasak script characters. Feature extraction in this context is performed using the Discrete Cosine Transform (DCT). This approach is effective in minimizing redundancy while emphasizing the most significant features. DCT has been shown to enhance classification performance by emphasizing relevant frequency information in images[3]. After feature extraction, classification is performed using a Multilayer Perceptron (MLP) with a feedforward architecture, which has the ability to model matrix multiplication between inputs and weights, and applies activation functions to map values into non-linear forms. We look at how important CNN methods are changing the way medical images are sorted. Researchers are making things more accurate and efficient by concentrating on data preprocessing, transfer learning, new architectures, and explainability. These models are already helping doctors figure out what's wrong with people who have lung and skin problems, diabetic retinopathy, brain tumors, bleeding, leukemia, and breast cancer[4].

To tackle the optimization challenge at hand, we will employ the Particle Swarm Optimization (PSO) algorithm. PSO is a sophisticated, population-based optimization technique that draws inspiration from the collective behaviors exhibited by certain organisms, such as birds and fish, in their pursuit of resources.[5]. PSO offers the advantage of exploring the solution space globally, thus avoiding local minima traps, and accelerating convergence, making it particularly suitable for optimizing the weights and biases of neural networks.[6]. As highlighted addressing preprocessing challenges particularly in cloud-prone regions like Indonesia through morphological operations significantly enhances the accuracy of classification models that rely on multispectral image inputs[7].

A number of previous studies have shown the effectiveness of Particle Swarm Optimization (PSO) in improving the accuracy and efficiency of classification systems. For example, in the classification of Chronic Kidney Disease (CKD), the combination of PSO with SVM and AdaBoost achieved an accuracy of up to 99.50% [6]. In brain tumor classification, PSO was applied during the segmentation stage and combined with transfer learning-based CNNs (e.g., AlexNet and Inception-V3), achieving up to 99.0% accuracy on the BRATS-2017 and BRATS-2018 datasets [5]. Other studies look at how to make coral reef image classification better by using Particle Swarm Optimization (PSO) for feature selection, Histogram of Oriented Gradients (HOG) for feature extraction, and Support Vector Machine (SVM) for classification. The research examined datasets of healthy, bleached, and deceased corals, categorized into training, validation, and test sets. This method greatly improved the accuracy of classification, going from 79.11% with the original SVM model to 85.44% with the PSO-optimized SVM[8].

However, several studies using the PSO algorithm do not always show good results. For instance, in text data classification tasks such as SMS spam and sentiment analysis, the best accuracy was achieved without PSO 98% for SMS spam and 84.03% for sentiment analysis while with PSO, the accuracy dropped to 65.26% and 71.23%, respectively [4]. In a meta-analysis it was found that although genetic algorithms increased the average output of neural networks by 3.44%, the effect was not statistically significant, highlighting the importance of careful algorithmic tuning in neural network optimization strategies[9].

Particle Swarm Optimization (PSO) is now the most accepted meta-heuristic optimization algorithm to use for neural networks and is particularly predominant when exploring. Recently, PSO has shown great success in augmenting performance of neural networks by exploring hyperparameters and alternatives to neural network architectures. For instance, [10] proposed a multi-level PSO for optimizing Convolutional Neural Networks (CNN), which resulted in improved classification accuracy on complex image datasets[10]. Similarly,[11] introduced CPSO-Net, a continuous PSO-based deep learning framework for hyperspectral image classification, achieving competitive performance compared to traditional optimization methods[11]. In addition, PSO variants have been explored for CNN optimization without velocity equations, providing efficient solutions for image classification with reduced computational costs[12]. Other approaches combine PSO with extreme learning machines (ELM) for hyperspectral image analysis, further confirming the adaptability of PSO in handling high-dimensional data[13]. More recently, multi-objective metaheuristic approaches including PSO have been investigated to optimize CNNs for balancing accuracy and complexity, indicating the potential of PSO in developing scalable and generalized classification models[14]. A recent study similarly emphasized the role of optimization techniques like genetic algorithms in refining network configurations, showing improved RMSE values when applied to time-series forecasting models such as RNNs. This knowledge makes it even more clear how important hyperparameters are for optimizing sets, which can be very different for different datasets and classification problems[15].

Research has demonstrated that the Particle Swarm Optimization (PSO) algorithm performs effectively with straightforward datasets, including numerical data and established datasets such as the Iris dataset. However, its efficacy tends to diminish when applied to more intricate datasets, particularly those involving image data with a high number of classes. This research is an extension of the author's previous study[16], in which Particle Swarm Optimization (PSO) was used to optimize a Multilayer Perceptron (MLP)

model without hidden layers, and comparisons were made with CNN and SVM methods on simple datasets such as Iris and dummy data. The study demonstrated that PSO was effective for classifying simple data but did not explore the impact of varying network architectures or its application to more complex image data. Therefore, this current study presents a further development by introducing one and two hidden layers into the MLP architecture and testing the model on the Sasak script image dataset. The aim is to evaluate the limitations of PSO in optimizing neural networks for the classification of more complex and multiclass datasets. This occurred because the network architecture used had not been further explored, so the potential of PSO in addressing multi-class image classification problems had not been fully optimized. In addition, research that specifically develops a classification machine for Sasak script images using PSO as the main optimization method is still very limited, thus this study carries a novelty value and can contribute to the development of classification methods for traditional image data with high complexity.

As a powerful non-derivative optimization algorithm, Particle Swarm Optimization (PSO) is particularly effective for addressing non-linear and complex optimization challenges. In light of these advantages, this study utilizes PSO to enhance the training of a classification model based on an advanced Multilayer Perceptron (MLP) architecture. The experimental scope is systematically expanded by evaluating the model's performance across varying class configurations specifically 6, 12, and 18 classes to explore the limitations of PSO-based classification when applied to datasets of increasing complexity, and to assess the adaptability and generalizability of the PSO approach across different data characteristics. Building upon the previous research, this study is designed to reflect its primary objective to develop and evaluate a classification machine based on a Multilayer Perceptron (MLP) optimized using Particle Swarm Optimization (PSO), while also identifying the limitations and potential improvements of this approach when applied to complex datasets.

2. RESEARCH METHOD

The research methodology in this study is designed to classify image data using a neural network model optimized by Particle Swarm Optimization (PSO). The overall workflow includes several main stages, starting with data collection, followed by preprocessing, and continuing to feature extraction utilizing the Discrete Cosine Transform (DCT). The extracted features are then selected and passed through a feedforward process to the neural network structure.

This model uses a Multi-Layer Particle Swarm Optimization (ML-PSO) architecture. The training process begins with initialization of particle positions and velocities, followed by iterative updates based on fitness evaluation. During this process, input data is effectively propagated through one or more hidden layers. Training ends after reaching a predetermined maximum number of iterations or after reaching a specified error threshold. Model performance is then assessed using standard classification metrics, including accuracy, precision, recall, and F1 score.

The process is comprehensively illustrated in Figure 1, which details the step-by-step workflow of the proposed classification approach that integrates neural networks with Particle Swarm Optimization (PSO).

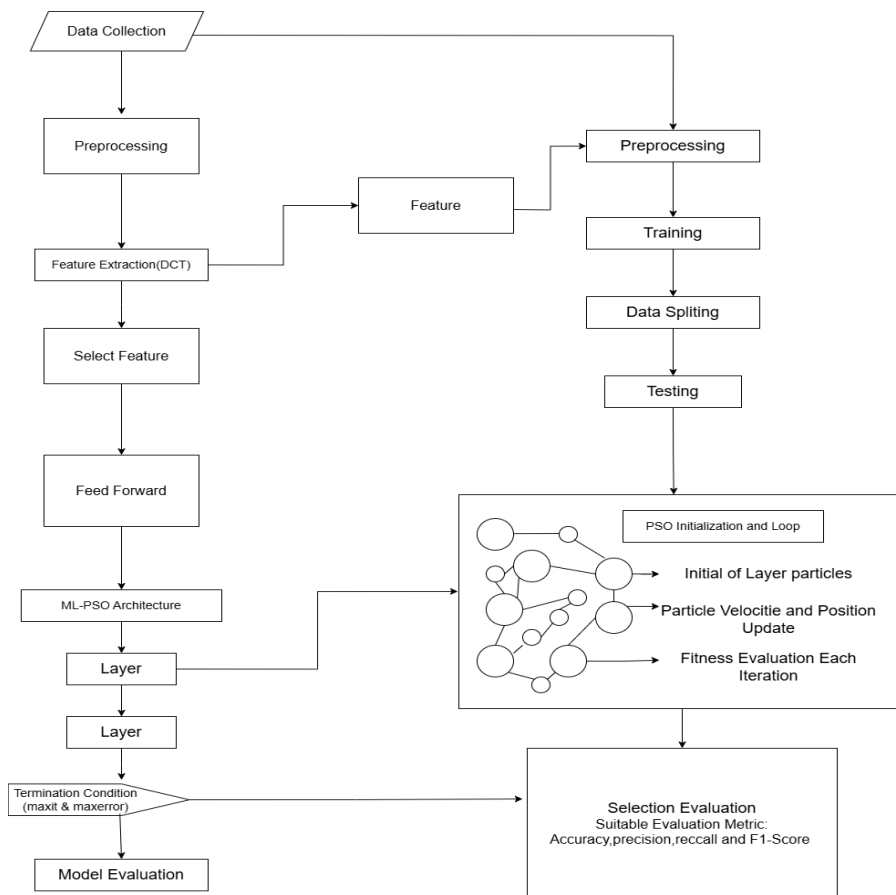


Figure 1. Flow Classification Machine Model

2.1. Dataset

This Research use three different datasets to evaluate the generalization capability of the proposed classification model:

1. Dummy Data

Manually generated data with simple numeric patterns, useful for baseline evaluation and model debugging that can be seen in the Figure 2.

```

[[0.8736303 0.12462297 0.01813066]
 [0.99155305 0.36300439 0.23412158]
 [0.63668365 0.40702516 0.54726649]
 [0.95208912 0.02228722 0.69382771]
 [1.09209963 1.05771963 0.63569161]
 [0.80880044 1.26390046 0.68082922]
 [0.56120452 1.08903958 0.81189573]
 [1.36469162 0.92173876 1.28186539]
 [0.95826879 1.07176646 0.65911766]]
[0. 0. 0. 0. 1. 1. 1. 1. 1.]
  
```

Figure 2. Dummy Dataset

2. Iris Dataset

The Iris dataset employed in this study is based on previous research comparing the classification performance of the K-Nearest Neighbor and Random Forest algorithms. The dataset includes 150 flower samples, each sample described with four traits: sepal length, sepal width, petal length, and petal width. These flower samples are usually categorized into three different species of flower: Iris setosa, Iris versicolor, and Iris virginica—as established in prior studies[17]. The image of those samples can be seen in the Figure 3.

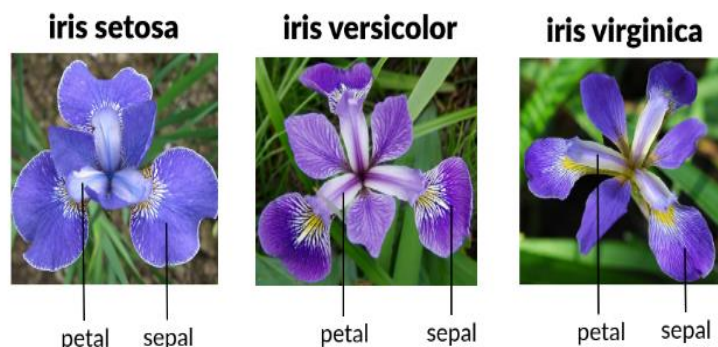


Figure 3. Iris Dataset

3. Sasak Script Image Dataset

The image dataset of Sasak script utilized in this study originates from previous research involving the implementation of Convolutional Neural Networks (CNN) for character recognition on Android platforms, which reported a high classification accuracy of 99.31% [18]. The dataset is organized systematically into three subsets; the first subset contains 18 classes, the second subset contains 12 classes, and the third subset contains 6 classes. Each class contains 600 handwritten character samples. Participants in the data collection represented a variety of educational backgrounds, including elementary school students, junior high school students, senior high school students, and university students. Participants were instructed to write Sasak characters on A4-sized paper that was formatted with a 4x4 cm grid, so that the placement of the characters was consistent. All character samples were digitized and saved in PNG file format and were saved at a resolution of 128×128 pixels. Examples of images of each Sasak script image can be seen in the Figure

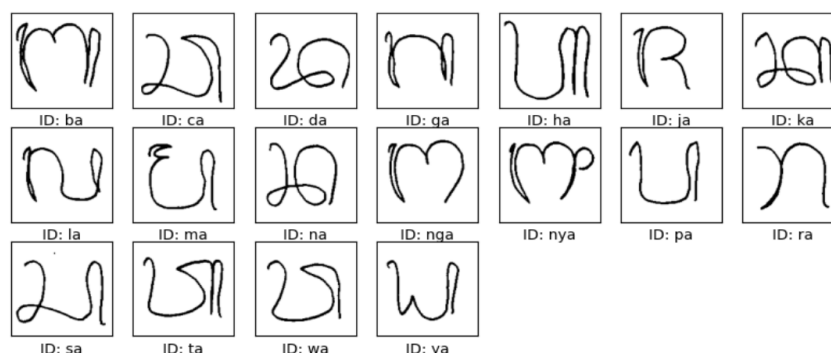


Figure 4. Sasak Script Image Dataset

2.2. Labeling

In the labeling stage, each of the three datasets used a different labeling approach. The dataset used in this analysis contained three columns as input variables, plus another variable column that indicated the dataset as a form of identifier. The Iris dataset includes measurements of sepal length, sepal width, petal length, petal width, and species of the iris. Specifically, the species, which were Setosa, Versicolor, and Virginica, have been formatted as numerical labels, specifically 0, 1, and 2. The conversion of the categorical labels into numerical labels allows the model to effectively optimize the weights and biases based on differences between classes.

In this study, the labeling process for the Sasak character dataset was conducted by assigning a unique numerical identifier to each class of character images during the dataset loading phase. As we loaded the images from the folder, we determined the class labels based on the folder names. Each subfolder represented a specific character class. Each image was then associated with an integer label corresponding to its respective class. The labels were subsequently converted into a one-hot encoded format utilizing the `to_categorical()` function from the Keras library. This transformation is crucial for multi-class classification tasks, as it effectively converts integer class labels into binary matrices that represent class membership. For example, an image belonging to class 3 out of 6 total classes would be represented as a binary vector [0, 0, 0, 1, 0, 0]. This one-hot encoded label matrix was then used as the target output (y) during the training and evaluation of the classification model.

2.3. Preprocessing

The preprocessing stages applied to each dataset in this study depend on the characteristics of the data used. For the dummy dataset, preprocessing was minimal. The data were directly initialized as NumPy arrays and split into features and labels without further normalization or standardization in contrast, the Iris dataset has undergone a series of preprocessing steps, which included the separation of features from labels, the conversion of categorical labels into a numerical format, and the division of the dataset into training and testing sets.

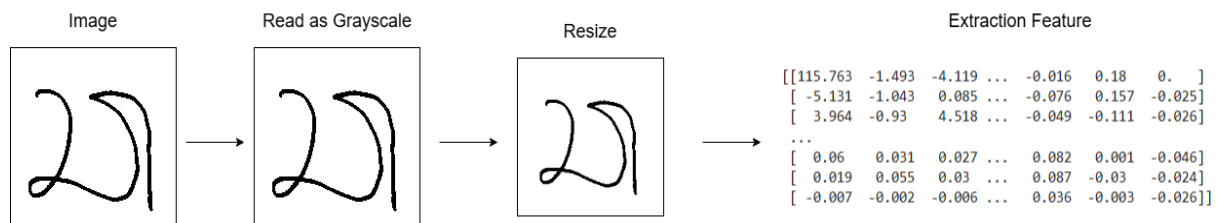


Figure 5. Preprocessing Flow

Meanwhile, in Figure 5 for the Sasak character image dataset, the preprocessing process begins with the original Sasak script image, which is first converted to grayscale to simplify color information and reduce computational complexity. The grayscale image undergoes resizing to a standardized dimension, specifically 128×128 pixels, to maintain consistent input sizes for the model. Subsequently, feature extraction is conducted utilizing the Discrete Cosine Transform (DCT), which effectively converts the image from the spatial domain to the frequency domain. This transformation highlights critical features while minimizing redundancy. The outcome of the DCT is a matrix of frequency coefficients that can be further refined, for instance, through the application of a Zigzag scanning technique, to produce a compact and representative feature vector. This feature vector then serves as the input for the neural network-based classification model.

2.4. Extraction Fature DCT

Feature extraction utilizing the Discrete Cosine Transform (DCT) is a prominent technique in the field of image processing, particularly in the context of classification tasks. The DCT serves to convert spatial image data into the frequency domain, enabling the capture of essential patterns and textures. This transformation effectively concentrates the image's energy within a limited number of low-frequency components. As a result, it facilitates dimensionality reduction while preserving vital information, thereby enhancing both the efficiency and accuracy of classification models[19].

Recent studies have demonstrated the efficacy of DCT in various applications. For instance, in the classification of brain tumors using MRI images, DCT-based feature extraction combined with machine learning models like Support Vector Machines (SVM) achieved high classification accuracy, underscoring DCT's capability to capture discriminative features in medical imaging. Similarly, in the assessment of tobacco leaf quality, DCT was utilized to extract texture features, leading to a classification accuracy of 90%, which highlights its effectiveness in agricultural product evaluation[20][21]. In the context of the provided code, DCT is employed to extract features from grayscale images of Sasak characters. Each image is first converted to grayscale to ensure uniformity in pixel intensity values. The Discrete Cosine Transform (DCT) is employed to convert the image into the frequency domain. To effectively manage the dimensionality of the resulting data, a ZigZag scanning pattern is utilized to select a subset of the most significant DCT coefficients. This method allows for the extraction of the most informative features from the image. These features are subsequently used as inputs for training a neural network model, which is optimized through Particle Swarm Optimization (PSO). This integrated approach harnesses the strengths of DCT in feature extraction and PSO in optimization, leading to high classification accuracy in the Sasak character recognition task.

2.5. Select Feature

In this process, which takes place after feature extraction using the Discrete Cosine Transform (DCT), only the most significant coefficients from the transformed matrix are retained to construct a compact and informative feature vector. The selection process is essential for minimizing the dimensionality of the input data, which in turn reduces computational complexity. This approach effectively preserves the key characteristics of the images that are necessary for accurate classification.

In the context of this study, 64 DCT coefficients were selected from each image to serve as the feature representation. This decision was based on an effort to capture more discriminative information compared to a previous experiment that utilized only 32 features[16]. The increase in the number of features was designed to enhance the model's capability to differentiate between classes by offering a more comprehensive set of

descriptors for each input. The effects of this modification were further assessed during the model testing phase to evaluate any improvements in classification performance. This process is vital for reducing computational complexity while maintaining essential information from the image. The selected features are subsequently passed to the feedforward stage of the neural network.

2.6. Feed Forward

The feedforward process is the basic computational operation of the Multilayer Perceptron (MLP) model where input features pass through several layers of neurons. Each neuron in the network computes the weighted sum of its inputs, along with a bias term and applies a nonlinear activation function, generally 'Rectified Linear Unit' (ReLU), to produce its output. This process will allow the network to learn complex, nonlinear relationships in the data, which is crucial to obtain high accuracy levels when classifying images.

In this study, the feedforward computation is pair with the Particle Swarm Optimization (PSO) algorithm for the purpose of optimizing the weights and biases of the network and to supplant traditional gradient-based optimization methods, like backpropagation. The Particle Swarm Optimization (PSO) has very powerful benefits including improved global search abilities and more robust against local minima, which improves the performance of training a neural network.

A primary focus of this study is yet to explore depth of network architectures, and specifically how many hidden layers would be needed to improve their classification performance. For the past few studies, the classification model was fit without hidden layers, using only direct connections between inputs and outputs. While this avoided fitting a complex network with hidden layers, it also restricted the network's ability to consider higher level abstractions of the data[16].

To address this limitation, the present study presents and compares two architectural configurations one with a single hidden layer and another with two hidden layers. The goal is to determine how increased depth in combination with PSO (Particle Swarm Optimisation) will improve the ability of the ANN (Artificial Neural Network) to generalise and classify the image data accurately.

Findings from our experiments indicate that the model was able to capture complex relationships more effectively with additional hidden layers. However, the presence of more than one hidden layer did not always improve model performance. In some cases, excessive depth, especially with high-dimensional image data, increased the amount of complexity and can result in overfitting or slow down convergence. These findings reinforce the importance of tuning architecture for neural networks, even with state-of-the-art optimization interventions like Particle Swarm Optimization (PSO).

2.7. ML-PSO Architecture

In this study, the classification engine was designed using three different datasets as inputs. Once the eventuality of pre-processing was reached, the data was processed through a feedforward structure in an ANN (artificial neural network). The structure calculated the dot product of the input matrix and the weights, followed by the use of the Rectified Linear Unit (ReLU) activation function. The ReLU activation function allows negative values to become zeros and positive values to remain, allowing the model to learn non-linear relationships in the data, providing relative representational power. The neural network had two hidden layers, which provided additional representational power but also increased the model's generalization ability when solving complex tasks, what are mostly classification tasks [22].

This study used the Particle Swarm Optimization (PSO) algorithm to optimize the model parameters, including the weights and biases. PSO is a population-based optimization method, inspired by the behaviors of natural systems, such as birds flocking or fish schooling and is suitable for optimizing complex real-life problems. In a PSO approach, each particle in the swarm represents a potential solution and moves in space based on both their own best position (local best), and the best position of the swarm (global best). The velocity and position of each particle are updated using the following equations:

$$\begin{aligned} v_i^{t+1} &= w \cdot v_i^t + c_1 \cdot r_1 \cdot (pbest - x_i^t) + c_2 \cdot r_2 \cdot (gbest - x_i^t) \\ x_i^{t+1} &= x_i^t + v_i^{t+1} \end{aligned} \quad (1)$$

where w is the inertia weight, c_1 and c_2 are cognitive and social learning factors respectively, and r_1 and r_2 are random values between 0 and 1. This process continues until a maximum number of iterations is reached, or convergence occurs, with the final solution based on the global best position (gbest) which defines the best set of weights and biases for the classification model. In this work, the PSO algorithm was employed to optimize weights and biases of an MLP architecture.

The PSO parameters were chosen as follows: A population size of 500, a max iteration value of 1500, an initial inertia weight (w) of 0.7 with a damping factor (w_{damp}) of 0.99, and both learning coefficients, personal and social, (c_1 and c_2) were set to 1.5. The parameter values were chosen based on a variety of

previous works and initial trials, which showed stability during the convergence process. The selected values of $w = 0.7$ and $w_{damp} = 0.99$ were intended to maintain a balance between exploration and exploitation, while $c1$ and $c2 = 1.5$ provided a balanced contribution between individual and collective experience.

To further adapt the algorithm to the increased dataset size and complexity, especially in the classification of Sasak script images, parameter ranges were adjusted. Specifically, the population size (nPop) was varied between 500 and 1500, and it tended to increase proportionally with the number of hidden layers and dataset complexity. Likewise, the maximum number of iterations was extended up to 2000 to allow sufficient convergence in more complex scenarios. By contrast, for simpler datasets such as Iris and dummy data, the default parameters adopted from previous studies [11] were retained, since they already yielded stable and optimal results.

The parameters of PSO, population size (nPop), inertia weight (w), inertia damping ratio (w_{Damp}), and the learning coefficients ($c1$ and $c2$) were optimized based on the features of each dataset. The population size (nPop) specifies how many particles are traveling through the search space. The inertia weight (w) controls how much a particle's past velocity affects its movement, while the damping factor (w_{Damp}) gradually reduces this influence to help the swarm focus on better solutions over time. The learning coefficients ($c1$ and $c2$) balance the influence of individual experience and the collective knowledge of the swarm, ensuring stable and reliable optimization. Larger datasets required more particles and iterations to explore a wider search space and ensure adequate exploration, whereas smaller datasets had lower dimensionality and required fewer particles to achieve optimal results more efficiently.

The hybrid approach of merging PSO and neural networks has been successful in many different classification applications. For example, developed a PSO based method to optimize neural network structure for image classification. These results produced comparable accuracies with significantly reduced computational costs than traditional methods [23].

2.8. Hidden Layer

The architecture includes one or more hidden layers, which are represented as consecutive Layer blocks in the diagram. These layers provide the MLP with the capability to represent the input data in increasingly abstract forms. In the context of this study, experiments were conducted using both a single hidden layer and a two-layer configuration to investigate how network depth impacts classification accuracy when combined with PSO-based optimization. Surprisingly, adding hidden layers did not always improve performance. This was particularly evident in experimenting with complex image data. This speaks to the importance of getting the architecture right when using PSO.

2.9. Termination Condition

In this study, the stopping criteria for the ML-PSO model are determined by two parameters: the maximum number of iterations (maxIt) and the minimum error threshold (maxError). The optimization process will terminate when either of these conditions is met. The maxIt parameter is implemented to ensure that the optimization does not continue indefinitely, which is especially relevant when working with complex datasets, such as Sasak script images that comprise 3,600 samples and 6 classes. In these cases, the reduction in error can be gradual, making convergence difficult to achieve within a limited number of iterations.

2.10. Testing

The MLP-PSO model was validated as part of the testing stage against a test dataset (x_{test} , y_{test}) that had not be utilized during training as test datasets allow for analysis of the model's ability to generalize findings to unseen instances. Therefore, the optimum weights and biases obtained in training were applied to the test inputs, along with predictions made using the forward propagation function. The model's performance was subsequently assessed through standard classification metrics, including accuracy, precision, recall, and F1-score, all derived from the confusion matrix. These metrics provide a comprehensive evaluation of the model's effectiveness under different class distributions and complexities. Furthermore, this evaluation highlights the performance gap between simple datasets (dummy and Iris) and the more complex multiclass dataset (Sasak script), thereby supporting the hypotheses regarding dataset complexity and classification challenges.

2.11. Training

The classification model training integrates a feedforward artificial neural network (ANN) architecture optimized by Particle Swarm Optimization (PSO). After data preprocessing, the input passes through the network layers, where each neuron calculates the weighted sum of inputs followed by the Rectified Linear Unit (ReLU) activation function to introduce non-linearity. Instead of traditional gradient-based

optimization, PSO a population-based metaheuristic inspired by social behavior is employed to optimize the network's weights and biases.

In particle swarm optimisation (PSO), multiple candidate solutions (particles) concurrently search the solution space based on their own best experience (local best), and the best experience of the swarm (global best). Each particle's fitness will continuously be assessed using a chosen loss/error function at every iteration. Each particle updates its velocity and position based on the movement rules for PSO which gives a good balance of exploration and exploitation during the search process. This iterative process continues through to some form of convergence, such as a maximum number of iterations or error level.

The use of multiple hidden layers in the ANN enhances the ability to model complex data patterns and improves classification accuracy. Combining ANN with PSO allows effective training by circumventing local minima issues commonly faced in gradient descent, leading to improved convergence and robustness across various datasets.

This hybrid training approach has demonstrated effectiveness in recent studies optimizing neural networks with swarm intelligence algorithms, achieving significant improvements in classification tasks [24][25].

2.12. Evaluation

The evaluation of the system occurs at this stage to confirm it operates correctly and achieves the intended goals. The evaluation also identifies potential shortfalls and enhancement opportunities for the system. Evaluation methodology is based upon the confusion matrix, which allows for the analysis of the classification performance. The performance of the model is evaluated utilizing standard performance metrics: accuracy (Equation 1), recall (Equation 2), and precision (Equation 3). All metrics are established for a number of pre-defined test cases to provide a comprehensive evaluation of the model's predictive potential.

$$Accuracy = \frac{\text{Number of correctly classified data points}}{\text{Total number of data points}} \quad (2)$$

$$Precision = \frac{\text{Number of correctly data points in a specific class}}{\text{Total number of predicted data points for that class}} \quad (3)$$

$$Recall = \frac{\text{Number pf correctly classified data points in a specific class}}{\text{Total number of actual data points in that class}} \quad (4)$$

3. RESULTS AND DISCUSSION

Implementing various scenarios was essential to determine the effectiveness of the classification model in this research study. This research used three separate datasets, and for the Sasak character image dataset, additional testing scenarios were needed as follows.

3.1 Testing on Dummy Dataset

The testing results of the classification engine model, built using an artificial neural network with two hidden layers and optimized by Particle Swarm Optimization (PSO), showed highly satisfactory outcomes, as presented in the following Table 1 and graph illustrates how the error decrease as the number of iterations increase in the following Figure 6.

Table 1. Classification Report Dummy Dataset

Model Optimization	Precision	Recall	F1-Score	Accuracy
ML-PSO	100%	100%	100%	100%
CNN	11%	33%	17%	33%
SVM	83%	67%	67%	67%

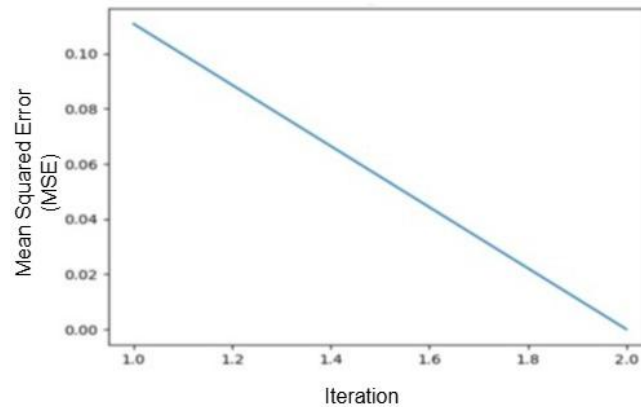


Figure 6. Test Result Graph on Dummy Dataset

The results for the dummy dataset indicate a rapid convergence, with the error showing a sharp decrease starting from the second iteration. This early drop in error demonstrates that the PSO algorithm is highly effective in optimizing the MLP weights and biases when applied to simple, linearly separable data. The simplicity of the dummy dataset allows PSO to quickly reach near-optimal solutions with minimal iteration, avoiding the need for extensive search across the solution space. The experimental results indicate that MLP optimized with PSO achieved the highest accuracy at 100%, confirming its effectiveness in learning from small, structured datasets through global weight optimization. This outstanding performance demonstrates the capability of the MLP+PSO hybrid model to identify patterns accurately and generalize well, even when trained on a limited number of samples. The use of PSO as a global optimizer allows the model to explore the weight space more effectively than traditional gradient-based methods, thereby avoiding local minima and improving convergence stability. Moreover, the MLP's architectural configuration, which incorporated hidden layers, significantly contributed to its ability to recognize nonlinear relationships within the data, thereby boosting its classification performance in this experimental context.

The SVM model followed with an accuracy of 67%, demonstrating its well-known strength in handling low-dimensional and linearly separable data. While SVM can adequately handle situations with a small number of features, its performance was not optimal in this instance due to its sensitivity to the distribution of features and the choice of kernel. Unlike MLP+PSO, SVM lacks the capacity to internally adapt its feature representations, which limits its flexibility when modeling more complex or nonlinear patterns, even in simple datasets.

In contrast, CNNs were not effective, obtaining a performance of only 33%, which can be associated with the relatively small size of the dataset and the lack of spatial structure in the dataset, both of which inhibit the generally well-functioning nature of CNNs. CNNs are constructed to process data with spatial hierarchies, like images which have specific pixel relationships and local patterns. In the absence of spatial dependencies, CNN architectures have a much harder time being able to learn meaningful features, which makes the use of convolutional layers not functional.

The results confirm that MLP combined with PSO is a very efficient classification method for simple, low-sample, non-image datasets. The ability to achieve perfect accuracy with such datasets is impressive and highlights the importance of matching model architecture and optimization strategies to the dataset characteristics. While traditional models like SVM can also generate acceptable results, the limitations of SVM in handling nonlinearity and accommodations of flexibility become apparent when compared to the combinations of neural networks training methods enhanced by PSO. Meanwhile, convolutional models such as CNNs require more appropriate data types to fully leverage their representational power, and therefore are not suitable for the type of dataset used in this experiment.

3.2 Testing on Iris Dataset

An experiment was conducted to develop a classification model using the Iris dataset by implementing a neural network architecture with two hidden layers. The network was composed of 4 input neurons, 8 neurons in the first hidden layer, 16 neurons in the second hidden layer, and 3 output neurons that represented the three classes of Iris flowers. ReLU activation functions were used in the hidden layers, and a softmax function in the output layer to produce class probabilities. The model's training was performed using the Particle Swarm Optimization (PSO) algorithm which optimized all weights and biases in the network. The results of the iteration to achieve the cost function value in this experiment can be seen in Figure 7.

Table 2. Classification Report Iris Dataset

Model	Precision	Recall	F1-Score
Optimization			
ML-PSO	98%	98%	98%
CNN	98%	98%	98%
SVM	98%	98%	98%

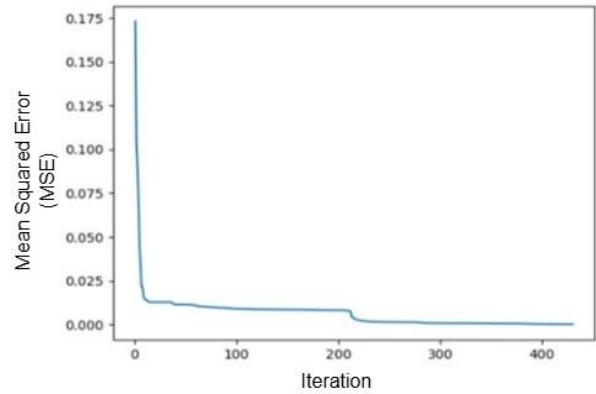


Figure 7. Test Result Graph on Iris Dataset

All three models CNN, SVM, and MLP optimized with PSO achieved the same accuracy of 98% on the Iris dataset that can be seen in the Table 2. The convergence plot for the Iris dataset reveals that the error value begins to decrease more consistently after approximately 400 iterations. This suggests that the PSO algorithm required an initial exploration phase before gradually refining the weights and biases toward an optimal solution. The relatively simple and structured nature of the Iris dataset likely contributed to the successful convergence, highlighting the suitability of PSO for handling low-dimensional and linearly separable classification tasks. This result indicates that the dataset, characterized by its well-organized and linearly separable feature space, is relatively straightforward and can be effectively classified using both traditional machine learning algorithms and more advanced neural network architectures.

The SVM model proved to be effective and efficient in dealing with tabular data that exhibited clear class separation, providing a lightweight and rapid training process. Although CNNs are primarily designed for spatial data like images, they adapted well to the reshaped tabular input, likely because of the dataset's low dimensionality and structured characteristics.

The MLP-PSO model, on the other hand, proved that swarm-based optimization techniques can successfully train neural networks without relying on gradient-based methods like backpropagation. Even though the resulting accuracy was the same, the MLP-PSO approach exhibited advantages in terms of robustness and global search capability, which may be particularly beneficial for more complex datasets or scenarios prone to local minima.

This result confirms that the classification machine developed using PSO is not only capable of matching the performance of traditional and deep learning methods on simple datasets, but also offers flexibility and potential scalability to handle more challenging classification tasks. In terms of computational complexity, SVM remains the most efficient choice for this specific case, but MLP-PSO presents a compelling alternative where gradient-free optimization or architectural flexibility is needed.

3.3 Testing on SasakChar Image Dataset

To evaluate the performance of the proposed classification model, a series of experiments were conducted using the Sasak script image dataset under various testing scenarios. These scenarios encompassed variations in the number of classes, specifically 6, 12, and 18 classes, with each class containing 600 images. Additionally, the architecture of the neural network was changed by implementing both 1 and 2 hidden layers to evaluate how model complexity affects classification performance. The results from each experimental setup are displayed in a tabular format, including evaluation metrics such as accuracy, precision, recall, and F1-score. These findings aim to provide a thorough overview of the model's capacity to identify patterns in Sasak script image data across different complexity levels.

3.1.1 Testing With One Hidden Layer

The evaluation was conducted using several scenarios on the model by employing a single hidden layer with different class configurations: 6, 12, and 18 classes. This testing aimed to observe the performance of the classification model across varying numbers of image dataset classes using a single hidden layer. The results of the classification report in this experiment can be seen in Table 3.

Table 3. Classification Report SasakChar Dataset One Hidden Layer

Class	Precision	Recall	F1-Score	Accuracy
6	71%	71%	70%	71%
12	25%	36%	26%	35%
18	22%	28%	21%	27%

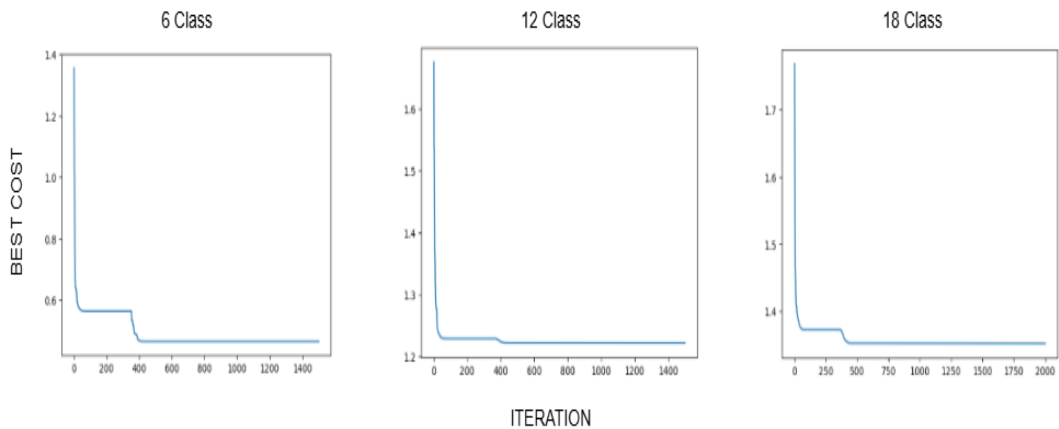


Figure 8. Test Result Graph on SasakChar Dataset with 1 Hidden Layer

In Figure 8, The convergence plots for the Sasak script image dataset, tested across 6, 12, and 18-class configurations, show that the error values did not reach zero in any scenario. However, every experiment had a consistent pattern that demonstrated a decreasing trend of error with additional iterations. This suggests that even though the PSO algorithm was able to progressively decrease error, it still struggled with completely optimizing the model with respect to high-dimensional image data. The average number of iterations until convergence in these experiments ranged from 1400 to 2000. The experiments were initially conducted with an artificial neural network model with one hidden layer. The evaluation was conducted across three different scenarios focused on image classification tasks with varying class counts of 6, 12, and 18, with each class comprising 600 images. The findings indicated that an increase in the number of classes notably decreased classification accuracy.

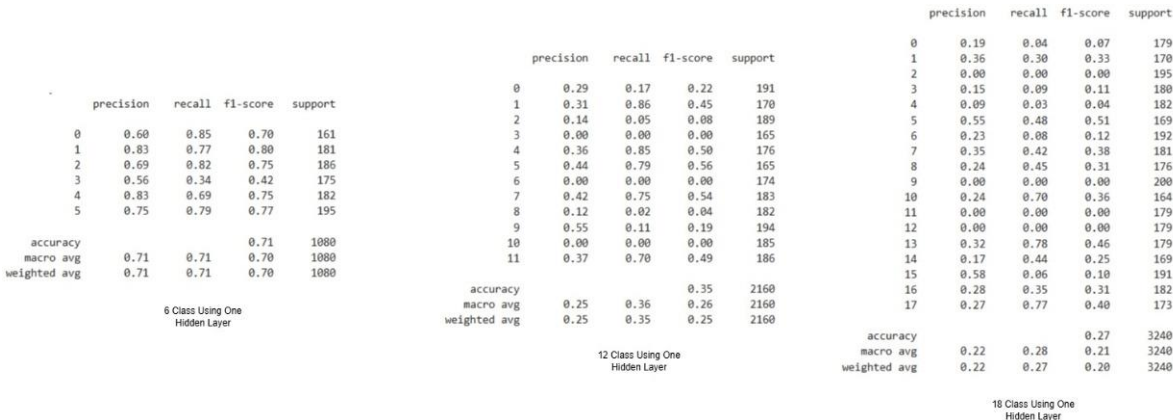


Figure 9. Confusion Matrix on SasakChar Dataset with 1 Hidden Layer

From the confusion matrix in Figure 9 the 6-class scenario, the model achieved an accuracy of 71%, indicating that DCT-based feature representation was relatively effective for simpler classification tasks. However, when the number of classes was increased to 12, the accuracy dropped significantly to 35%, and further decreased to 27% for the 18-class scenario. This decline suggests that the single hidden layer

architecture has limited capability in distinguishing features when dealing with more complex, multi-class datasets.

These findings indicate that a network with a single hidden layer can be ineffective for classification tasks with a large number of classes. Typically, as the number of classes and the amount of data increases, the overall model performance decreases as seen in the lower precision scores, recall scores, F1-scores, and overall accuracy. This suggests that the representational capacity of a single hidden layer is less capable of capturing the complex patterns required for accurate multi-class classification. Therefore, to further examine the model’s performance under these more challenging conditions, the next phase of experiments employed a neural network architecture with two hidden layers, aiming to determine whether deeper networks can enhance classification accuracy and provide better generalization for larger and more complex datasets.

3.2.1 Testing With Two Hidden Layer

The evaluation was conducted using several scenarios on the model by employing two hidden layer with different class configurations: 6, 12, and 18 classes. This testing aimed to observe the performance of the classification model across varying numbers of image dataset classes using two hidden layer. The results of the classification report in this experiment can be seen in Table 4.

Table 4. Classification Report SasakChar Dataset Two Hidden Layer

Class	Precision	Recall	F1-Score	Accuracy
6	48%	56%	51%	56%
12	14%	26%	15%	25%
18	8%	7%	4%	7%



Figure 9. Test Result Graph on SasakChar Dataset with 2 Hidden Layer

In Figure 9, you can see the best cost results obtained for each class tested and each class was tested with a different number of iterations to obtain optimal results in the test. The Mean Squared Error (MSE), defined as the average squared difference between the target values and the PSO outputs, consistently decreased until the iteration limit of 1500 to 2000 was reached in the different class trials. In the subsequent experiment, the neural network architecture was enhanced by adding a second hidden layer, with the goal of improving classification performance in more complex scenarios. The same three class configurations were used 6, 12, and 18 classes with each class containing 600 images, consistent with the earlier setup.

					precision	recall	f1-score	support		
					0	0.10	0.11	0.10	179	
					1	0.09	0.25	0.13	170	
					2	0.00	0.00	0.00	195	
					3	0.00	0.00	0.00	180	
					4	0.00	0.00	0.00	182	
					5	0.00	0.00	0.00	169	
					6	0.05	0.01	0.01	192	
					7	0.00	0.00	0.00	181	
					8	0.15	0.14	0.15	176	
					9	0.20	0.10	0.13	200	
					10	0.50	0.01	0.02	164	
					11	0.06	0.58	0.10	179	
					12	0.20	0.01	0.01	179	
					13	0.00	0.00	0.00	179	
					14	0.04	0.08	0.06	169	
					15	0.02	0.01	0.01	191	
					16	0.00	0.00	0.00	182	
					17	0.03	0.01	0.02	173	
					accuracy			0.07	3240	
					macro avg	0.08	0.07	0.04	3240	
					weighted avg	0.08	0.07	0.04	3240	
					18 Class Using One Hidden Layer					
					precision	recall	f1-score	support		
					0	0.00	0.00	0.00	191	
					1	0.00	0.00	0.00	170	
					2	0.00	0.00	0.00	189	
					3	0.15	0.65	0.25	165	
					4	0.00	0.00	0.00	176	
					5	0.45	0.80	0.57	165	
					6	0.21	0.94	0.34	174	
					7	0.41	0.63	0.49	183	
					8	0.50	0.09	0.16	182	
					9	0.00	0.00	0.00	194	
					10	0.00	0.00	0.00	185	
					11	0.00	0.00	0.00	186	
					accuracy		0.25	2160		
					macro avg	0.14	0.26	0.15	2160	
					weighted avg	0.14	0.25	0.15	2160	
					12 Class Using Two Hidden Layer					
					precision	recall	f1-score	support		
					0	0.50	0.84	0.63	161	
					1	0.63	0.52	0.57	181	
					2	0.52	0.55	0.54	186	
					3	0.00	0.00	0.00	175	
					4	0.57	0.67	0.61	182	
					5	0.62	0.77	0.69	195	
					accuracy		0.56	1080		
					macro avg	0.47	0.56	0.51	1080	
					weighted avg	0.48	0.56	0.51	1080	
					6 Class Using One Hidden Layer					

Figure 10. Confusion Matrix on SasakChar Dataset with 2 Hidden Layer

Surprisingly, the results with two hidden layers showed lower accuracy across all scenarios compared to the single hidden layer configuration. For the 6-class model accuracy was 56% which is 15% worse than the previous model. In the same vein, the 12-class and 18-class model accuracies were 25% and 7% respectively both underperforming the single layered model. There are multiple factors that can be inferred from the confusion matrices that explain this change in effectiveness. Firstly, having added multiple hidden layers increased the model complexity and if it isn't a one to one classification, one can assume that overfitting occurred for certain classes and did not generalize well to new data. Second, the optimization process using PSO became more challenging in a higher-dimensional search space, making it harder to converge toward optimal weight and bias values. Thirdly, the confusion matrices indicate that misclassifications were mixed across a wide breadth of classes; this suggests that the deeper architecture had difficulty extracting consistently discriminative features. Hence, instead of increasing accuracy, the more complex architecture compromised the stability of the learning process which ultimately led to a decline in overall performance. A major limitation is that PSO often falls into local minima during optimization in large searching landscapes [26].

These results illustrate that a deeper network will not lead to better performance by the mere act of increasing depth with no tuning of hyperparameters or regularization. In this situation, the additional complexity likely caused overfitting or convergence issues, especially since Particle Swarm Optimization (PSO) has been used in training and thus may need to be further tuned when the model is deeper. Even with the implementation of a two-hidden-layer configuration, the results showed a further decline in precision, recall, F1-score, and accuracy across individual classes, indicating that the deeper model did not yield improved classification outcomes.

Future studies should consider the use of additional methods such as dropout, batch normalization, or different activation functions to improve the model's capacity for effective learning of higher-dimensional class distributions. This concept represents the anchor of the final conclusion for this study, which provides a summary on the impacts of architectural depth and optimization strategies on classification results. This study has several limitations. First, although the use of the Sasak script dataset with a large number of classes provides a realistic representation of data complexity, it poses a significant challenge for the PSO-MLP model, as performance decreases with the increasing number of classes and feature dimensions. Second, the model is limited to an MLP architecture optimized with PSO, without the integration of advanced feature extraction methods, data augmentation, or deep learning architectures such as CNNs, which are more suitable for large-scale image data. Third, the experiments were only conducted on networks with one and two hidden layers, leaving the potential of deeper architectures or hybrid approaches with other algorithms unexplored.

3.4 Testing On JAFFE (Japanese Female Facial Expression) Dataset

In addition this test uses a image datasets were used: facial expression images from the JAFFE (Japanese Female Facial Expression) Database [27], and Sasak script character images. The JAFFE dataset contains 210 grayscale images with a resolution of 256×256 pixels, classified into 7 fundamental facial expressions. Graph illustrates how the error decrease as the number of iterations increase in the following Figure 10.

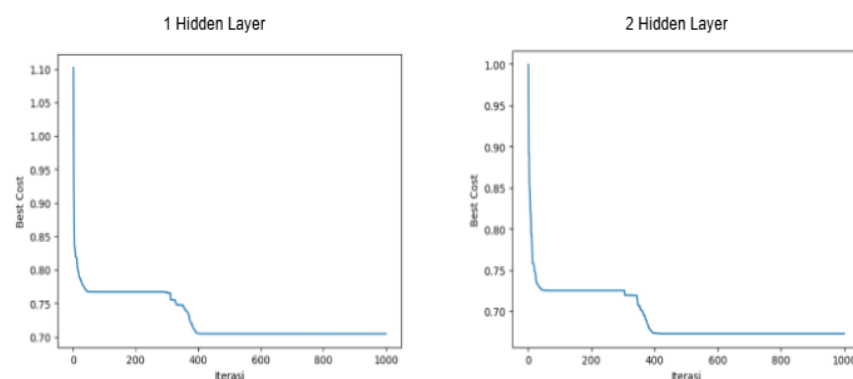


Figure 10. Test Result Graph on JAFFE Dataset with 1 and 2 Hidden Layer

Experiments were conducted using a Multilayer Perceptron (MLP) architecture with both one and two hidden layers, optimized using the Particle Swarm Optimization (PSO) algorithm. Both experiments were performed under the same parameter settings: a population size of 200, a maximum of 500 iterations, and an error threshold of 0.001.

The results of the study show that the model trained on the Sasak character dataset consistently achieved higher classification accuracy compared to the model trained on the facial expression dataset with a

comparison of the number of datasets and the fewest classes from the Sasak script image testing. The model's accuracy on the facial expression dataset was only 22% when testing with a single hidden layer; however, the accuracy rose to 32% when using two hidden layers. Due to the small sample size and the high visual complexity between facial expression classes, the model finds it challenging to identify unique patterns, which may be the cause of the poor accuracy on facial images. On the other hand, the larger number of samples and lower intra-class variation in the Sasak character dataset enabled the model to learn more effective feature representations, resulting in significantly better classification performance overall.

4. CONCLUSION

This study successfully developed a classification model using a Multilayer Perceptron (MLP) trained with the Particle Swarm Optimization (PSO) algorithm, addressing the demand for efficient automatic classification systems in handling datasets of varying complexity. Initially, the model, which was previously built without hidden layers, was enhanced by integrating hidden layers to evaluate performance across three types of datasets: dummy data, the Iris dataset, and Sasak script image data. Experimental results showed that the inclusion of hidden layers significantly improved performance on simpler datasets, with the model achieving 100% accuracy on dummy data and 98% on the Iris dataset, demonstrating the strength of the PSO-MLP combination in handling linearly separable and structured data.

However, when tested on the more complex Sasak script image dataset, the model's performance declined with increased data complexity and class count. For 6, 12, and 18 classes, the model's accuracies with a single hidden layer were 71%, 35%, and 27%, respectively. But the accuracy dropped dramatically to 56%, 25%, and 7% when two hidden layers were used. These findings imply that PSO struggles to generalize well to high-dimensional, large-scale image data, even though it is successful at optimizing neural network parameters for simpler datasets. This highlights a critical limitation of PSO in complex classification tasks, as raised in the introduction and abstract.

This study expands upon the previous work[16] by conducting a more in-depth analysis of the impact of varying the number of hidden layers in the MLP architecture on classification performance across datasets of different complexity levels. The experimental findings suggest that adding more hidden layers does not inherently enhance performance, especially when applied to the complex Sasak script image dataset. These results highlight PSO's shortcomings in large-scale image classification tasks and imply that improvements might be required, which could involve integrating CNNs or using hybrid optimization strategies. As a result, this study not only highlights the PSO-MLP model's performance limitations but also offers a strong basis for creating more sophisticated classification techniques in subsequent research.

The findings provide valuable insights for the development of automatic classification systems, especially for the preservation and digitalization of traditional scripts such as the Sasak script. Furthermore, the results highlight the strengths of PSO in handling simpler datasets, which can be leveraged in applications requiring efficient optimization with limited computational resources. This study was limited to the use of MLP with PSO optimization, without incorporating advanced feature extraction methods, data augmentation, or deep learning techniques such as Convolutional Neural Networks (CNNs). As a result, the model's performance on complex multi-class image data was constrained.

Future studies should concentrate on examining hybrid optimization techniques that combine PSO with other metaheuristic algorithms like Differential Evolution or Genetic Algorithms, or with gradient-based learning. Furthermore, investigating convolutional neural networks (CNNs), dimensionality reduction, and feature extraction strategies may help increase classification accuracy in image-based datasets such as the Sasak script. These improvements are essential for creating classification systems that are more resilient, scalable, and flexible so they can handle challenging real-world data situations.

Future research could specifically look into combining PSO with CNNs to optimize CNN architectures' initial weights, hyperparameters, or filter configurations. By better utilizing the spatial and hierarchical nature of image data, this hybrid approach may be able to overcome the optimization stagnation seen in deep MLPs trained exclusively with PSO. Given the high-dimensionality and variability of the Sasak script image dataset, such an approach may lead to improved generalization and accuracy, especially in multi-class scenarios.

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

REFERENCES




- [1] T. Jung and J. Kim, "A New Support Vector Machine for Categorical Features," *Expert Syst. Appl.*, vol. 229, p. 120449, 2023, doi: <https://doi.org/10.1016/j.eswa.2023.120449>.
- [2] Y. J. Krishna, G. Murari, M. Murali, and P. Raghu, "IRIS Flower Species Prediction Using Machine Learning and Web Based Interactive Tool for Non Technical Users," pp. 1–6, 2025.

- [3] K. Su *et al.*, “DetViT: Discrete Cosine Transform meet vision transformers,” *Neural Networks*, vol. 172, no. October 2023, 2024, doi: 10.1016/j.neunet.2024.106139.
- [4] C. Chen, N. A. Mat Isa, and X. Liu, “A Review of Convolutional Neural Network Based Methods for Medical Image Classification,” *Comput. Biol. Med.*, vol. 185, p. 109507, 2025, doi: <https://doi.org/10.1016/j.compbiomed.2024.109507>.
- [5] T. O. Aro, H. B. Akande, K. S. Adewole, K. M. Aregbesola, and M. B. Jibrin, “Enhanced Textual Data Classification using Particle Swarm Optimization Algorithm,” *J. ICT Dev. Appl. Res.*, vol. 2, no. April 2020, pp. 1–14, 2020.
- [6] M. Ali *et al.*, “Brain Tumor Detection and Classification Using PSO and Convolutional Neural Network,” *Comput. Mater. Contin.*, vol. 73, no. 3, pp. 4501–4518, 2022, doi: 10.32604/cmc.2022.030392.
- [7] K. Tang and C. Meng, “Particle Swarm Optimization Algorithm Using Velocity Pausing and Adaptive Strategy,” *Symmetry*, vol. 16, no. 6. 2024. doi: 10.3390/sym16060661.
- [8] J. C. Bastiaans, J. Hartojo, R. A. Pramunendar, and P. N. Andono, “Evaluating the Impact of Particle Swarm Optimization Based Feature Selection on Support Vector Machine Performance in Coral Reef Health Classification,” *IJNMT (International J. New Media Technol.)*, vol. 11, no. 2, pp. 90–99, Jan. 2025, doi: 10.31937/ijnmt.v11i2.3761.
- [9] L. I. LI, “Application of Artificial Neural Networks and Genetic Algorithm in Optimization of Concrete Shear Wall Design,” *Int. J. Interact. Des. Manuf.*, vol. 18, no. 7, pp. 4775–4785, 2024, doi: 10.1007/s12008-024-01739-9.
- [10] P. Singh, S. Chaudhury, and B. K. Panigrahi, “Hybrid MPSO-CNN: Multi-level Particle Swarm Optimized Hyperparameters of Convolutional Neural Network,” *Swarm Evol. Comput.*, vol. 63, p. 100863, 2021, doi: <https://doi.org/10.1016/j.swevo.2021.100863>.
- [11] X. Liu, C. Zhang, Z. Cai, J. Yang, Z. Zhou, and X. Gong, “Continuous Particle Swarm Optimization-Based Deep Learning Architecture Search for Hyperspectral Image Classification,” *Remote Sensing*, vol. 13, no. 6. 2021. doi: 10.3390/rs13061082.
- [12] D. Elhani, A. C. Megherbi, A. Zitouni, F. Dornaika, S. Sbaa, and A. Taleb-Ahmed, “Optimizing convolutional neural networks architecture using a modified particle swarm optimization for image classification,” *Expert Syst. Appl.*, vol. 229, p. 120411, 2023, doi: <https://doi.org/10.1016/j.eswa.2023.120411>.
- [13] A. Ye, X. Zhou, and F. Miao, “Innovative Hyperspectral Image Classification Approach Using Optimized CNN and ELM,” *Electronics*, vol. 11, no. 5. 2022. doi: 10.3390/electronics11050775.
- [14] A. Rashno and S. Fadaei, “Convolutional Neural Networks Optimization Using Multi-Objective Particle Swarm Optimization Algorithm,” *Inf. Sci. (Ny)*, vol. 689, p. 121443, 2025, doi: <https://doi.org/10.1016/j.ins.2024.121443>.
- [15] W. Hussain *et al.*, “Ensemble Genetic and CNN Model-Based Image Classification by Enhancing Hyperparameter Tuning,” *Sci. Rep.*, vol. 15, no. 1, pp. 1–24, 2025, doi: 10.1038/s41598-024-76178-3.
- [16] L. Abualigah, “Particle Swarm Optimization: Advances, Applications, and Experimental Insights,” *Computers, Materials & Continua*, vol. 82, no. 2. 2025. doi: 10.32604/cmc.2025.060765.
- [17] Y. Zhou, “Study for Iris Classification Based on Multiple Machine Learning Models,” *Highlights Sci. Eng. Technol.*, vol. 23, pp. 342–349, 2022, doi: 10.54097/hset.v23i.3620.
- [18] Y.-T. Jou, H.-L. Chang, and R. M. Silitonga, “Sustainable Optimization of the Injection Molding Process Using Particle Swarm Optimization (PSO),” *Applied Sciences*, vol. 15, no. 15. 2025. doi: 10.3390/app15158417.
- [19] Z. Zhao, C. Yang, Z. Qiu, and Q. Wu, “Discrete Cosine Transform-Based Joint Spectral–Spatial Information Compression and Band-Correlation Calculation for Hyperspectral Feature Extraction,” *Remote Sensing*, vol. 16, no. 22. 2024. doi: 10.3390/rs16224270.
- [20] Q. Zeng, B. Hui, Z. Liu, Z. Xu, and M. He, “A Method Combining Discrete Cosine Transform with Attention for Multi-Temporal Remote Sensing Image Matching,” *Sensors*, vol. 25, no. 5. 2025. doi: 10.3390/s25051345.
- [21] R. V. Nahari, A. S. Editya, and R. Alfita, “Ekstrasi Fitur Daun Tembakau Berbasis Discrete Cosine Transform (DCT),” *J. Appl. Informatics Comput.*, vol. 4, no. 1, pp. 8–12, 2020, doi: 10.30871/jaic.v4i1.1756.
- [22] M. Maimouni, A. E. M. Badr, and M. and Bouya, “RFID Network Planning Using a New Hybrid ANNs-Based Approach,” *Conn. Sci.*, vol. 34, no. 1, pp. 2265–2290, Dec. 2022, doi: 10.1080/09540091.2022.2115011.
- [23] I. Classification, “Texture Filter Optimization Using Particle Swarm Optimization for Efficient Lung Image Classification,” *J. Popul. Ther. Clin. Pharmacol.*, vol. 30, no. 15, pp. 67–75, 2023, doi: 10.47750/jptcp.2023.30.15.007.
- [24] S. Lankford and D. Grimes, “Neural Architecture Search Using Particle Swarm and Ant Colony Optimization,” *CEUR Workshop Proc.*, vol. 2771, pp. 229–240, 2020.
- [25] S. Rahnamayan, P. Mazaheri, and A. Asilian Bidgoli, “Designing Artificial Neural Network Using Particle Swarm Optimization: A Survey,” M. A. Aceves-Fernández, Ed., Rijeka: IntechOpen, 2022. doi: 10.5772/intechopen.106139.
- [26] S. Aote, M. M. Raghuwanshi, and L. Malik, “Brief Review on Particle Swarm Optimization: Limitations & Future Directions,” *Int. J. Comput. Sci. Eng.*, vol. 2, pp. 196–200, Jan. 2013.
- [27] M. Kamachi, M. Lyons, and J. Gyoba, “The Japanese Female Facial Expression (JAFFE) Database,” Available <http://www.kasrl.org/jaffe.html>, Jan. 1997.




BIOGRAPHIES OF AUTHORS

Made Agus Dwiputra completed his Bachelor's degree in Informatics Engineering from the University of Mataram, Indonesia, with a specialization in artificial intelligence. His research interests encompass intelligent systems, cloud computing, and optimization algorithms in machine learning. During his studies, he participated in the Kampus Merdeka program, focusing on cloud computing, where he developed projects utilizing the Google Cloud Platform. In addition, he made significant contributions to student organizations, achieving over 90% completion of the assigned work programs during his tenure. He possesses programming skills in Python and Java. He can be contacted via email at agusdwiputram@gmail.com.



I Gede Pasek Suta Wijaya    is a full-time lecturer in the Informatics Engineering Study Program at the Faculty of Engineering, University of Mataram. He holds a Master's degree in Information Engineering from Gadjah Mada University and a Doctorate from Kumamoto University, Japan. His research interests span artificial intelligence, digital image processing, machine learning, and intelligent systems. In addition to teaching core informatics courses, he actively supervises students, participates in applied research, and contributes to the development of outcome-based curricula. He is also engaged in both national and international research collaborations and scientific publications. He can be contacted via email at gpsutawijaya@unram.ac.id.



Wirarama Wedashwara    completed his doctoral program in Computer Science and Design Engineering at Yamaguchi University in 2016. Prior to this, he pursued a Master's degree in Energy Management and a Bachelor's degree in Electrical Engineering at Udayana University. His research interests encompass the application of evolutionary computation and fuzzy databases to big data, the Internet of Things (IoT), and parallel processing. His work primarily utilizes evolutionary fuzzy association rule mining derived from genetic network programming. Additionally, his research extends to text classification using genetic programming. He can be contacted via email at wirarama@unram.ac.id.