**Assessment 2**

**Enhancing Neural Network Performance with Particle Swarm Optimization (PSO)**

**Unit Coordinator : Umair Ullah Tariq**

**Class : Tutorial 1**

**Date : 11 April 2025**

**Student Number Student Name**

**12223508 Furkan Tekkartal**

**COIT29224**

**Evolutionary Computation**

**2025 Term 1**

Table of Contents

[0](#_Toc195335024)

[1 Introduction 2](#_Toc195335025)

[2 Execution Instructions 3](#_Toc195335026)

[3 Dataset and Preprocessing 4](#_Toc195335027)

[4 Neural Network and Optimization Methods 5](#_Toc195335028)

[4.1 Neural Network Design 5](#_Toc195335029)

[4.2 Grid Search Approach 5](#_Toc195335030)

[4.3 Particle Swarm Optimization (PSO) Approach 6](#_Toc195335031)

[5 Results and Comparison 7](#_Toc195335032)

[5.1 Optimization Outcomes 7](#_Toc195335033)

[5.2 Performance Comparison 8](#_Toc195335034)

[5.3 Hyperparameter Selection 12](#_Toc195335035)

[6 Discussion 13](#_Toc195335036)

[7 Conclusion 14](#_Toc195335037)

[8 References 15](#_Toc195335038)

# Introduction

Neural network (NN) performance is highly sensitive to hyperparameter choices (e.g., learning rate, network structure). Traditional tuning methods like Grid Search can be computationally expensive and may not effectively explore the search space. This project investigates Particle Swarm Optimization (PSO), a bio-inspired metaheuristic, as an alternative for optimizing NN hyperparameters.

The chosen application is wine quality prediction, using a public dataset (Luo et al. 2023). The goal is to classify wines into Low, Medium, or High quality based on physicochemical features.

**Objectives:**

1. Develop a baseline MLP neural network for wine quality classification.
2. Optimize the MLP’s hyperparameters using both traditional Grid Search and PSO.
3. Compare the performance (primarily test accuracy) and characteristics of the models optimized by each method.
4. Analyse the relative effectiveness and efficiency of PSO versus Grid Search for this task.

The methodology involves preprocessing the data, defining the NN and search spaces, running both optimization algorithms, training final models with the best-found parameters, and evaluating their performance.

# Execution Instructions

The project was implemented in Python (wine\_4.py) using TensorFlow/Keras, Scikit-learn, NumPy, Pandas, Matplotlib, and Seaborn. The code is structured logically into sections for data handling, model building, optimization methods (Grid Search, PSO), comparison, and main execution flow.

This instruction for Windows PowerShell. Other operation systems may need to adjust these commands.

1. **Environment**: Any python IDE is okay, however Spyder is recommended.
2. **Prerequisites**:

install tensorflow scikit-learn numpy pandas matplotlib seaborn

1. **Dataset**: Download the zip file which uploaded to the Moodle. It contains winequality-red.csv and winequality-white.csv files with Assessment1\_PSO\_NN in same folder.
2. **Run**: Execute python Assessment1\_PSO\_NN, be sure 2 csv file is in same directory.
3. A screenshot of a computer

   AI-generated content may be incorrect.**Output**: Console logs progress; plots appear sequentially (close each to continue); final results printed to console.

Figure 1: Console logs

1. **GitHub**: https://github.com/furkantekkartal/COIT29224/

# Dataset and Preprocessing

The study uses the combined red and white “Vinho Verde” wine quality datasets (Luo et al. 2023). Features include 11 physicochemical measurements (e.g., **fixed acidity**, **alcohol**) plus an added **wine\_type** indicator.

The original quality score (0-10) was mapped to three classes:

* Low (0-5),
* Medium (6),
* High (7-10).

Initial analysis showed a class imbalance, with the Medium class being most frequent.

A chart of different colored squares

AI-generated content may be incorrect.

Figure 2: Wine Quality Class Distribution

**Preprocessing Steps:**

1. Load and combine datasets.
2. Bin quality scores into 3 classes.
3. Split data into 80% training / 20% testing sets (stratified).
4. Standardize features using StandardScaler.
5. A blue background with white text

   AI-generated content may be incorrect.One-hot encode the target variable (quality\_class).

Figure 3: Load and handle the data

# Neural Network and Optimization Methods

## Neural Network Design

A standard MLP was used, built with Keras:

* **Architecture:** Input layer -> (1 to 3) Hidden layers (ReLU activation, Dropout) -> Output layer (3 neurons, Softmax activation).
* **Optimizer:** Adam.
* **Loss:** categorical\_crossentropy.
* **Metrics:** accuracy.
* **Regularization:** Dropout and Early Stopping (monitoring val\_loss).
* **Optimized Hyperparameters:** hidden\_layers, neurons per layer, learning\_rate, dropout\_rate.

## Grid Search Approach

A screenshot of a computer code

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.Grid Search exhaustively evaluated all combinations from a predefined discrete grid:

Figure 4: Grid Search Parameters

Figure 5: Grid Search

Each combination was used to train and evaluate the MLP. The best combination based on test accuracy was selected.

## Particle Swarm Optimization (PSO) Approach

PSO uses a swarm of particles, each representing a hyperparameter set, moving through the search space based on personal (**pBest**) and global (**gBest**) best positions found so far. Key aspects:

* **Search Space:** Continuous bounds were defined:

A screenshot of a computer code

AI-generated content may be incorrect.

Figure 5: PSO Search Space

Integer parameters (layers, neurons) were rounded during evaluation.

* A screen shot of a computer

  AI-generated content may be incorrect.**PSO Parameters:**  
  Linearly decaying inertia weight was used.

Figure 6: PSO Parameters

* **Fitness Function:** Average validation loss from 3-Fold Cross-Validation on the training set (lower is better). Early stopping used within folds.
* **Boundary Handling:** Clamping positions and dampening velocity.
* **Final Model:** Best parameters found (**gBest**) used to train a final model on the full training set.

# Results and Comparison

## Optimization Outcomes

* A screenshot of a computer screen

  AI-generated content may be incorrect.**Grid Search:** Evaluated all combinations in its predefined grid.

Figure 7: Grid Search Results

Test accuracy is the proportion of correct predictions on unseen test data. Higher is better. Best test accuracy: **0.5985.**

* A screenshot of a computer program

  AI-generated content may be incorrect.**PSO:** Ran for 20 iterations with 20 particles.

Figure 8: PSO Iteration & Particles

The output above shows the last fitness (validation losses) found by PSO during optimization (lower is better). Final PSO model test accuracy (higher is better): **0.6238.**

## Performance Comparison

A screenshot of a computer

AI-generated content may be incorrect.A graph with red and blue squares

AI-generated content may be incorrect.

Figure 10: Model Comparison

Figure 9: Best Test Accuracy Comparison

Figure 9 visually compares the overall test accuracy (Y-axis: Proportion, higher is better) of the best models. The best model found by PSO achieved a higher test accuracy (0.6238) compared to the best model from Grid Search (0.5977), representing an improvement of approximately 4.38%.

**A screenshot of a graph

AI-generated content may be incorrect.Per-class metrics reveal more detail**:

Figure 11: Per-Class Metrics Comparison

Figure 10 compares **Precision** (correct positive predictions relative to total positive predictions), **Recall** (correct positive predictions relative to actual positives), and **F1-Score** (harmonic mean of Precision and Recall) for each quality class (X-axis). Higher values (Y-axis: Score 0.0-1.0) are better for all metrics. PSO demonstrates notably better Recall and F1-Scores for the minority ‘Low’ and ‘High’ quality classes, indicating it’s better at identifying these wines. Grid Search showed slightly better Recall for the majority ‘Medium’ class.

**A blue squares with white text

AI-generated content may be incorrect.Confusion matrices visualize these error patterns**:

Figure 12: Confusion Matrices (Grid Search vs. PSO)

Figure 12 shows **confusion matrices**, where rows represent the *actual* quality class and columns represent the *predicted* quality class. The numbers in the cells are sample counts. High values on the diagonal indicate correct predictions. The matrices confirm PSO’s stronger performance on the ‘High’ quality class (128 correct vs. 86 for Grid Search) and slightly better performance on the ‘Low’ class (306 vs 288).

**A graph of a graph showing the number of people in the same direction

AI-generated content may be incorrect.Analysis of the optimization process**:

Figure 13: PSO Convergence Curve (gBest Fitness vs. Iteration)

Figure 13 tracks the **best fitness (validation loss)** (Y-axis: Loss value, lower is better) found by the entire PSO swarm as iterations progress (X-axis: Iteration count). The PSO algorithm successfully **converged**, iteratively decreasing the validation loss, finding the best value around iteration 17.

A graph with a line

AI-generated content may be incorrect.

Figure 14: Optimization Progress Comparison (PSO gBest vs. Grid Search Best).

Figure 14 compares the **validation loss** (Y-axis: Loss value, lower is better) achieved by the best PSO particle over iterations (X-axis) against the single best validation loss found by Grid Search (constant red line). It shows that PSO found a significantly better (lower) final validation loss (~0.738) compared to the best achieved by Grid Search (~0.825) within its limited grid. In terms of efficiency, Grid Search evaluated a fixed number of combinations, while PSO performed **[Swarm Size \* Iterations]**evaluations (e.g., 20 \* 20 = 400), potentially exploring the space more effectively with fewer total model trainings than an exhaustive grid might require for similar coverage.

## Hyperparameter Selection

A comparison of a graph

AI-generated content may be incorrect.The optimal **hyperparameters** (tunable settings of the learning algorithm) identified by each method differed significantly:

Figure 14: Comparison of Optimal Hyperparameters Found

Figure 17 compares the values (Y-axis) for the selected hyperparameters (X-axis). Both methods selected 3 hidden layers. However, PSO favored a configuration with significantly more neurons (52 vs. 16), a higher learning rate (~0.0028 vs. 0.0010), and a lower dropout rate (0.10 vs. 0.20) compared to the best Grid Search model. This suggests PSO identified a more complex model structure with different regularization needs as being optimal.

**A screenshot of a computer

AI-generated content may be incorrect.Top model details:**

Figure 14: Comparison of Optimal Hyperparameters Found

This table shows the parameters and resulting **accuracy** (higher is better) for the top 5 models found by each method.

# Discussion

Based on the experimental results, **PSO yielded significantly better performance** than Grid Search for this wine quality classification task, achieving a higher test accuracy (0.6238 vs 0.5977).

The performance improvement appears linked to PSO’s ability to effectively explore the continuous hyperparameter space. As seen in Figure 3 and Figure 4, PSO was particularly better at identifying the minority classes (‘Low’ and ‘High’ quality wines), suggesting the hyperparameters it found generalized better across the imbalanced dataset. Grid Search, while simpler, was limited by its discrete grid and settled on a less effective configuration.

The convergence plots (Figure 5 and Figure 6) show PSO successfully minimized the validation loss, reaching a better minimum than Grid Search. Figure 7 clearly indicates that PSO identified a different optimal network configuration (more neurons, higher learning rate, lower dropout) compared to Grid Search.

The experiment highlighted the trade-offs: Grid Search is straightforward but computationally expensive for larger spaces and constrained by its grid definition. PSO demonstrated superior effectiveness in finding better hyperparameters within the defined search space, likely achieving this with comparable or fewer total model evaluations than a similarly fine-grained Grid Search would require. However, PSO introduces its own meta-parameters and a degree of randomness.

# Conclusion

This study successfully implemented and compared Grid Search and PSO for optimizing MLP hyperparameters for wine quality classification. PSO demonstrably outperformed Grid Search, achieving higher test accuracy (0.6238 vs. 0.5977) by identifying a superior set of hyperparameters. The objectives were met, providing valuable insights into the effectiveness of using PSO for neural network tuning, especially in navigating complex search spaces and handling class imbalance implicitly through finding better generalizing parameters.

# References

Luo, B, Cheng, L, Wu, Z-G, Li, H & Li, C 2023, Neural Information Processing, Springer Nature.