Sri Sivasubramaniya Nadar College of Engineering, Chennai

(An Autonomous Institution affiliated to Anna University)

Degree & Branch	B.E Computer Science & Engineering	**Semester**	VI
Subject Code & Name	UCS2612 – Machine Learning Laboratory		
Academic Year	2025–2026 (Even)	**Batch**	2023–2027

Experiment 5

Perceptron vs Multilayer Perceptron with Hyperparameter Tuning

Objective

To implement and compare the performance of:

- **Model A:** Single-Layer Perceptron Learning Algorithm (PLA)
- **Model B:** Multilayer Perceptron (MLP) with hidden layers and nonlinear activation functions

Students must select, tune, and justify hyperparameters such as learning rate, activation function, optimizer, batch size, and number of hidden layers.

Dataset

English Handwritten Characters Dataset

Download Link: https://www.kaggle.com/datasets/dhruvildave/english-handwritten-characters-dataset

- Total samples: 3,410
- Number of classes: 62 (0–9, A–Z, a–z)
- Image type: Grayscale

Theory

Perceptron Learning Algorithm (PLA)

The Perceptron is a linear classifier that uses a step activation function to separate data into two classes.

$$w_{t+1} = w_t + \eta(y - \hat{y})x$$

- Uses a step activation function
- Converges only for linearly separable data
- Extended to multi-class problems using One-vs-Rest strategy

Limitation: PLA cannot learn nonlinear decision boundaries required for handwritten character recognition.

Multilayer Perceptron (MLP)

An MLP consists of one or more hidden layers with nonlinear activation functions.

- Activation functions: ReLU, Sigmoid, Tanh
- Loss function: Categorical Cross-Entropy
- Optimizers: SGD, Adam
- Learns nonlinear decision boundaries using backpropagation

Steps for Implementation

1. Preprocess the dataset (resize, flatten, normalize).
2. Implement **PLA** from scratch:
    - Use step activation function.
    - Apply perceptron weight update rule.
    - Extend PLA to multi-class classification using One-vs-Rest.
3. Implement **MLP**:
    - Select number of hidden layers and neurons.
    - Choose activation functions.
    - Train using backpropagation.
4. Perform **hyperparameter tuning**:
    - Learning rate
    - Batch size
    - Optimizer
    - Activation function
5. Compare PLA and MLP using evaluation metrics.

Performance Evaluation

Table 1: Overall Performance Comparison between PLA and MLP

Model	**Accuracy (%)**	**Precision**	**Recall**	**F1-score**
PLA (OvR)				
MLP (Tuned)				

Table 2: Hyperparameter Tuning Results for MLP

Hidden Layers	**Activation**	**Optimizer**	**Learning Rate**	**Batch Size**	**Accuracy (%)**
1 (128)	ReLU	SGD	0.01	32	
2 (256–128)	ReLU	Adam	0.001	64	
3 (512–256–128)	Tanh	Adam	0.0005	64	

Table 3: Training Convergence Comparison

Model	**Epochs**	**Final Training Loss**	**Convergence Behavior**
PLA			Non-convergent
MLP (Tuned)			Stable convergence

Evaluation Metrics

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix
- ROC Curves (micro/macro average)
- Training loss vs epochs curve

A/B Experiment Comparison

Students must clearly highlight:

- Final tuned hyperparameters for MLP
- Strengths and weaknesses of PLA vs MLP
- Effect of hyperparameter tuning on convergence and accuracy

Observation Questions

- Why does PLA underperform compared to MLP?
- Which hyperparameter had the most impact on MLP performance?
- How does optimizer choice affect convergence?
- Does increasing hidden layers always improve performance?
- How can overfitting be detected and mitigated?

Report Checklist

- Aim and Objective
- Dataset Description and Preprocessing
- PLA Implementation and Results
- MLP Implementation and Results
- Hyperparameter Tuning Analysis
- Performance Comparison Tables
- Observations and Conclusion