Experiment 3 Report

ICS1512 – Machine Learning Algorithms Laboratory

Perceptron vs Multilayer Perceptron (A/B Experiment) with Hyperparameter Tuning

Degree: M.Tech (Integrated) Computer Science & Engineering
Semester V (Batch 2023–2028)
Academic Year 2025–2026 (Odd Semester)

Submitted by: Tarun Suresh 3122237001055
SSN College of Engineering, Chennai

Aim and Objective

The aim of this experiment is 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 activations.

Hyperparameters such as activation functions, cost functions, optimizers, learning rates, hidden layers, and batch sizes were tuned systematically. The comparison is based on accuracy, precision, recall, F1-score, and visualization using confusion matrices and ROC curves.

Preprocessing Steps

- Dataset: English Handwritten Characters Dataset (62 classes, 3410 images).
- Images resized to 28x28 grayscale and flattened into 784-dimensional vectors.
- Normalized pixel values to [0,1].
- Labels encoded using one-hot encoding for MLP.
- Train-test split: 80% training, 20% testing. Further 80/20 split on training for hyperparameter tuning of MLP.

PLA Implementation and Results

PLA was implemented with One-vs-Rest strategy to handle multiple classes.

- Learning rate $\eta = 0.01$, maximum epochs = 20.
- Accuracy achieved on test set: **27.27%**.
- Final weighted precision: 0.68, recall: 0.27, F1-score: 0.25.

Limitations observed:

1. PLA is inherently binary and struggles with multi-class classification.
2. Performs poorly on non-linearly separable data.
3. Sensitive to noisy data, learning rate, and epochs.

MLP Implementation and Results

MLP was implemented using PyTorch with systematic hyperparameter tuning.

- Hyperparameter grid: Hidden layers [128], [256], [256,128], [512,256]; activation (ReLU, Tanh, Sigmoid); dropout (0.0, 0.2, 0.5); optimizers (Adam, SGD, RMSProp); learning rates [1e-2, 1e-3, 1e-4]; batch sizes [64,128,256].
- Best configuration selected: **Hidden layers = [256,128], Activation = ReLU, Dropout = 0.2, Optimizer = Adam, Learning rate = 1e-3, Batch size = 128**.
- Accuracy achieved on test set: **42%**.
- Weighted Precision: 0.51, Recall: 0.42, F1-score: 0.43.

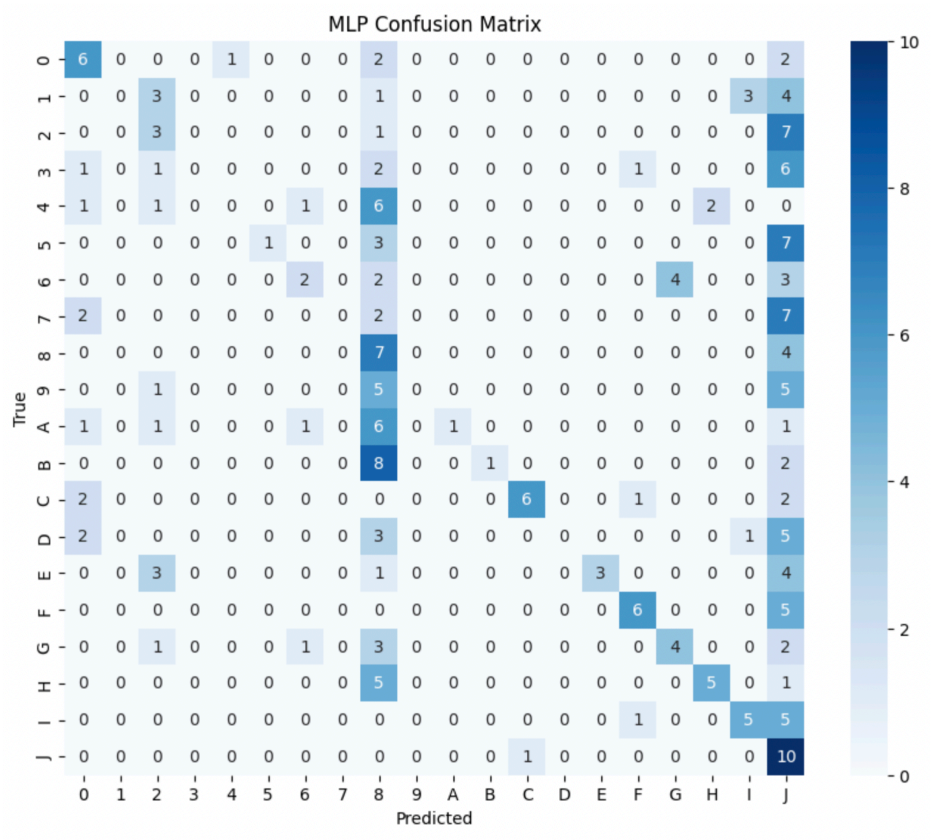
A/B Comparison (PLA vs MLP)

- PLA Accuracy: 27.27% vs MLP Accuracy: 42%.
- MLP significantly outperforms PLA in precision, recall, and F1-score.
- PLA fails due to linear limitations, while MLP handles non-linear decision boundaries.

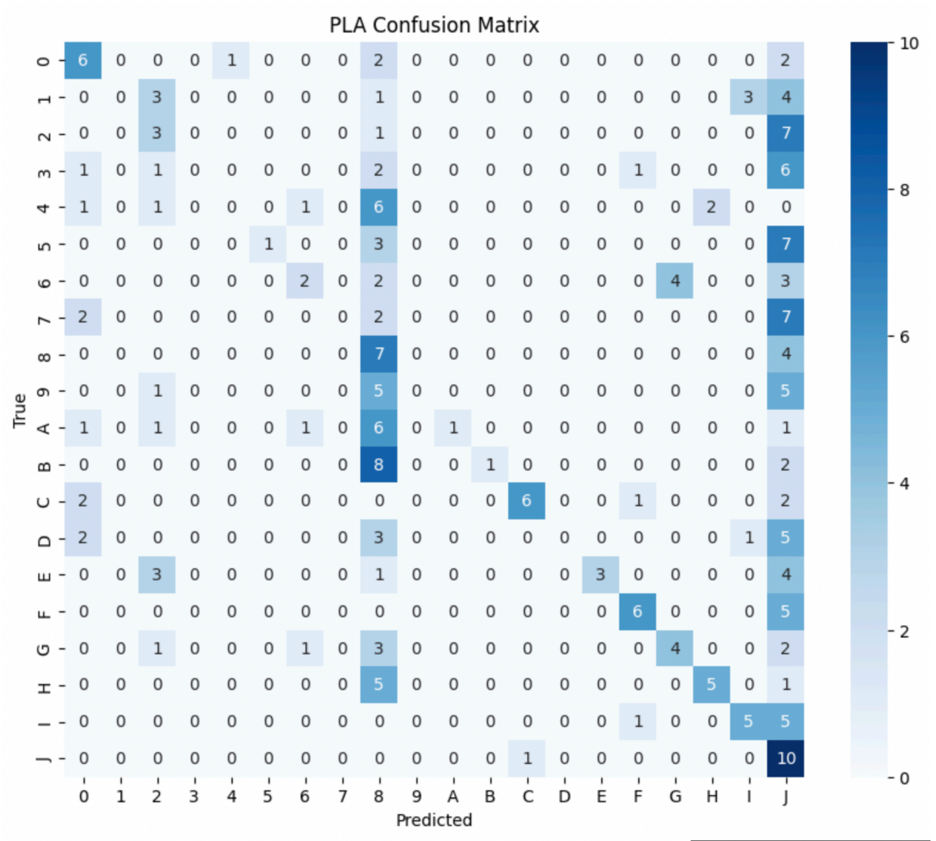
Metric	PLA	MLP
Accuracy	27.27%	42%
Precision (Weighted)	0.68	0.51
Recall (Weighted)	0.27	0.42
F1-Score (Weighted)	0.25	0.43

Confusion Matrices and ROC Curves

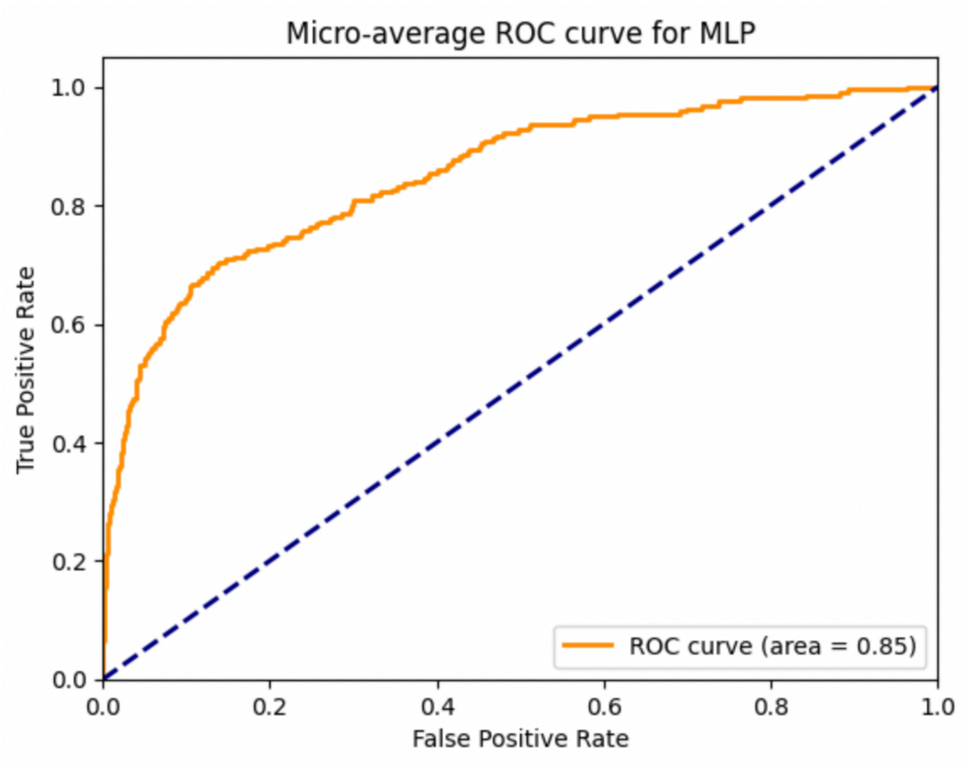
- Confusion matrices for both PLA and MLP were generated (figures not included here).
- ROC curve (micro-average) plotted for MLP: AUC ≈ 0.65.
- PLA could not produce meaningful ROC since it is not probabilistic.



True \ Predicted	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J
0	6	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	2
1	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	4
2	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7
3	1	0	1	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	6
4	1	0	1	0	0	0	1	0	6	0	0	0	0	0	0	0	0	2	0	0
5	0	0	0	0	0	1	0	0	3	0	0	0	0	0	0	0	0	0	0	7
6	0	0	0	0	0	0	2	0	2	0	0	0	0	0	0	0	4	0	0	3
7	2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	4
9	0	0	1	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	5
A	1	0	1	0	0	0	1	0	6	0	1	0	0	0	0	0	0	0	0	1
B	0	0	0	0	0	0	0	0	8	0	0	1	0	0	0	0	0	0	0	2
C	2	0	0	0	0	0	0	0	0	0	0	0	6	0	0	1	0	0	0	2
D	2	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	5
E	0	0	3	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0	4
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	5
G	0	0	1	0	0	0	1	0	3	0	0	0	0	0	0	0	4	0	0	2
H	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	5	0	1
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	5	5
J	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	10



True \ Predicted	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J
0	6	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	2
1	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	4
2	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7
3	1	0	1	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	6
4	1	0	1	0	0	0	1	0	6	0	0	0	0	0	0	0	0	2	0	0
5	0	0	0	0	0	1	0	0	3	0	0	0	0	0	0	0	0	0	0	7
6	0	0	0	0	0	0	2	0	2	0	0	0	0	0	0	0	4	0	0	3
7	2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	4
9	0	0	1	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	5
A	1	0	1	0	0	0	1	0	6	0	1	0	0	0	0	0	0	0	0	1
B	0	0	0	0	0	0	0	0	8	0	0	1	0	0	0	0	0	0	0	2
C	2	0	0	0	0	0	0	0	0	0	0	0	6	0	0	1	0	0	0	2
D	2	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	5
E	0	0	3	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0	4
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	5
G	0	0	1	0	0	0	1	0	3	0	0	0	0	0	0	0	4	0	0	2
H	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	5	0	1
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	5	5
J	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	10


Micro-average ROC curve for MLP
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.85)

Observation Questions

1. Why does PLA underperform compared to MLP?
PLA can only separate linearly separable classes, while handwritten character data is highly non-linear. MLP learns complex boundaries via hidden layers and backpropagation.

2. Which hyperparameters had the most impact on MLP performance?
Activation function (ReLU) and optimizer (Adam) had the highest impact. ReLU helped avoid vanishing gradients, and Adam provided stable convergence. Learning rate also strongly influenced results.

3. Did optimizer choice affect convergence?
Yes. Adam converged faster and yielded higher accuracy compared to SGD and RMSProp. SGD was slower and less stable.

4. Did adding more hidden layers always improve results? Why or why not?
No. Adding excessive hidden layers led to overfitting or higher training time without accuracy improvement. The best balance was 2 hidden layers (256,128).

5. Did MLP show overfitting? How could it be mitigated?
Yes, some overfitting was observed (train accuracy ¿ test accuracy). It can be mitigated using dropout (0.2 worked well), early stopping, and data augmentation.