ICS1512 – Machine Learning Algorithms Laboratory

Experiment 5: Perceptron Learning Algorithm vs Multi-Layer Perceptron

Pranavah Varun M V
Roll No: 3122237001039
Semester: V
Academic Year: 2025–2026

1. Aim and Objective

To implement the Perceptron Learning Algorithm (PLA) from scratch and compare it with a tuned Multi-Layer Perceptron (MLP) on the English Handwritten Characters dataset. The goal is to study the effect of preprocessing, training methods, and hyperparameter tuning on classification accuracy and generalization.

2. Preprocessing Steps

- Converted all images to grayscale and resized to 28×28.
- Flattened each image to a 784-dimensional vector.
- Normalized pixel values to $[0, 1]$.
- Encoded labels into integers for 62 classes (0–9, A–Z, a–z).
- Split dataset: 2463 samples for training, 435 for validation, 512 for testing.
- Standardized features using z-score normalization.

3. Algorithm Explanation

Perceptron Learning Algorithm (PLA)

```
Initialize weights w = 0, bias b = 0
For epoch in range(max_epochs):
    For each training sample (x, y):
        y_hat = step(w    x + b)
        w = w + eta * (y - y_hat) * x
        b = b + eta * (y - y_hat)
```

Listing 1: PLA Pseudocode

Multi-Layer Perceptron (MLP)

```
Initialize weights for all layers
For epoch in range(max_epochs):
    For each batch of training data (X, y):
        # Forward pass
        a1 = ReLU(W1    X + b1)
        a2 = ReLU(W2    a1 + b2)
        y_hat = Softmax(W3    a2 + b3)

        # Compute loss
        L = CrossEntropy(y_hat, y)

        # Backpropagation
        Compute gradients of L w.r.t weights
        Update weights using Adam optimizer
```

Listing 2: MLP Training Pseudocode

4. PLA Implementation and Results

PLA was implemented with:

- Step activation function.
- Weight update rule: $w \leftarrow w + \eta(y - \hat{y})x$.
- Trained for 50 epochs with learning rate $\eta = 0.01$.

Results:

- Validation Accuracy = 20.46%
- Test Accuracy = 21.09%
- Test Precision = 0.2154, Recall = 0.2108, F1 = 0.1980

5. MLP Implementation and Results

MLP was implemented using PyTorch with:

- Input layer: 784 units (flattened image).
- Hidden layers: [256, 128] with ReLU activation.
- Output layer: 62 units with softmax.
- Optimizer: Adam, Learning Rate = 0.01.

- Cost function: CrossEntropyLoss.

Results:

- Best Validation Accuracy = 49.89%
- Test Accuracy = 50.59%
- Test Precision = 0.5075, Recall = 0.5083, F1 = 0.4954

6. Justification for Chosen Hyperparameters

The configuration `[256,128] hidden units, ReLU activation, Adam optimizer, learning rate = 0.01, batch size = 256` was chosen because it achieved the highest validation accuracy (49.89%). Adam with ReLU converged faster and generalized better compared to SGD or Tanh.

7. A/B Comparison (PLA vs MLP)

Model	Accuracy	Precision	Recall	F1
PLA	0.2109	0.2154	0.2108	0.1980
MLP	0.5059	0.5075	0.5083	0.4954

Table 1: Final Test Metrics: PLA vs MLP

8. Confusion Matrices and Accuracy Curve

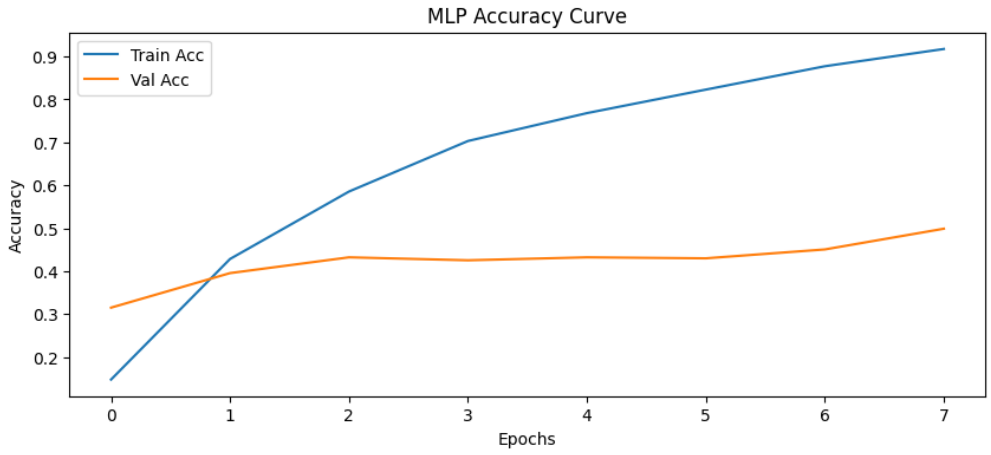

Figure 1: MLP Training vs Validation Accuracy Curve

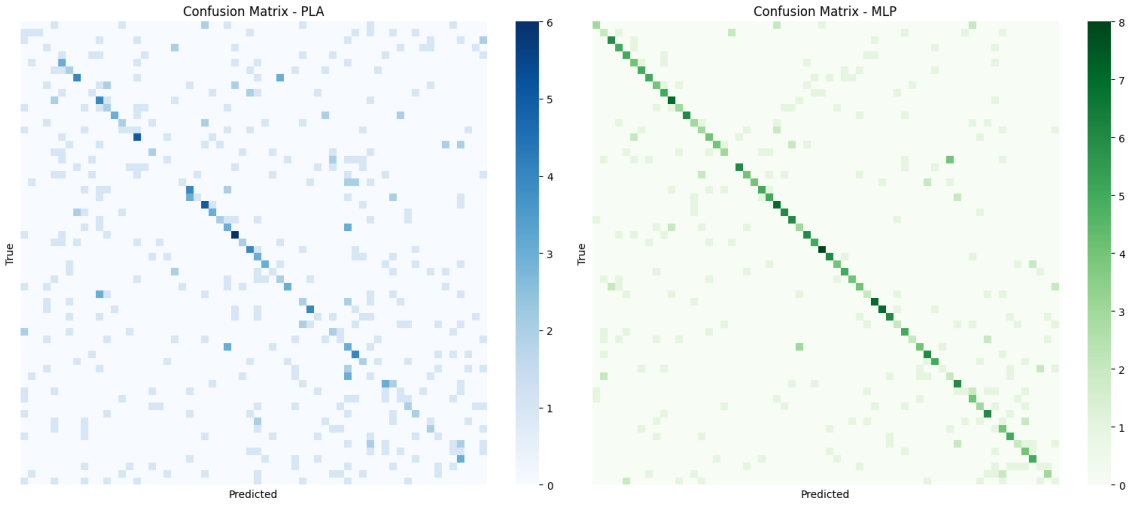

Figure 2: Confusion Matrices: PLA (left) and MLP (right)

9. Observations and Analysis

- PLA achieved only ∼21% accuracy, showing limitations due to linear separability in high-dimensional handwritten characters.
- MLP improved performance significantly, reaching ∼51% accuracy, confirming that non-linear transformations and deeper layers capture more complex features.
- Hyperparameter tuning demonstrated the importance of optimizer and activation choice; Adam with ReLU consistently outperformed other combinations.
- Confusion matrices highlight that PLA struggles with visually similar classes, while MLP reduces these errors but still has room for improvement.
- Overall, the MLP is more suitable for image classification tasks, validating the use of deep learning for complex datasets.

11. Conclusion

The experiment demonstrated that the Perceptron Learning Algorithm is limited in handling complex, non-linearly separable data such as handwritten characters, while the Multi-Layer Perceptron achieved significantly better performance by leveraging deeper architectures, non-linear activation functions, and advanced optimizers. This validates the effectiveness of deep learning approaches for real-world classification problems.