CSE343: Machine Learning

Assignment-4

Himanshu Raj (2022216)

November 27, 2024

Section A

1) Consider the forward pass of a convolutional layer in a neural network architecture, where the input is an image of dimensions M ×N (where min(M, N) ≥1) with P channels (P ≥1) and a single kernel of size K ×K (1 ≤K ≤min(M, N)).

a) stride of 1 and no padding
feature map dimension = floor((input dimension - kernel size + 2 x padding) / stride) + 1
Resulting feature map dimensions = **(M - K + 1) x (N - K + 1)**

b) Number of elementary operations to compute one pixel in the resulting feature map:
Multiplications in one channel = K x K = K^2
Multiplications in P channels = K x K x P = K^2P
Additions in one channel = K x K - 1 = K^2 - 1
Additions in P channels = P (K x K - 1) = K x K x P - P = K^2P - P
Additions for final values of P channels = P - 1
Total operations = $K^2P + K^2P - P + P - 1$ = **$2K^2P - 1$**

c) Total operations for feature map by one kernel = $(2K^2P - 1) \times (M - K + 1) \times (N - K + 1)$
Total operations for Q kernels = $Q \times (2K^2P - 1) \times (M - K + 1) \times (N - K + 1)$
Time complexity for forward pass = **$O(Q K^2 P (M-K) (N-K))$**

assuming min(M, N) ≫ K, time complexity for forward pass = **$O(Q K^2 P M N)$**

2) K-Means algorithm:
Decide a value for K, the number of clusters.
Initialize the K cluster centres (random centroids if initial centroids are not provided).

Assignment step:
Each data point is assigned to the cluster with the nearest centroid based on a distance function, Euclidean distance or Mahalanobis distance. The objective is to minimize the sum of distances of the points to their respective centroid.

Updation step:
After the assignment step, the positions of cluster centroids are updated. The new position of a cluster centroid is estimated by calculating the mean of all data points assigned to that cluster.

Convergence: Assignment and updation steps are repeated until none of the data points change their cluster membership.

Determining the optimal number of clusters:
We can use the Elbow method to determine the optimal number of clusters. This technique involves running the K-Means algorithm with different values of k and plotting the within-cluster sum of squares (WCSS) against k. The "elbow" point, where the rate of decrease sharply changes, suggests a suitable number of clusters, balancing between model complexity and variance.

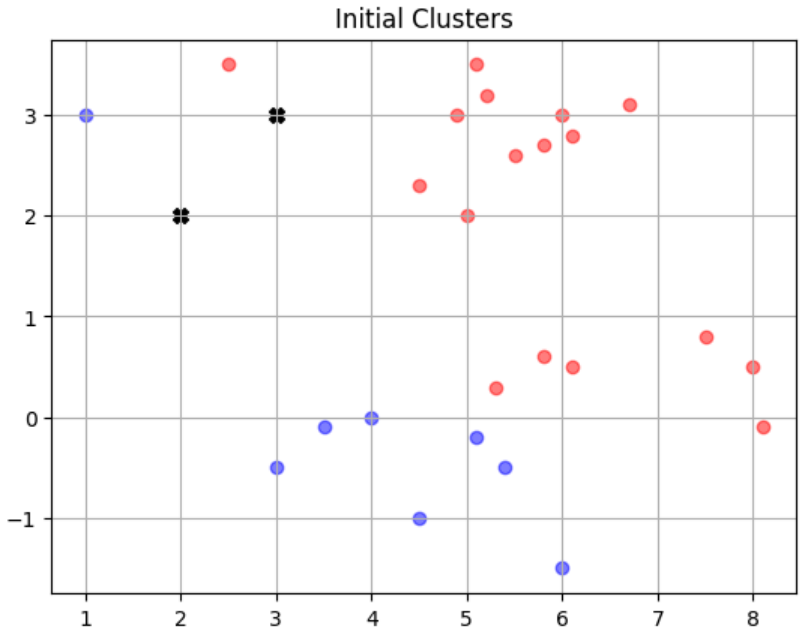
Random Initialization and Global Minima:
Randomly assigning initial cluster centroids does not guarantee that the K-Means algorithm will converge to the global minimum. Instead, it may lead to convergence at a local minimum depending on the initial positions of the centroids.

Section B

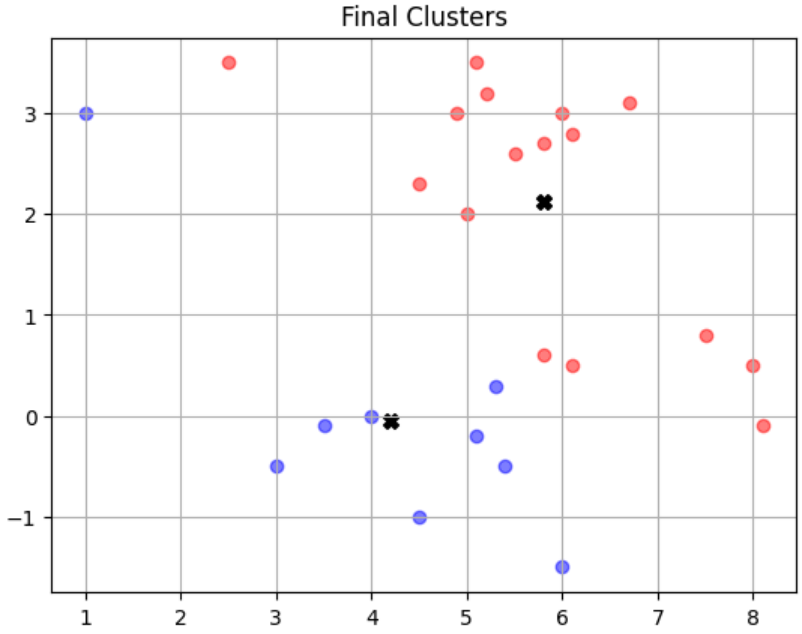
In this section, we will implement the K-Means clustering algorithm from scratch in Python, using the Euclidean distance as the distance function.

1) The implementation initializes centroids, assigns clusters, updates centroids and checks for convergence at the end of each iteration.
We use k=2 and initial centroids as [(3.0, 3.0), (2.0, 2.0)].

2) After the convergence of the K-Means algorithm, our final centroids are [(5.8, 2.125), (4.2, -0.056)] and observed WCSS is 83.67.

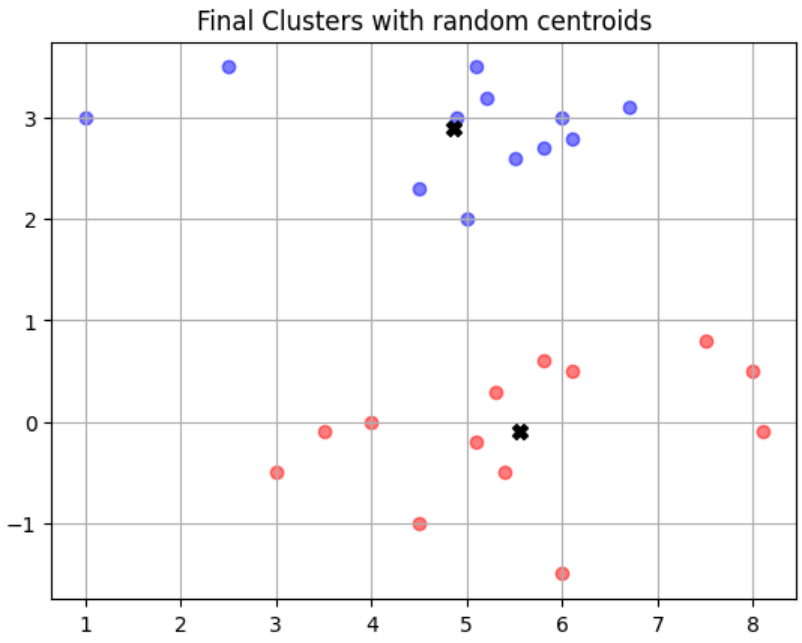


The clusters at the start of the algorithm.

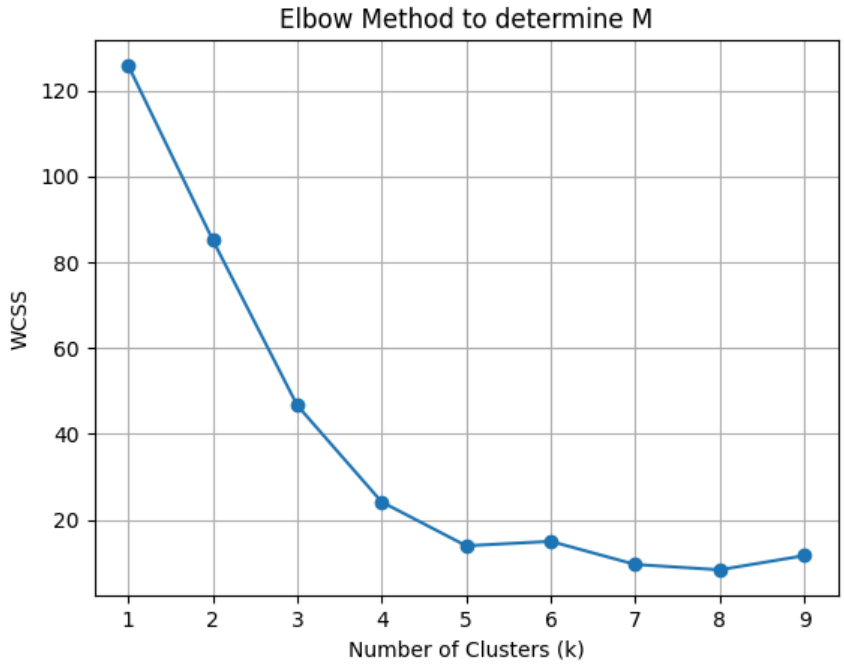


The clusters at the end of the algorithm.

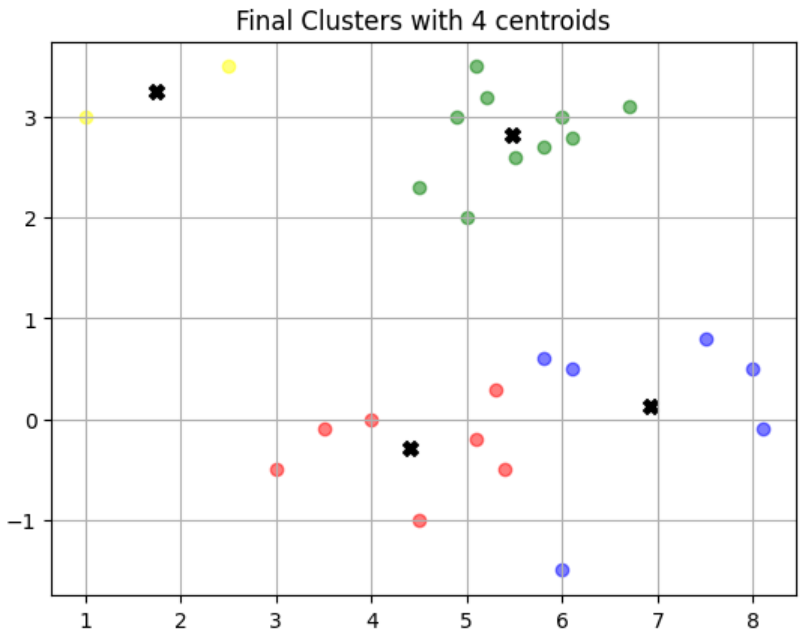
3) We also try random initialization of centroids, and the clusters are shown below. The observed WCSS value with random centroids is 67.16, which is less than what we saw with initialized centroids, indicating better clustering.



4) Elbow method to determine the optimal number of clusters, M



The elbow point is 4, so M = 4. The observed WCSS for 4 centroids is 22.63.



Section C

In this section, we will use the PyTorch package to train CNN and MLP on the CIFAR-10 dataset. The CIFAR-10 dataset consists of 60,000 32x32 RGB images of 10 classes, with 6,000 images per class. We will only work with 3 classes, i.e. [0,1,2].

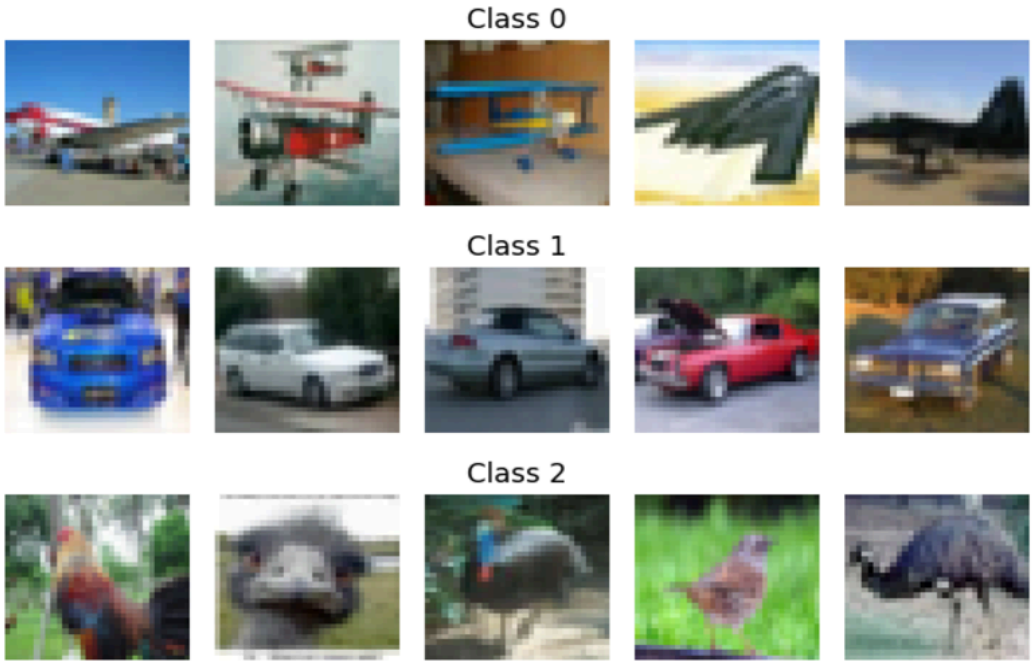
- **Data Preparation**

We load the dataset and filter it to work with 3 classes. We have created a custom Dataset class for the data. We convert the images to tensor objects and normalize them so that values are in the range [-1, 1].

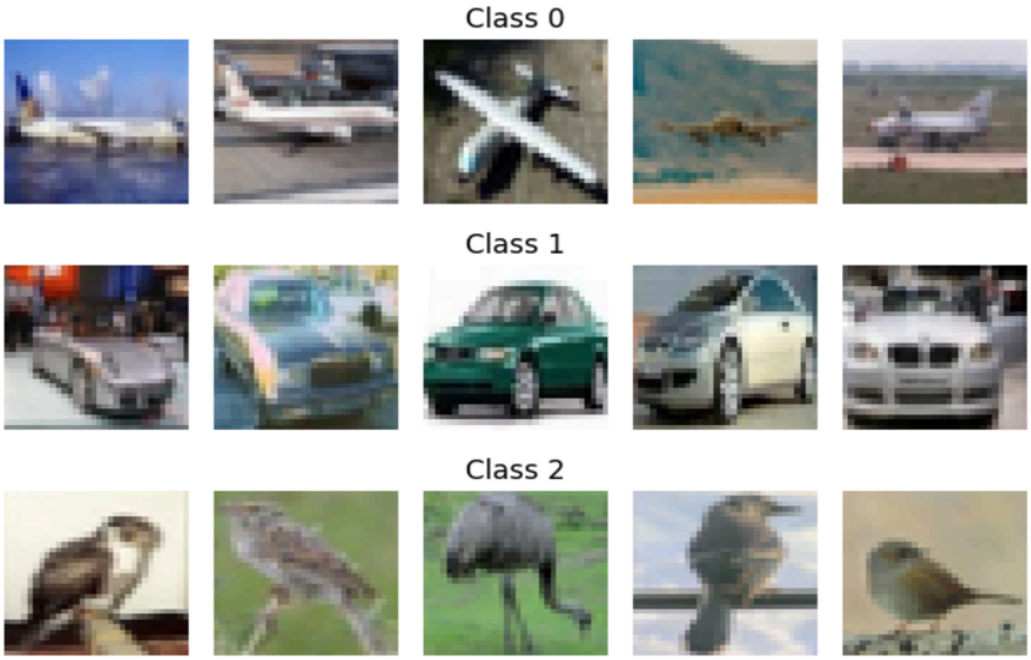
```
Distribution of classes [0, 1, 2]
Training set: [4000 4000 4000]
Validation set: [1000 1000 1000]
Testing set: [1000 1000 1000]
```

- **Visualization**

Training set



Validation set

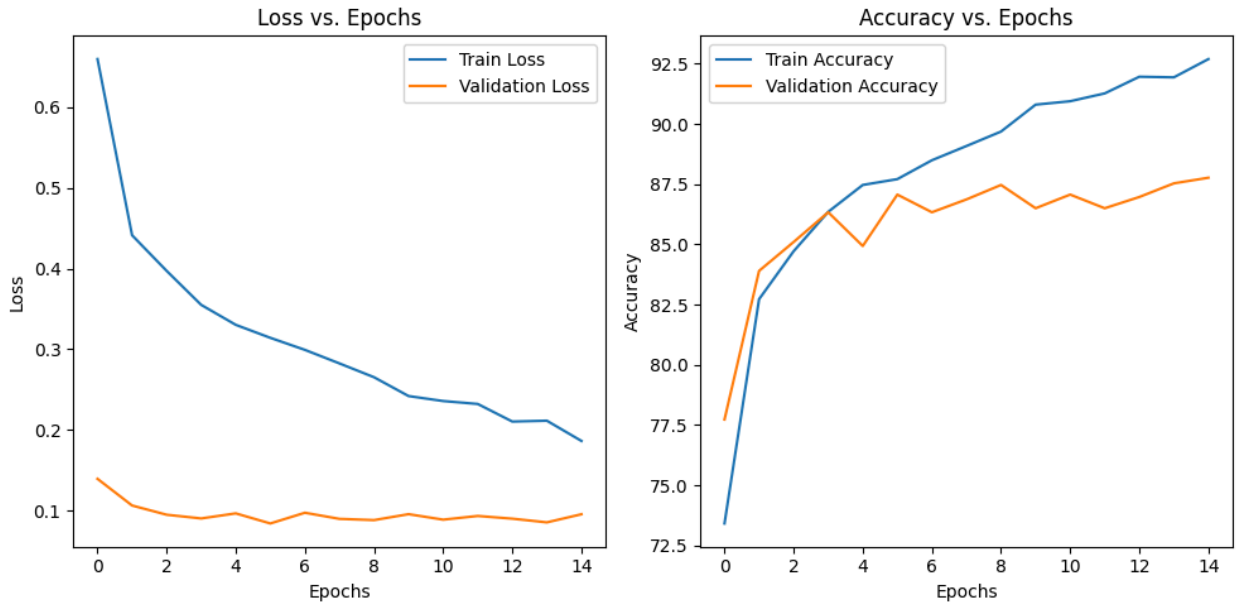


- **CNN**

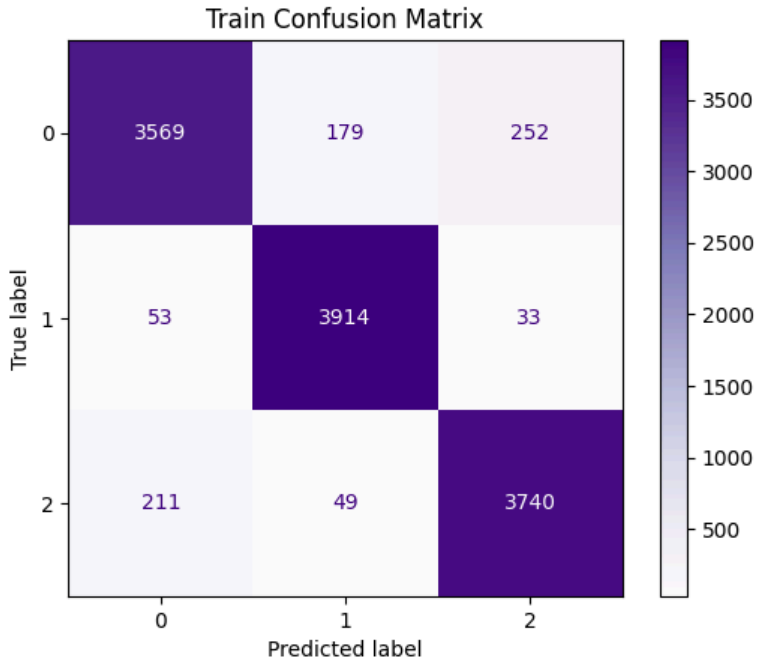
We will train a CNN model with 2 convolutional layers using the cross-entropy loss function with the Adam optimizer with lr = 0.005 for 15 epochs.

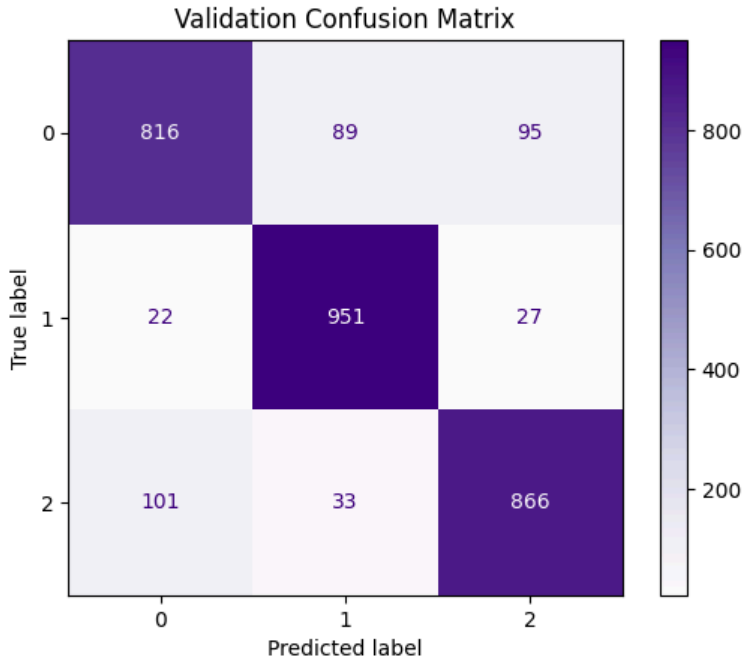
```
----- Epoch 15/15 -----
Train Loss: 0.1866, Train Accuracy: 92.69%
Val Loss: 0.0958, Val Accuracy: 87.77%
```

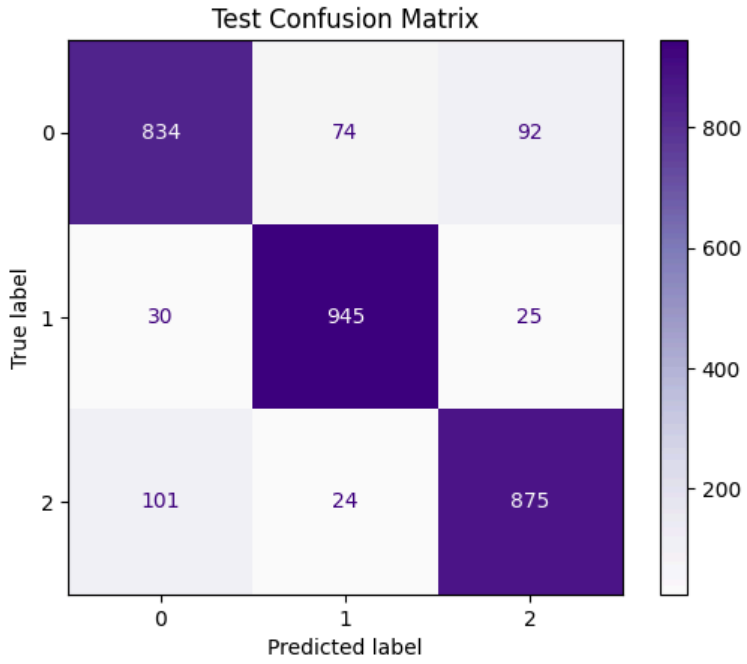
```
Test set Accuracy: 88.47%
Test set F1-Score: 0.8842
```



The accuracy increases over the epochs. The training loss decreases over the epochs, and the split is unfortunate because validation loss is less than training loss over the epochs.


Train Confusion Matrix
3569
179
252
53
3914
33
211
49
3740
True label
Predicted label
0
1
2
3500
3000
2500
2000
1500
1000
500


Validation Confusion Matrix
816
89
95
22
951
27
101
33
866
True label
Predicted label
0
1
2
800
600
400
200



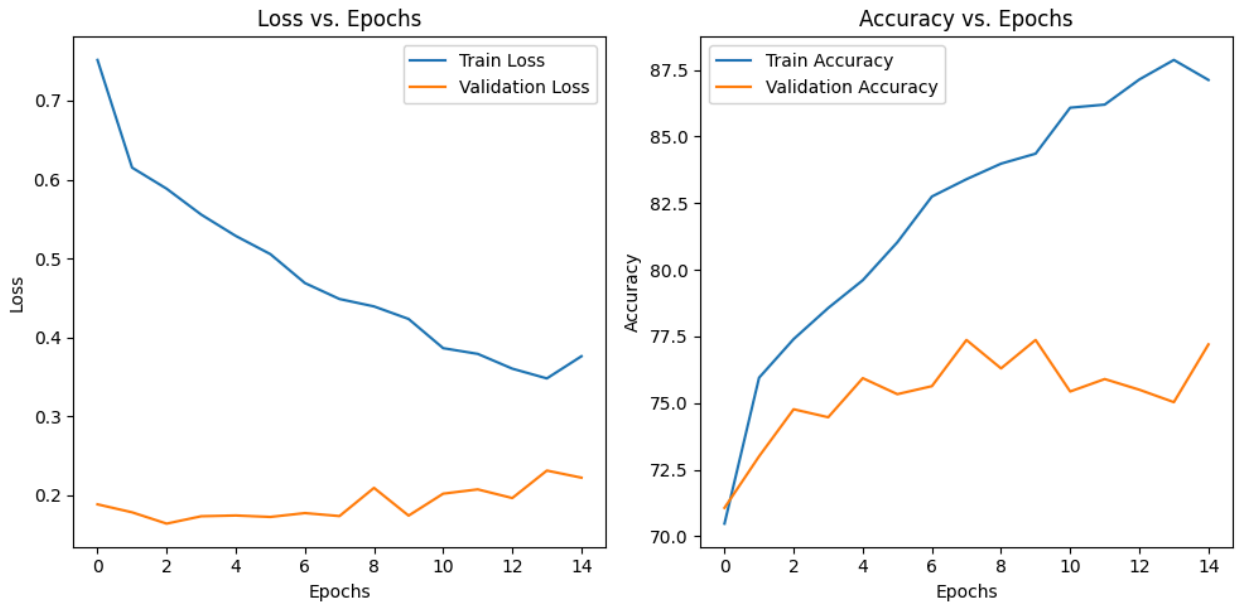
- **MLP**

We will train an MLP model using the cross-entropy loss function with the Adam optimizer with lr = 0.005 for 15 epochs.

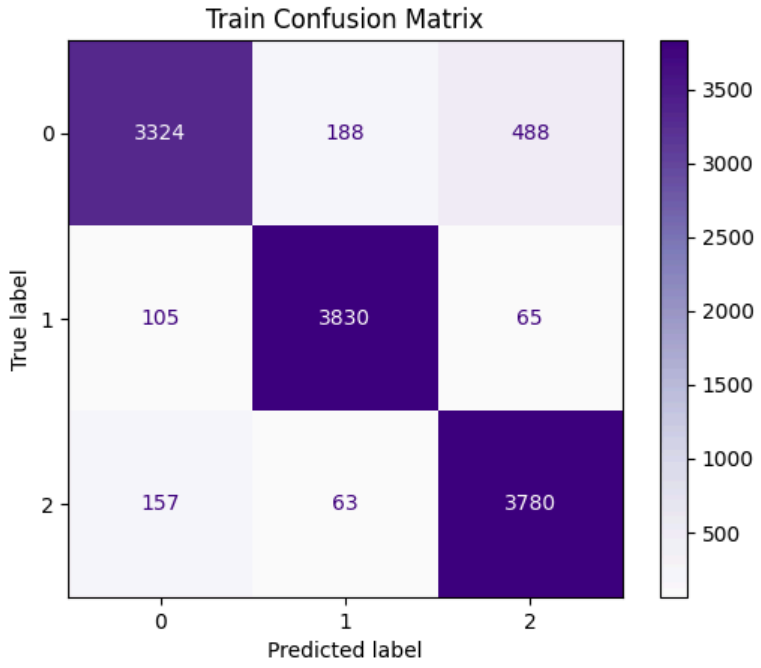
```
----- Epoch 15/15 -----
Train Loss: 0.3761, Train Accuracy: 87.12%
Val Loss: 0.2222, Val Accuracy: 77.20%
```

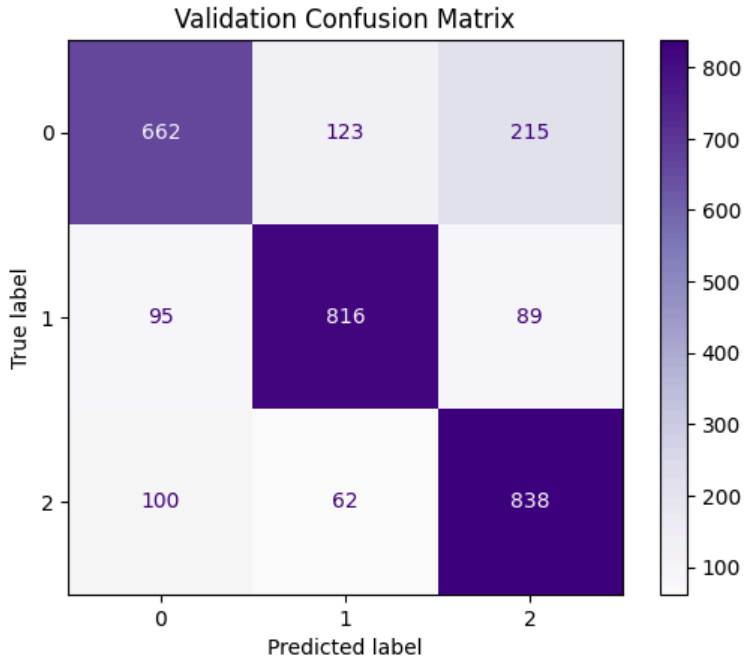
```
Test set Accuracy: 78.97%
Test set F1-Score: 0.7886
```

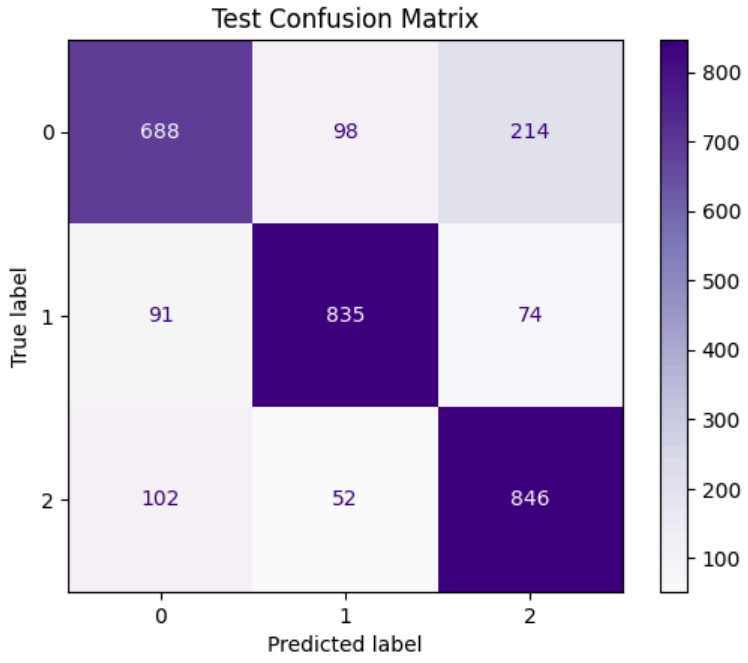
CNN performs better than MLP on this image dataset because CNN has 2 convolutional layers that better capture the correlation between the pixels among images. MLP directly flattens the input image, and the correlation between pixels is lost while training the model.



The accuracy increases over the epochs. The training loss decreases over the epochs, and the split is unfortunate because validation loss is less than training loss over the epochs.




Validation Confusion Matrix
662
123
215
95
816
89
100
62
838
True label
Predicted label
0
1
2
800
700
600
500
400
300
200
100


Test Confusion Matrix
688
98
214
91
835
74
102
52
846
True label
Predicted label
0
1
2
800
700
600
500
400
300
200
100