report_2022266

Section A

Lakshay Kumar
2022266

SECTION - A

PART - A

(a) We took stride = 1 and padding = 0.
$\therefore$ Both height(M) & width(N) will be reduced by,
Output height $= 1+M-K$
Output width $= 1-K+N$
$\Rightarrow$ Result feature map dimension $= (1+M-K) \times (1-K+N)$

(b) For a single pixel in output feature map:

① Kernel of size $K \times K$ spans K^2 positions in the input image.
Each kernel element performs:
- one multiplication with the corresponding input pixel. A total of K^2 multiplications.

② Results of K^2 multiplications are summed up to produce a single output pixel. This requires K^2-1 addition.

③ If input image has P channels, kernel will span all channels. $\therefore$ Total no of oper.
- Multiplication $- P \times K^2$
- Addition $- P \times K^2 - 1$

④ Total Mul $- P \times K^2$
Total Add $- P \times K^2 - 1$
$\therefore$ Total op $= P \times K^2 + P \times K^2 - 1$
$\Rightarrow P \times 2K^2 - 1$

DATE
PAGE

(c) To analyze computional T.C. of forward pass with Q kernels, :-

~~No of~~
From (a), resulting feature map dimensions for a single kernel are: $(M-K+1) \times (N-K+1)$.
Thus, total output pixels in feature map:
No of output pixels $= (M-K+1)(N-K+1)$.

From (b), cost to compute one output pixel for a single kernel is :-
Cost for 1 pixel $\Rightarrow P \times 2K^2 - 1$

For a single kernel, Total cost is the cost per pixel multiplied by no of pixels :-
$\Rightarrow (M-K+1)(N-K+1) \times P \times 2K^2 - 1)$

Cost for Q kernel $= Q \times M-K+1 \times N-K+1 \times P \times 2K^2 - 1$

Considering $M \times N$ are large,
$\Rightarrow T.C = O(Q \times P \times K^2 \times M \times N)$

Now assuming $Min(M, N) >> K$:
$\Rightarrow T.C = O(Q \times P \times M \times N)$

DATE
PAGE

(B) ① Assignment Step:
Using Euclidean dist: $Cluster(x_i) = \arg\min_k ||x_i - \mu_k||^2$

② Update step:
After assigning all points to clusters, the centroids are updated by finding mean of all points assigned to each cluster:

$$\mu_k = \frac{1}{n_k} \sum_{x_i \in C_k} x_i$$

We repeat these steps until centroids stop changing or assignment stabilises.

Elbow Method -

① Run K-Means Algo for various values of k.
② For each k, compute inertia.
③ Plot inertia versus k.
④ Look for elbow point, where the decrease in inertia slows significantly.

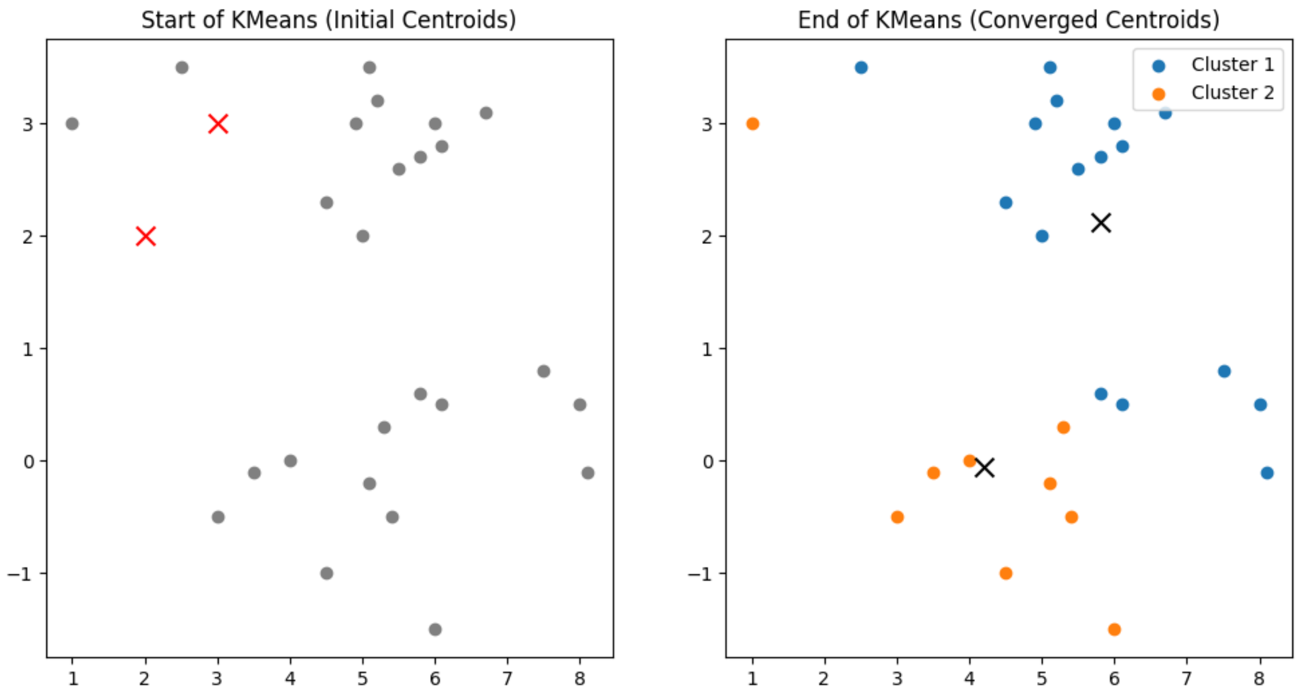
No, it isn't guaranteed that random initialization of centroids can lead to Global Minima.
We can use K-Means ++ for this.

Section B

1. Implement the k-means clustering : Initialization, Assignment, Update, Convergence Check (convergence threshold of 1e-4.)

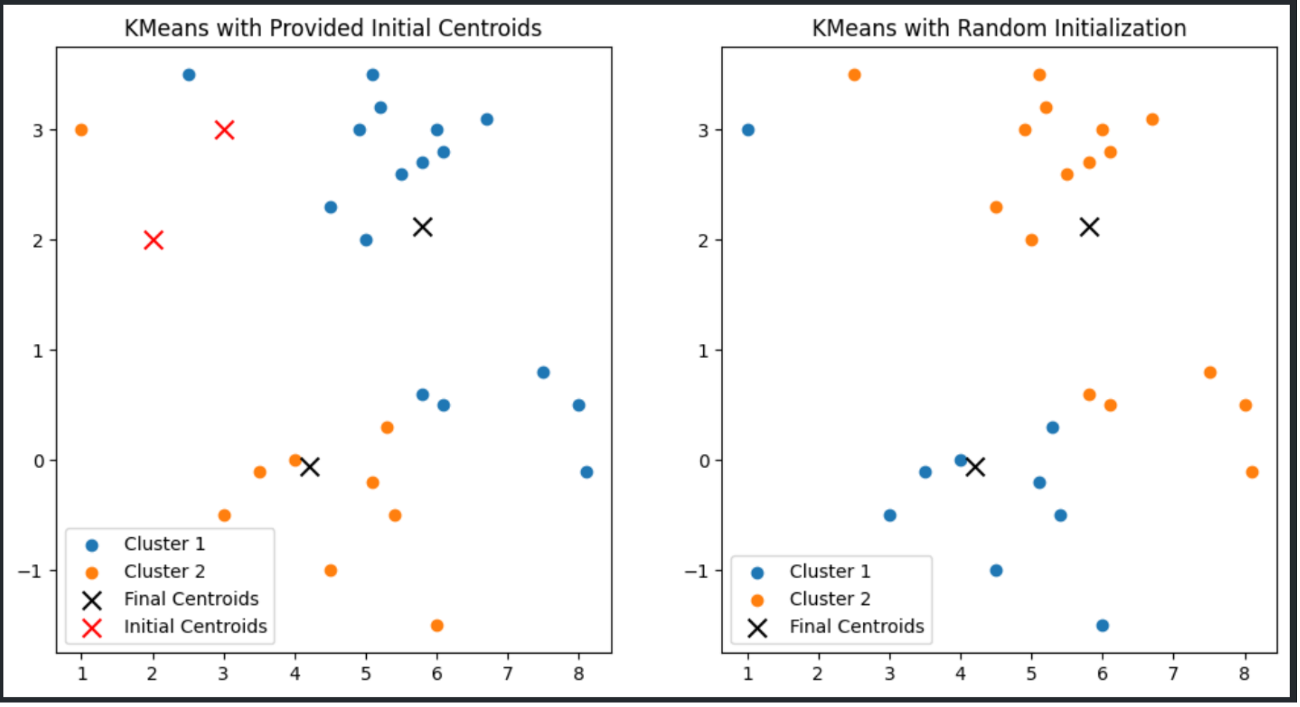
```
Final Centroids after convergence:
[[ 5.8         2.125     ]
 [ 4.2        -0.05555556]]
```

2. Final centroids



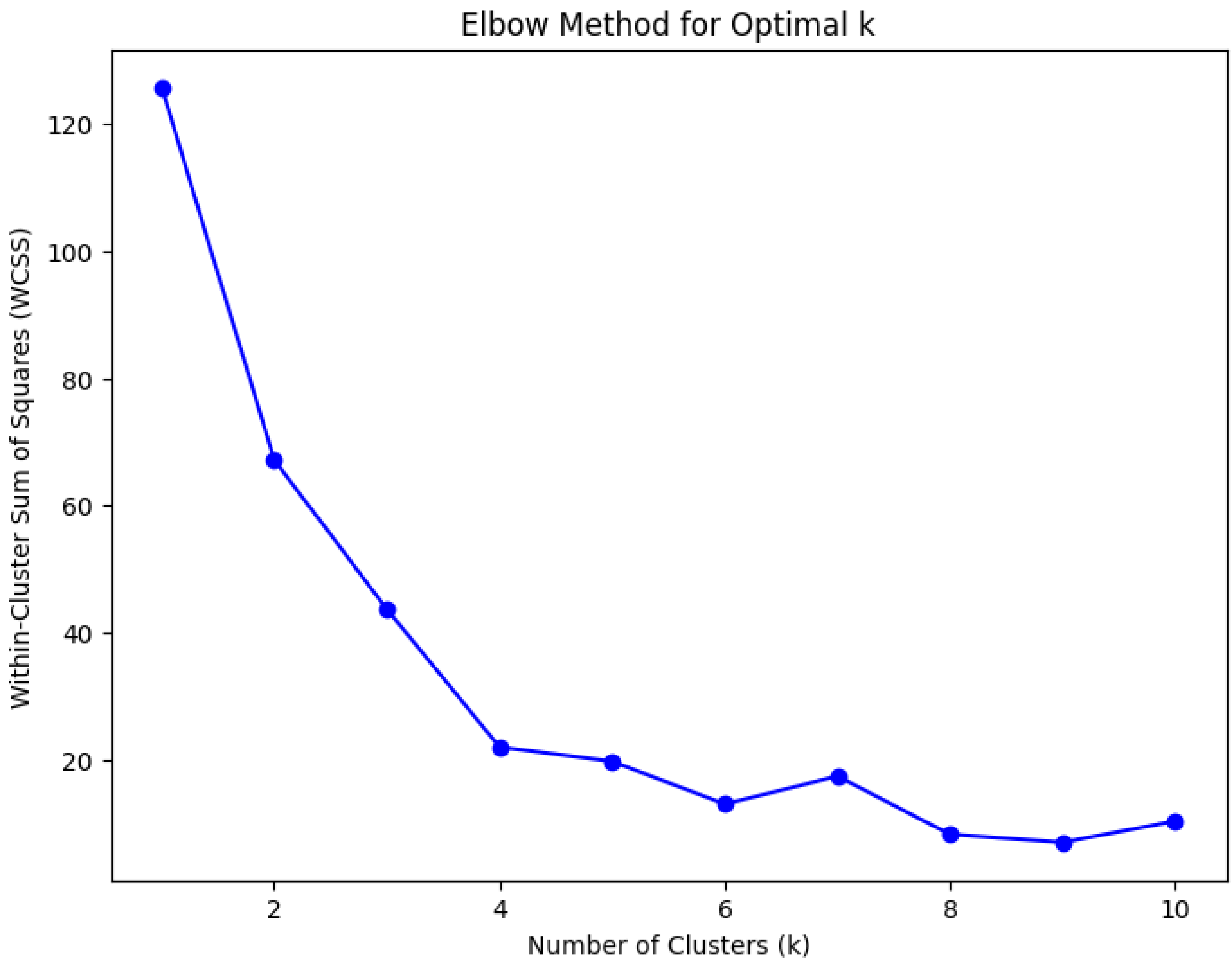
3. Provided initialization vs Random initialization of centroids

```
Convergence reached at iteration 3
Convergence reached at iteration 2
```

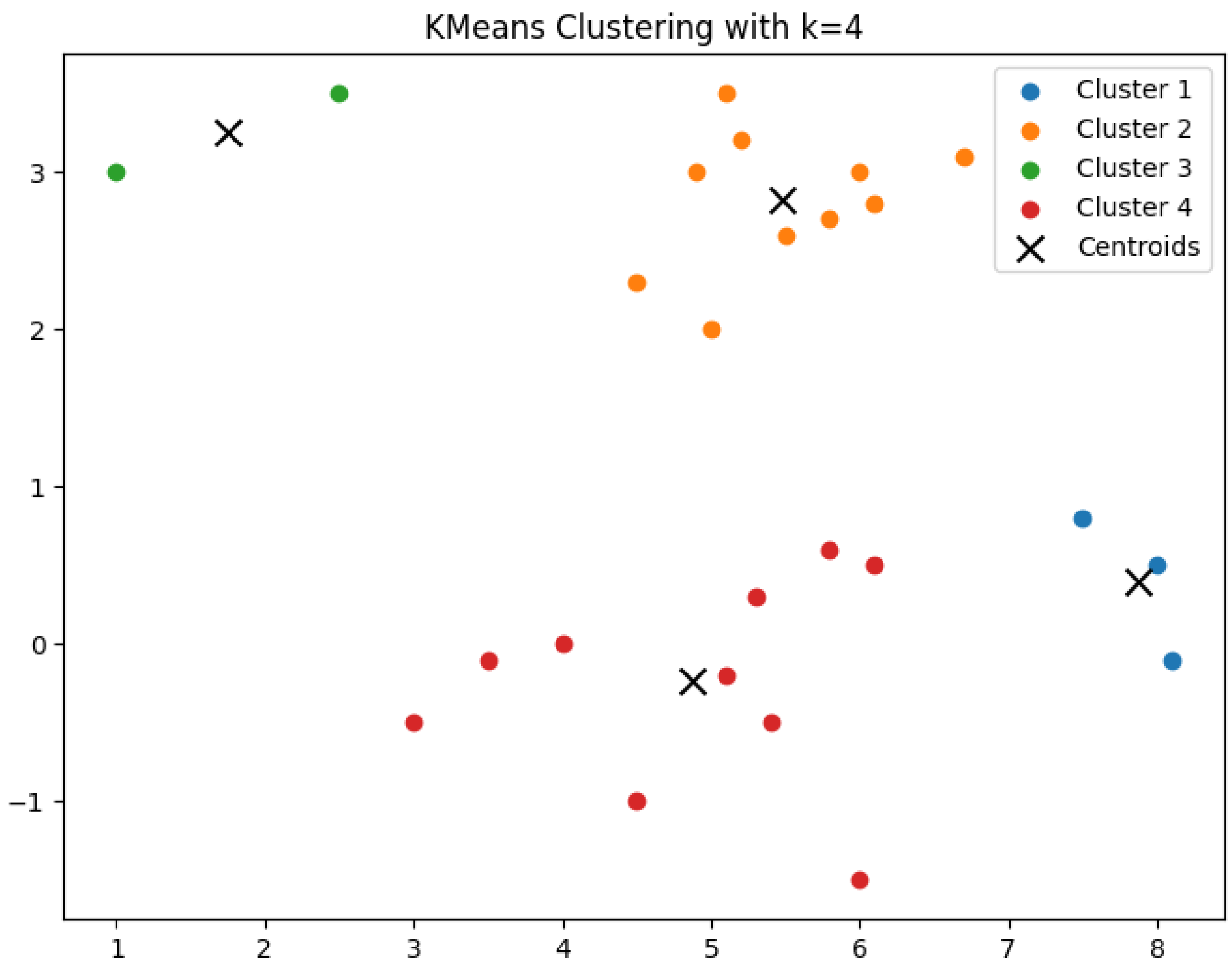



```
Final Centroids (Provided Initialization):
[[ 5.8          2.125     ]
 [ 4.2         -0.05555556]]
Final Centroids (Random Initialization):
[[ 4.2         -0.05555556]
 [ 5.8          2.125     ]]
```

4. The optimal number of clusters using the Elbow method.



Elbow Method for Optimal k
Within-Cluster Sum of Squares (WCSS)
Number of Clusters (k)
120
100
80
60
40
20
2
4
6
8
10



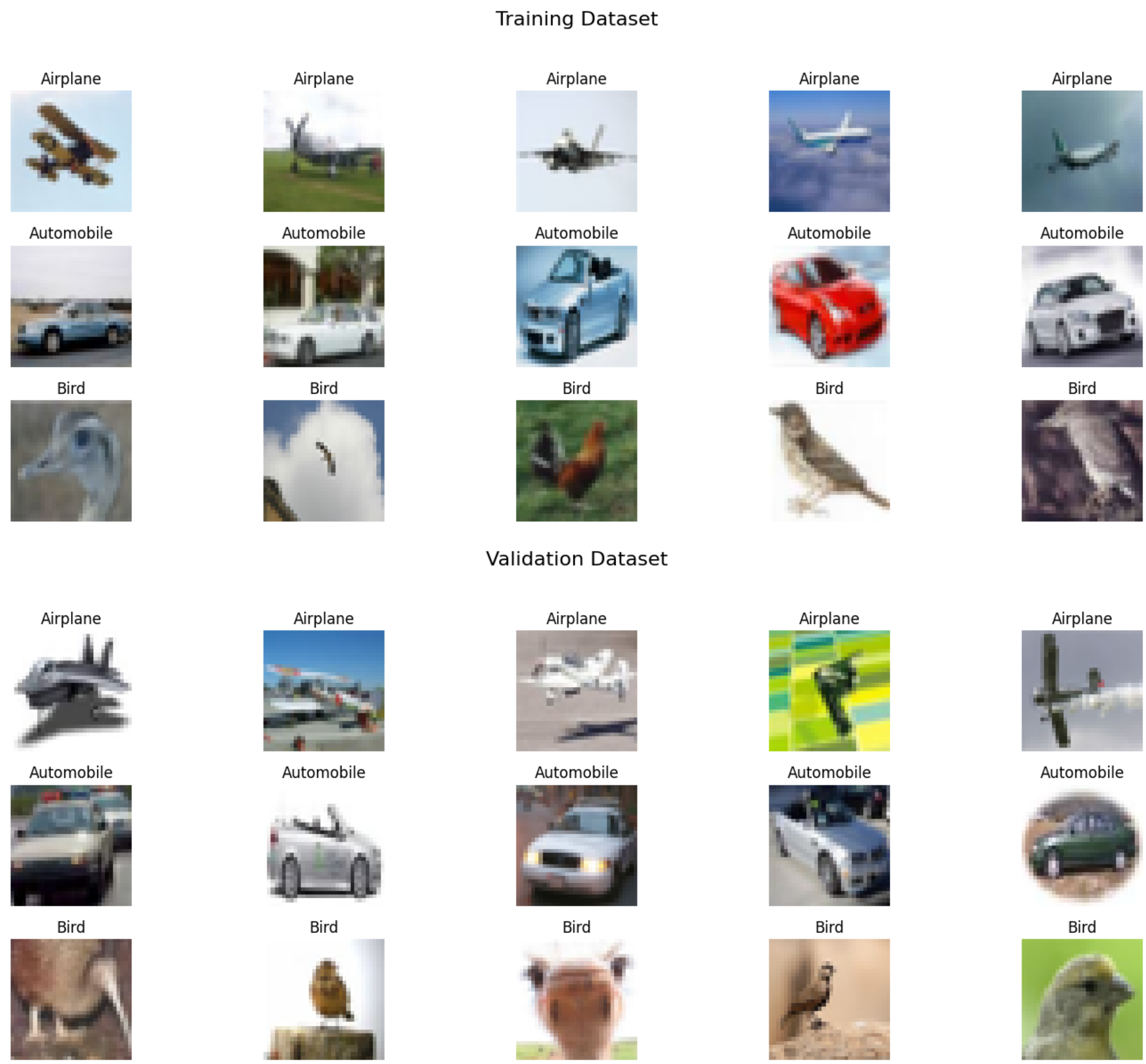
KMeans Clustering with k=4
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroids
3
2
1
0
−1
1
2
3
4
5
6
7
8

SECTION-C

1. Data Preparation

```
Files already downloaded and verified
Files already downloaded and verified
Train dataset size: 12000
Validation dataset size: 3000
Test dataset size: 3000
```

2. Visualization



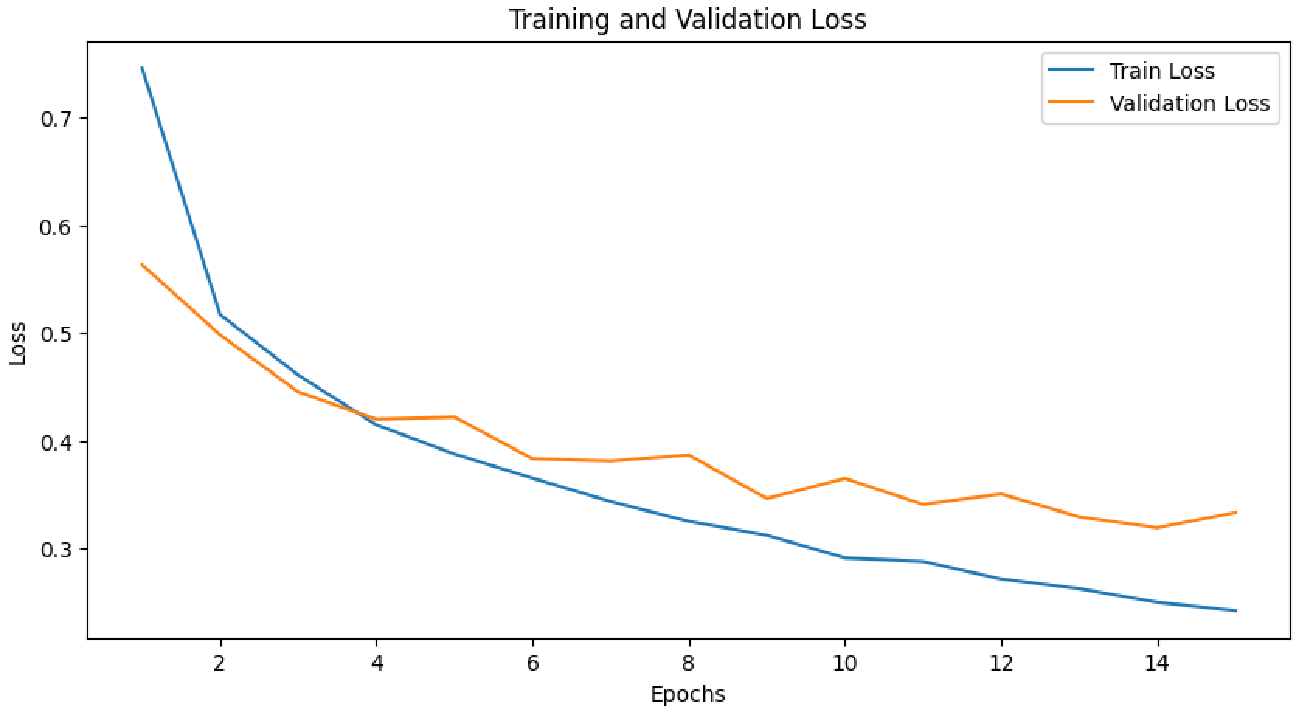
4. Training the model

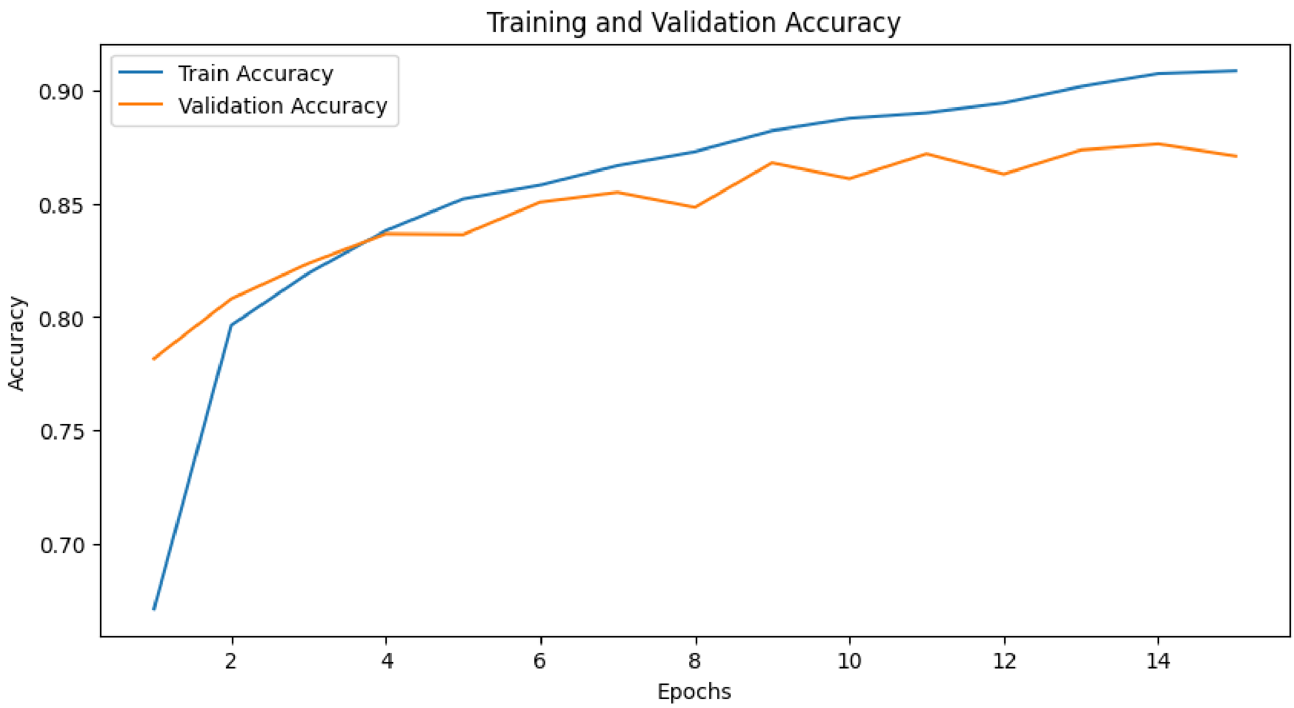
```
Epoch 1/15
Train Loss: 0.7458, Train Accuracy: 0.6713, Val Loss: 0.5634, Val Accuracy: 0.7817
Best model saved at epoch 1
Epoch 2/15
Train Loss: 0.5171, Train Accuracy: 0.7963, Val Loss: 0.4982, Val Accuracy: 0.8080
Best model saved at epoch 2
Epoch 3/15
Train Loss: 0.4610, Train Accuracy: 0.8193, Val Loss: 0.4453, Val Accuracy: 0.8237
Best model saved at epoch 3
Epoch 4/15
Train Loss: 0.4148, Train Accuracy: 0.8381, Val Loss: 0.4200, Val Accuracy: 0.8367
Best model saved at epoch 4
Epoch 5/15
Train Loss: 0.3878, Train Accuracy: 0.8521, Val Loss: 0.4221, Val Accuracy: 0.8363
Epoch 6/15
Train Loss: 0.3653, Train Accuracy: 0.8582, Val Loss: 0.3833, Val Accuracy: 0.8507
Best model saved at epoch 6
Epoch 7/15
Train Loss: 0.3437, Train Accuracy: 0.8668, Val Loss: 0.3813, Val Accuracy: 0.8550
Best model saved at epoch 7
Epoch 8/15
Train Loss: 0.3254, Train Accuracy: 0.8729, Val Loss: 0.3868, Val Accuracy: 0.8483
Epoch 9/15
Train Loss: 0.3124, Train Accuracy: 0.8822, Val Loss: 0.3465, Val Accuracy: 0.8680
Best model saved at epoch 9
Epoch 10/15
Train Loss: 0.2914, Train Accuracy: 0.8877, Val Loss: 0.3651, Val Accuracy: 0.8610
```

```
Train Loss: 0.2881, Train Accuracy: 0.8901, Val Loss: 0.3411, Val Accuracy: 0.8720
Best model saved at epoch 11
Epoch 12/15
Train Loss: 0.2719, Train Accuracy: 0.8944, Val Loss: 0.3507, Val Accuracy: 0.8630
Epoch 13/15
Train Loss: 0.2628, Train Accuracy: 0.9017, Val Loss: 0.3295, Val Accuracy: 0.8737
Best model saved at epoch 13
Epoch 14/15
Train Loss: 0.2503, Train Accuracy: 0.9074, Val Loss: 0.3194, Val Accuracy: 0.8763
Best model saved at epoch 14
Epoch 15/15
Train Loss: 0.2426, Train Accuracy: 0.9087, Val Loss: 0.3335, Val Accuracy: 0.8710
Final model saved.
```

5. Testing

Plot Training and Validation Metrics

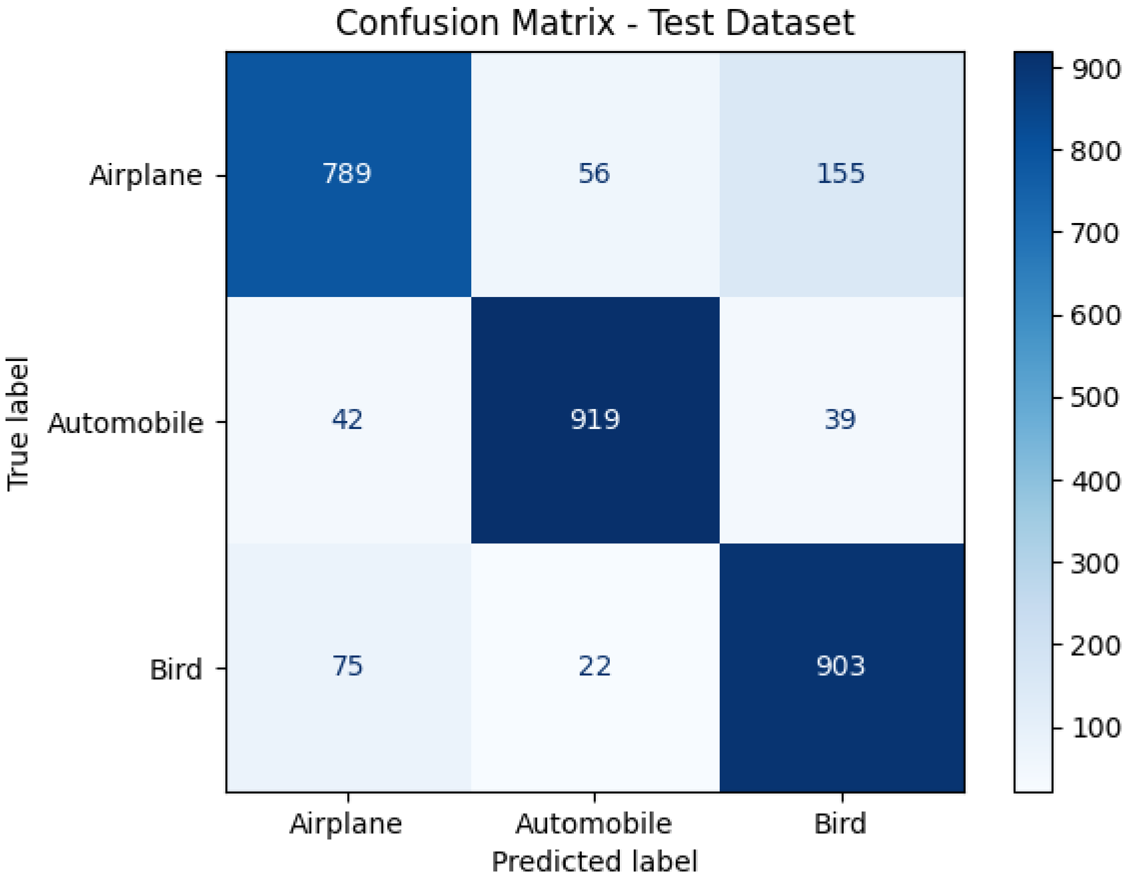


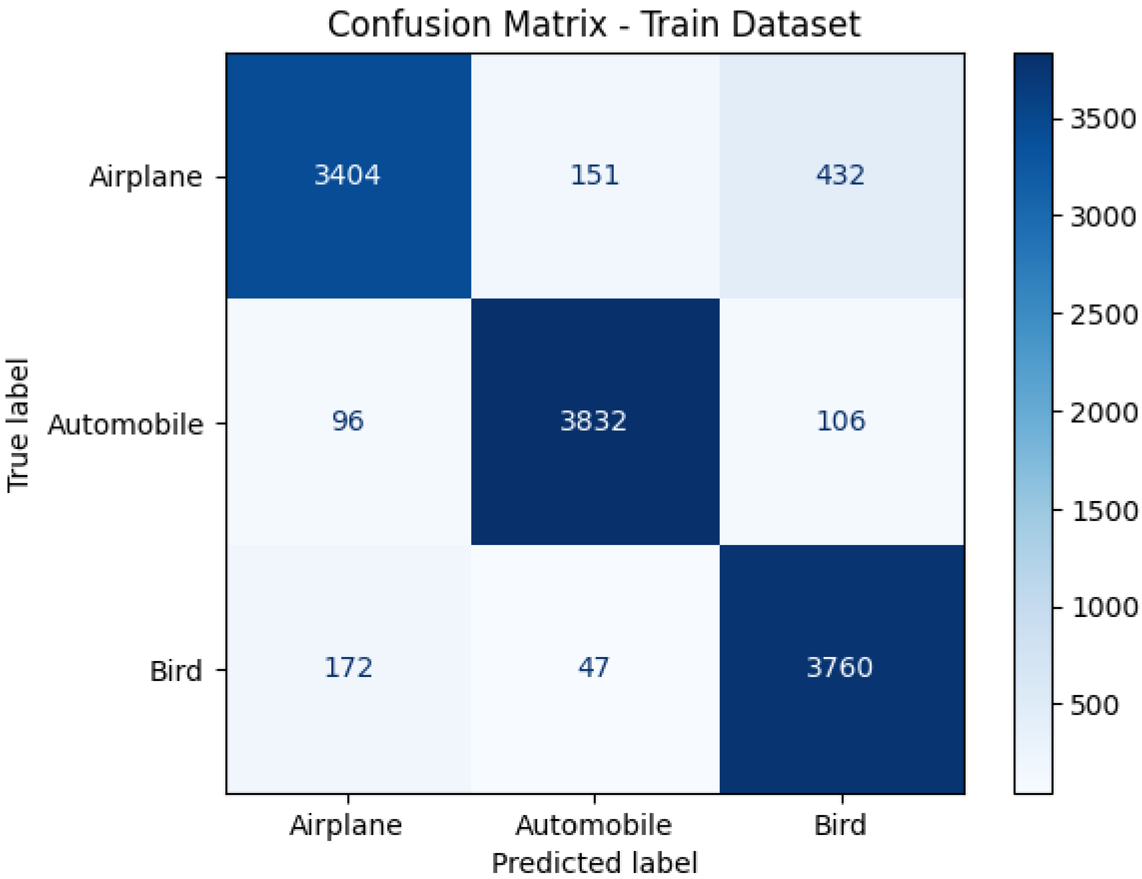


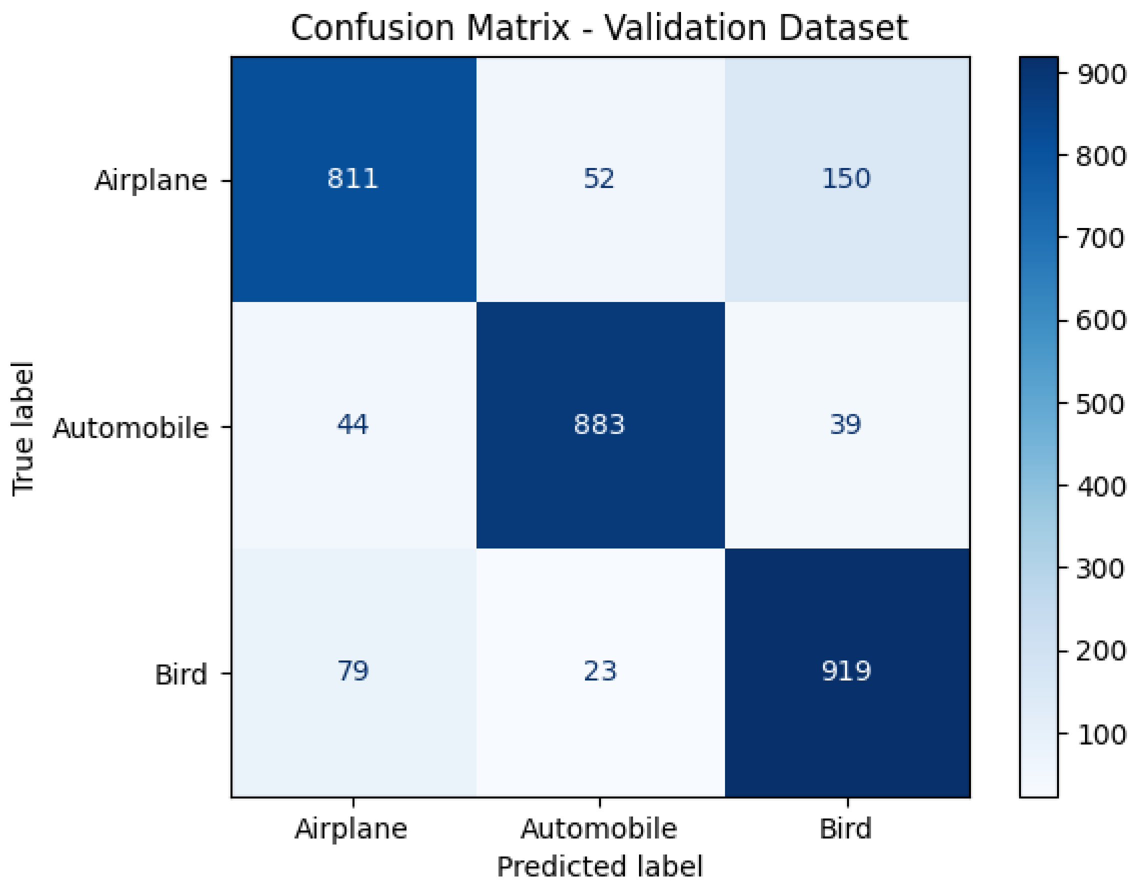
Evaluate Test Dataset Accuracy and F1-Score

```
Test Loss: 0.3294
Test Accuracy: 0.8703
Test F1-Score: 0.8698
```

Plot Confusion Matrices for Train, Val, and Test Datasets


Confusion Matrix - Test Dataset
True label
Predicted label
Airplane
Automobile
Bird
789
56
155
42
919
39
75
22
903
900
800
700
600
500
400
300
200
100


Confusion Matrix - Train Dataset
True label
Predicted label
Airplane
Automobile
Bird
3404
151
432
96
3832
106
172
47
3760
3500
3000
2500
2000
1500
1000
500



6. Training an MLP

Define the MLP Model

```
Model Architecture:
MLPModel(
  (fc1): Linear(in_features=3072, out_features=64, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=64, out_features=3, bias=True)
)
```

Training the MLP Model

```
Epoch 1/15
Train Loss: 0.6841, Train Accuracy: 0.7179, Val Loss: 0.6499, Val Accuracy: 0.7403
Best model saved at epoch 1
Epoch 2/15
Train Loss: 0.5649, Train Accuracy: 0.7773, Val Loss: 0.6175, Val Accuracy: 0.7600
Best model saved at epoch 2
Epoch 3/15
Train Loss: 0.5064, Train Accuracy: 0.8030, Val Loss: 0.5971, Val Accuracy: 0.7667
Best model saved at epoch 3
Epoch 4/15
Train Loss: 0.4601, Train Accuracy: 0.8175, Val Loss: 0.5864, Val Accuracy: 0.7670
Best model saved at epoch 4
Epoch 5/15
Train Loss: 0.4260, Train Accuracy: 0.8363, Val Loss: 0.5889, Val Accuracy: 0.7753
Epoch 6/15
Train Loss: 0.3907, Train Accuracy: 0.8531, Val Loss: 0.5818, Val Accuracy: 0.7740
Best model saved at epoch 6
Epoch 7/15
Train Loss: 0.3660, Train Accuracy: 0.8622, Val Loss: 0.6088, Val Accuracy: 0.7803
Epoch 8/15
Train Loss: 0.3417, Train Accuracy: 0.8728, Val Loss: 0.6088, Val Accuracy: 0.7767
Epoch 9/15
Train Loss: 0.3097, Train Accuracy: 0.8832, Val Loss: 0.6058, Val Accuracy: 0.7860
Epoch 10/15
Train Loss: 0.2992, Train Accuracy: 0.8866, Val Loss: 0.6569, Val Accuracy: 0.7787
```

```
Epoch 11/15
Train Loss: 0.2764, Train Accuracy: 0.9008, Val Loss: 0.6424, Val Accuracy: 0.7793
Epoch 12/15
Train Loss: 0.2509, Train Accuracy: 0.9085, Val Loss: 0.7072, Val Accuracy: 0.7670
Epoch 13/15
Train Loss: 0.2286, Train Accuracy: 0.9192, Val Loss: 0.6404, Val Accuracy: 0.7860
Epoch 14/15
Train Loss: 0.2047, Train Accuracy: 0.9277, Val Loss: 0.6920, Val Accuracy: 0.7743
Epoch 15/15
Train Loss: 0.1947, Train Accuracy: 0.9310, Val Loss: 0.7125, Val Accuracy: 0.7877
Final MLP model saved.
```

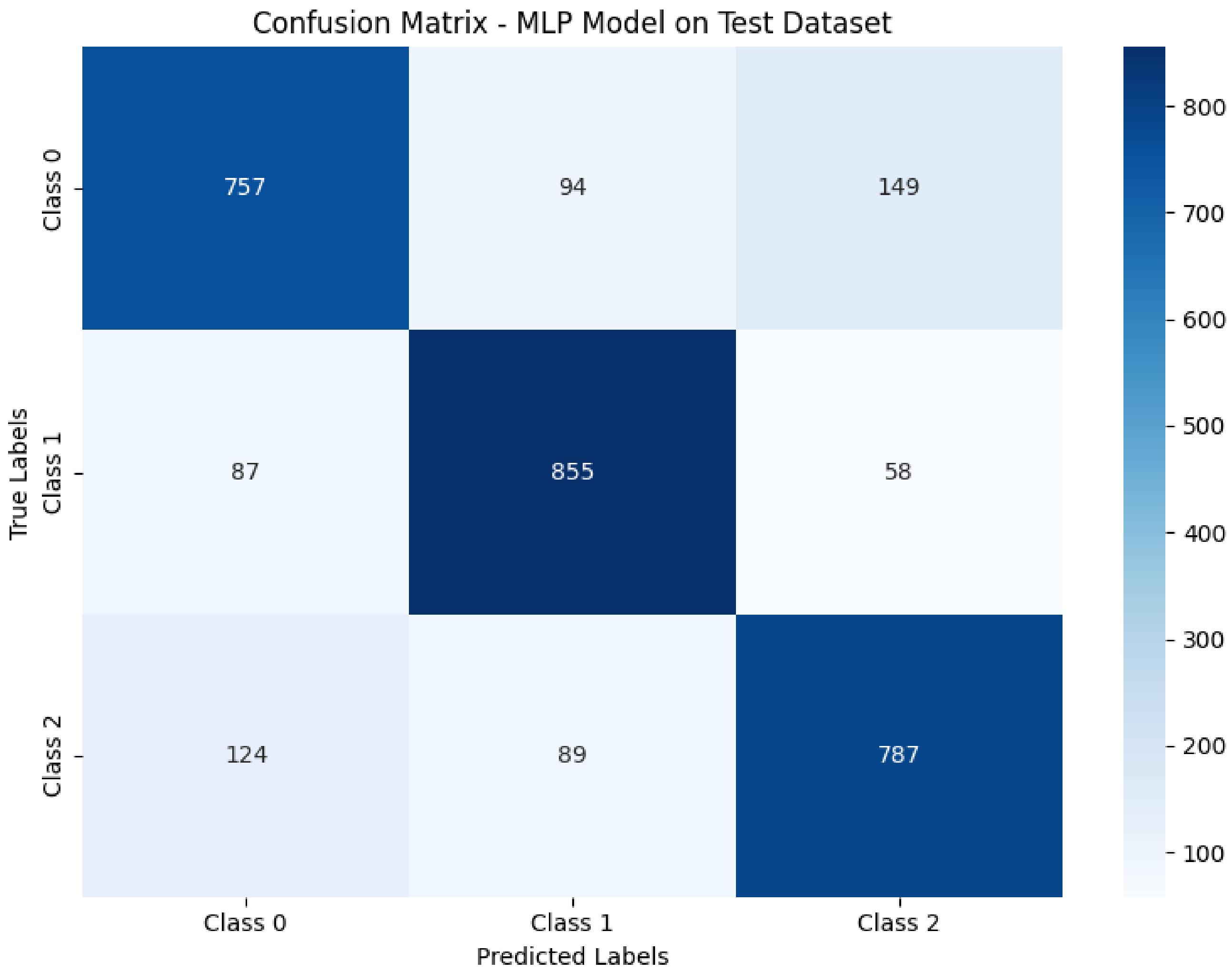
7. Infer and Compare

Evaluate MLP Model on Test Data

```
MLP Test Performance:
Loss: 0.5212, Accuracy: 0.7997, F1-Score: 0.7992
```

Plot Confusion Matrix for MLP Model

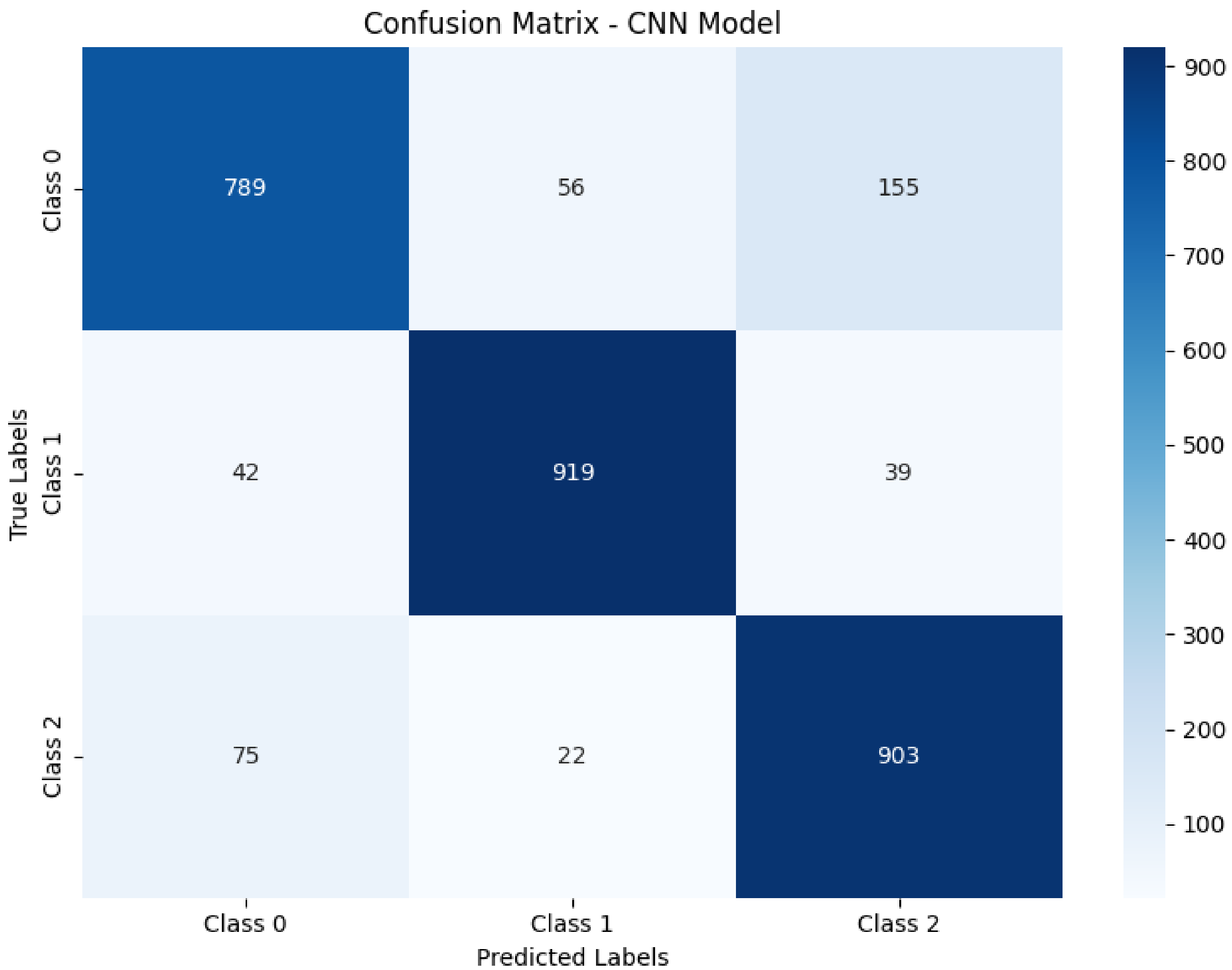
```
Confusion Matrix:
[[757  94 149]
 [ 87 855  58]
 [124  89 787]]
```



Compare Results with CNN

```
Model Performance Comparison:
----------------------------------------------------
Metric          CNN             MLP
----------------------------------------------------
Test Accuracy   0.8703 |   0.7997
Test F1-Score   0.8698 |   0.7992

Comparing Confusion Matrices:
----------------------------------------------------
CNN Model Confusion Matrix:
Confusion Matrix:
[[789  56 155]
 [ 42 919  39]
 [ 75  22 903]]
```



```
MLP Model Confusion Matrix:
Confusion Matrix:
[[757  94 149]
 [ 87 855  58]
 [124  89 787]]
```