Section -A

A)

(a) Dimensions of the resulting feature map

Given the input dimensions M×N with P channels and a kernel size of K×K, with stride = 1 and no padding:

The resulting feature map dimensions will be:

(M−K+1)×(N−K+1)

(b) Number of elementary operations for a single output pixel

To compute a single output pixel, the kernel slides over a K×K region for each channel:

- **Multiplications**: For each channel, there are K×K weights. Across P channels, there are P×K×K multiplications.
- **Additions**: After multiplications, the results are summed, which requires (K×K×P)-1 additions.

Thus, the total elementary operations per output pixel are:

Operations=P×K×K(multiplications)+((K×K×P)-1)(additions)
=2×K×K×P-1

(c) Time complexity for the entire image

With Q kernels:

1. Each kernel computes a feature map of size (M−K+1)×(N−K+1).
2. Each feature map computation involves (M−K+1)×(N−K+1) output pixels.

Elementary operations per output pixel: 2×P×K×K−1.

Total operations for one kernel:

Operations per kernel=(M−K+1)×(N−K+1)×(2×P×K×K−1).

For Q kernels:

Total operations=Q×(M−K+1)×(N−K+1)×(2×P×K×K−1).

Big-O time complexity: For the entire computation, ignoring constants:

O(Q×P×K2×(M−K+1)×(N−K+1))

Assuming min(M,N)$\gg$K : Here,$M-K+1\approx M$ and $N-K+1\approx N$, so the complexity becomes, so the complexity becomes:

$O(Q\times P\times K\times K\times M\times N)$

B)

K-Means Algorithm: Assignment Step and Update Step

1. Assignment Step:
    - For each data point, compute the distance to each cluster centroid (using Euclidean distance).
    - Assign each data point to the cluster with the nearest centroid.
    - This creates new clusters by grouping data points that are closest to the same centroid.
2. Update Step:
    - For each cluster, calculate the mean of all the data points assigned to it.
    - Update the centroid of the cluster to this mean.
    - This ensures that the centroid represents the "center" of its assigned points.

These two steps are repeated iteratively until the centroids no longer change significantly or a maximum number of iterations is reached.

Elbow Method:

- Plot the Within-Cluster Sum of Squares (WCSS) against the number of clusters K.
    - WCSS measures the total distance of points from their cluster centroids.
- As K increases, WCSS decreases because points are grouped more closely.
- Look for an "elbow point" where the rate of WCSS decrease slows down significantly. This indicates the optimal K.

Random initialization of centroids in the K-Means algorithm will not always work because it can lead to **poor convergence** or **suboptimal results**. The objective of K-Means is to minimize the **Within-Cluster Sum of Squares (WCSS)**, given by:

$$WCSS = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2$$

Here:

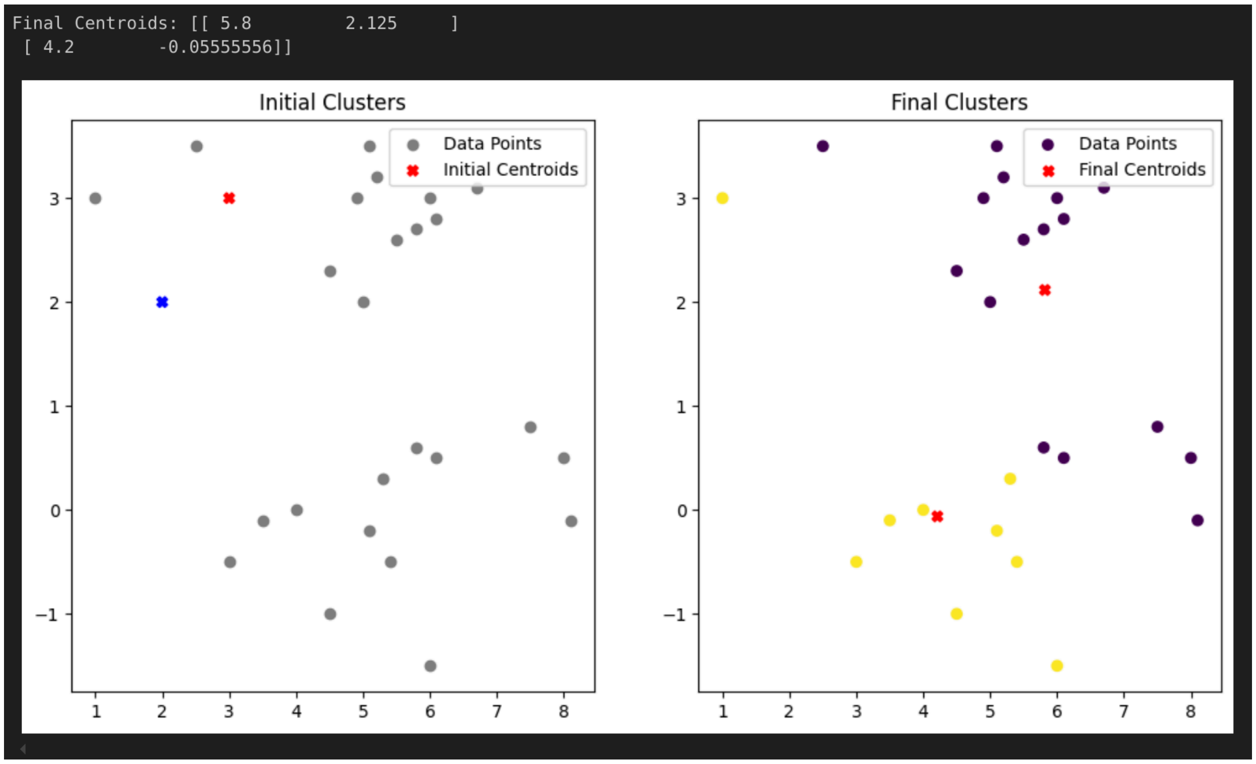
- K is the number of clusters.
- Ck represents the set of points in cluster k.
- μk\ is the centroid of cluster k.
- // x−μk // ^2 is the squared distance between a point x and its cluster centroid μk.

Since the objective function is **non-convex**, the algorithm can converge to a **local minimum** rather than the global minimum, depending on the initial positions of the centroids. Random initialization may result in centroids that:

1. Are too close together, leading to redundant clusters.
2. Are far from the true cluster centers, requiring more iterations or failing to converge to an optimal solution.

Section -B

b)



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

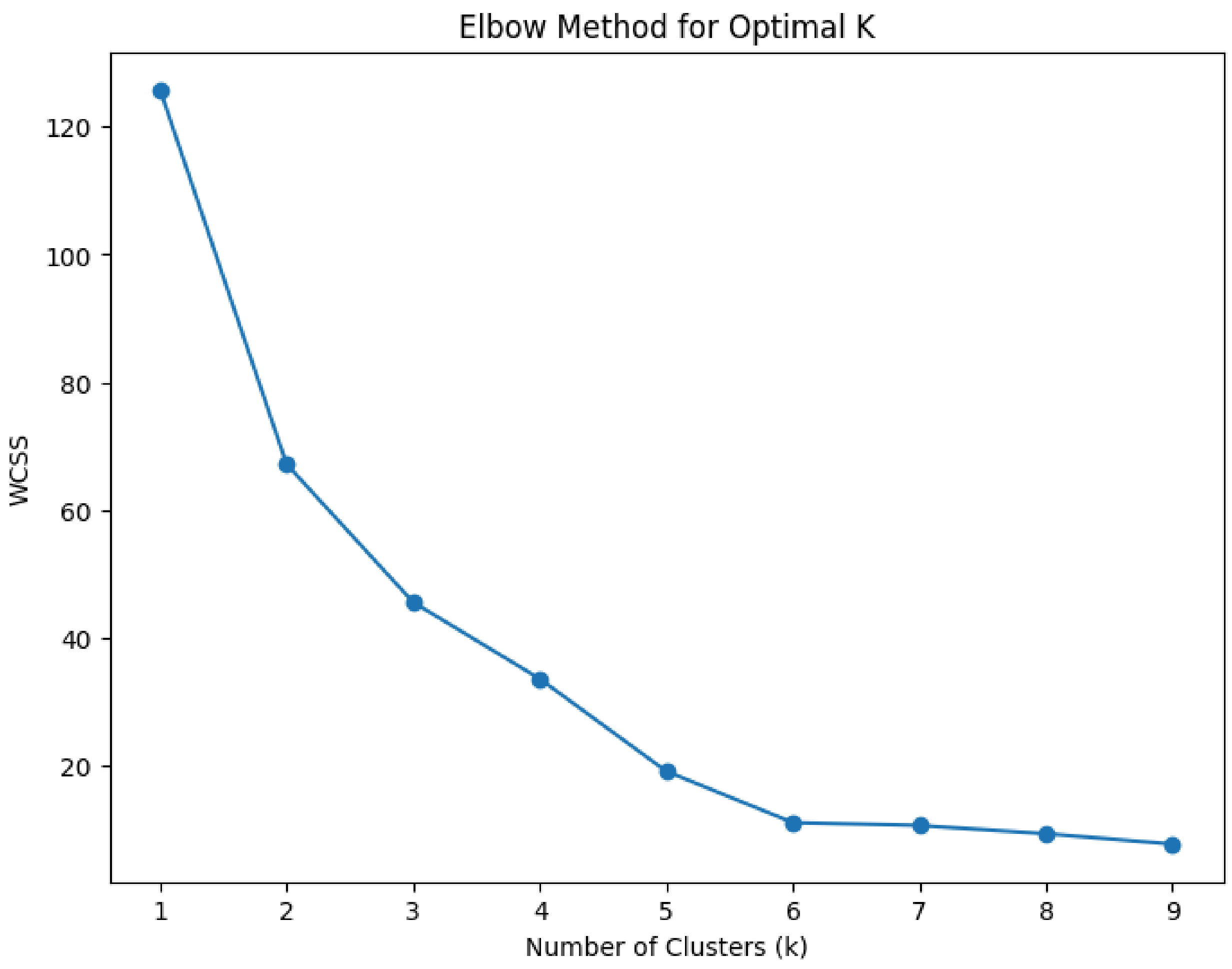
c)

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

Random Initialization Final Centroids: [[4.85833333 2.89166667]
 [5.56153846 -0.09230769]]

The results show that both initialization strategies lead to centroids that converge to different points. This difference is expected due to the random nature of centroid initialization, which can lead to variations in clustering outcomes. However, both sets of centroids likely represent stable solutions that minimize the clustering objective (WCSS), but the provided centroids may start closer to the optimal solution, potentially leading to faster convergence.

d)

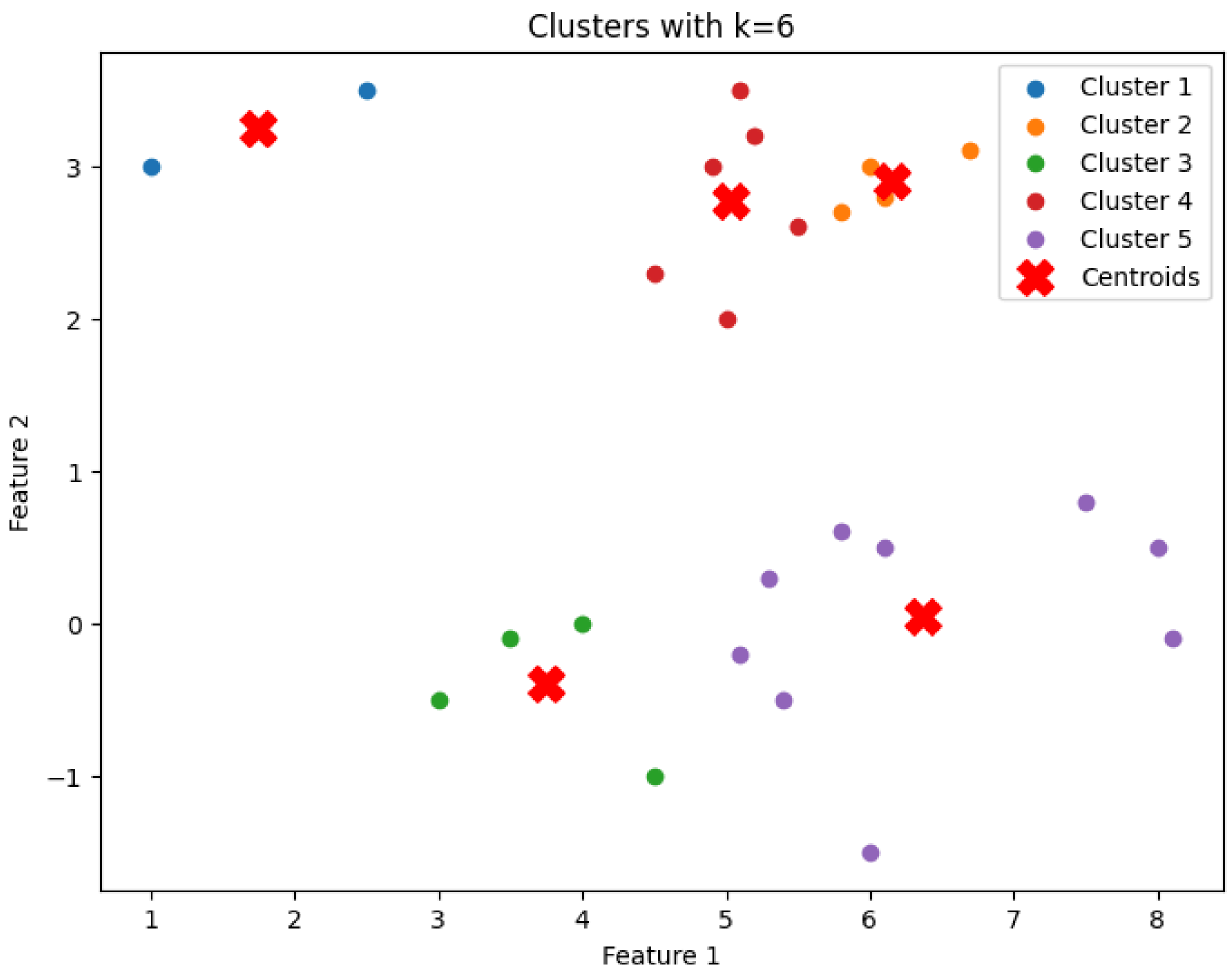


To determine the optimal number of clusters, the Elbow Method was applied by plotting the Within-Cluster Sum of Squares (WCSS) against different values of k (as shown in the graph above). The graph displays a sharp decline in WCSS as k increases, followed by a slower decrease.

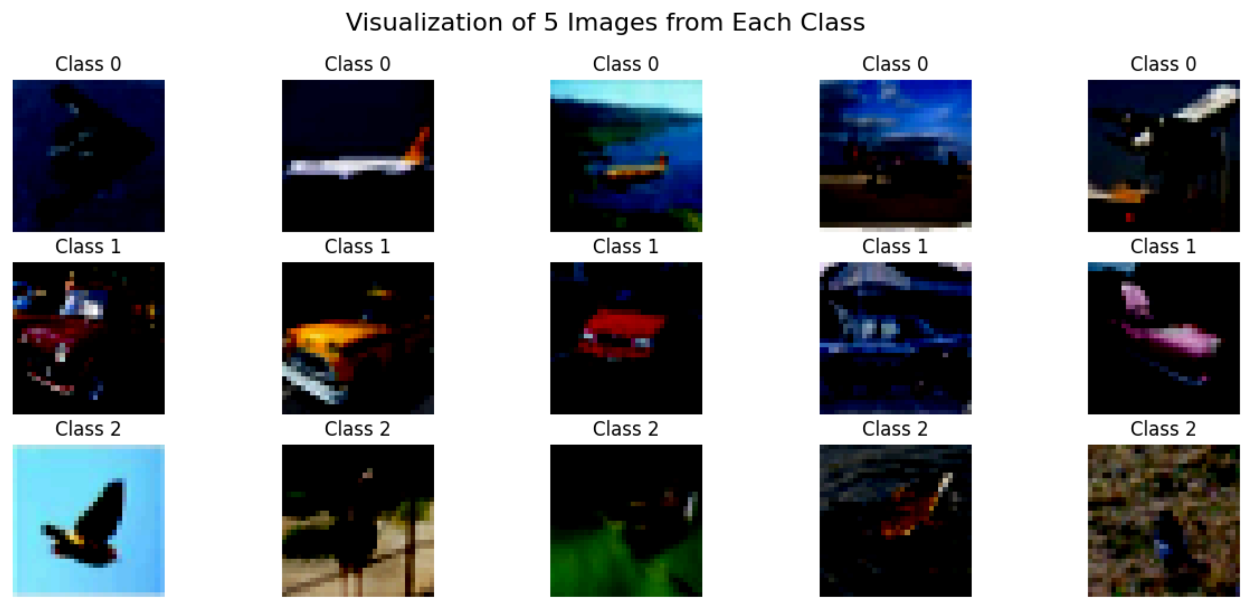
The "elbow point" is observed at k=5x, where the rate of decrease in WCSS slows significantly. This indicates that three clusters capture most of the variance in the data while avoiding overfitting.

Using M=5(the optimal number of clusters), centroids were randomly initialized, and clustering was performed. The resulting clusters showed a clear grouping of the data into three meaningful clusters, as indicated by the reduction in WCSS

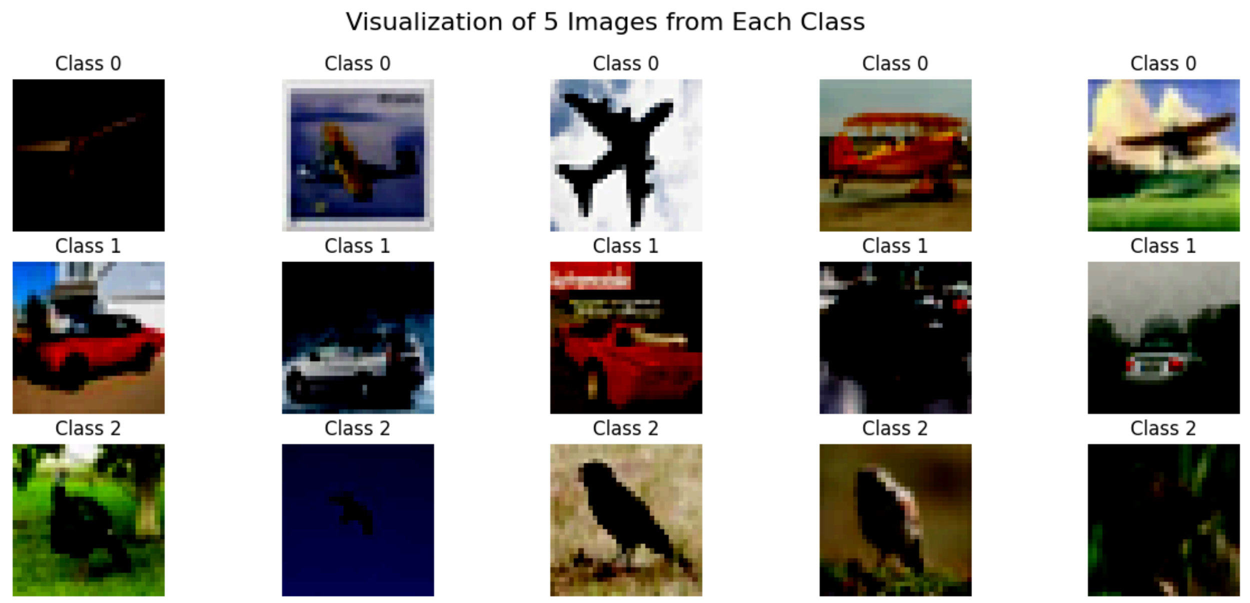
```
Centroids: [[ 1.75       3.25      ]

 [ 6.15       2.9       ]
 [ 3.75        -0.4       ]
 [ 5.03333333  2.76666667]
 [ 6.36666667  0.04444444]]
```



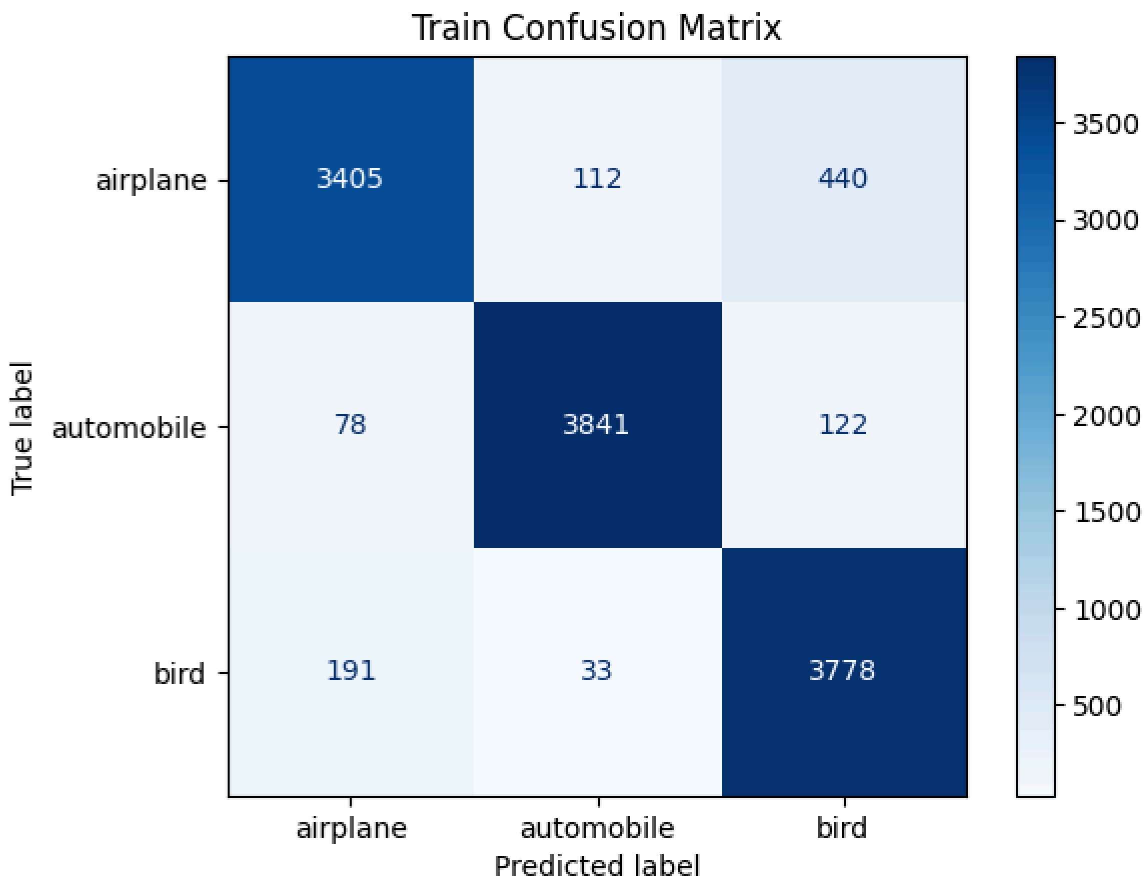
Section -C
2)
training-

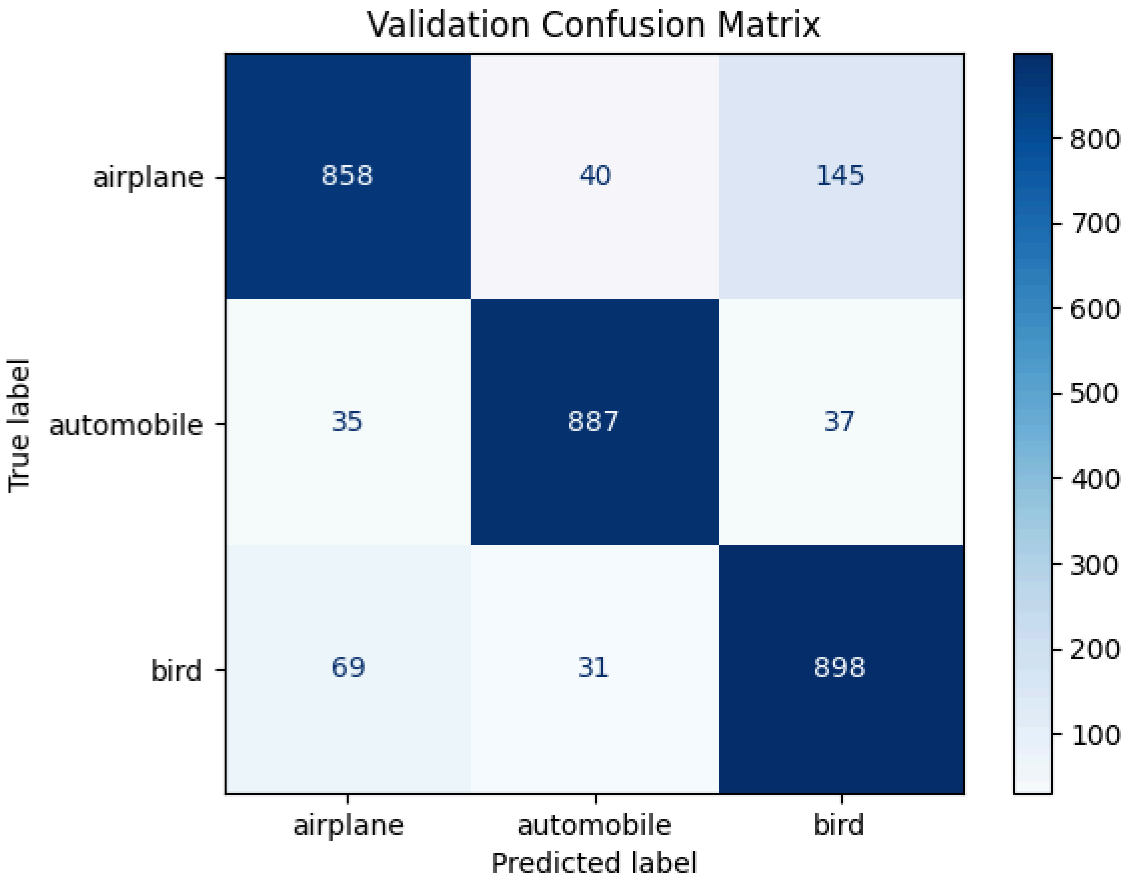


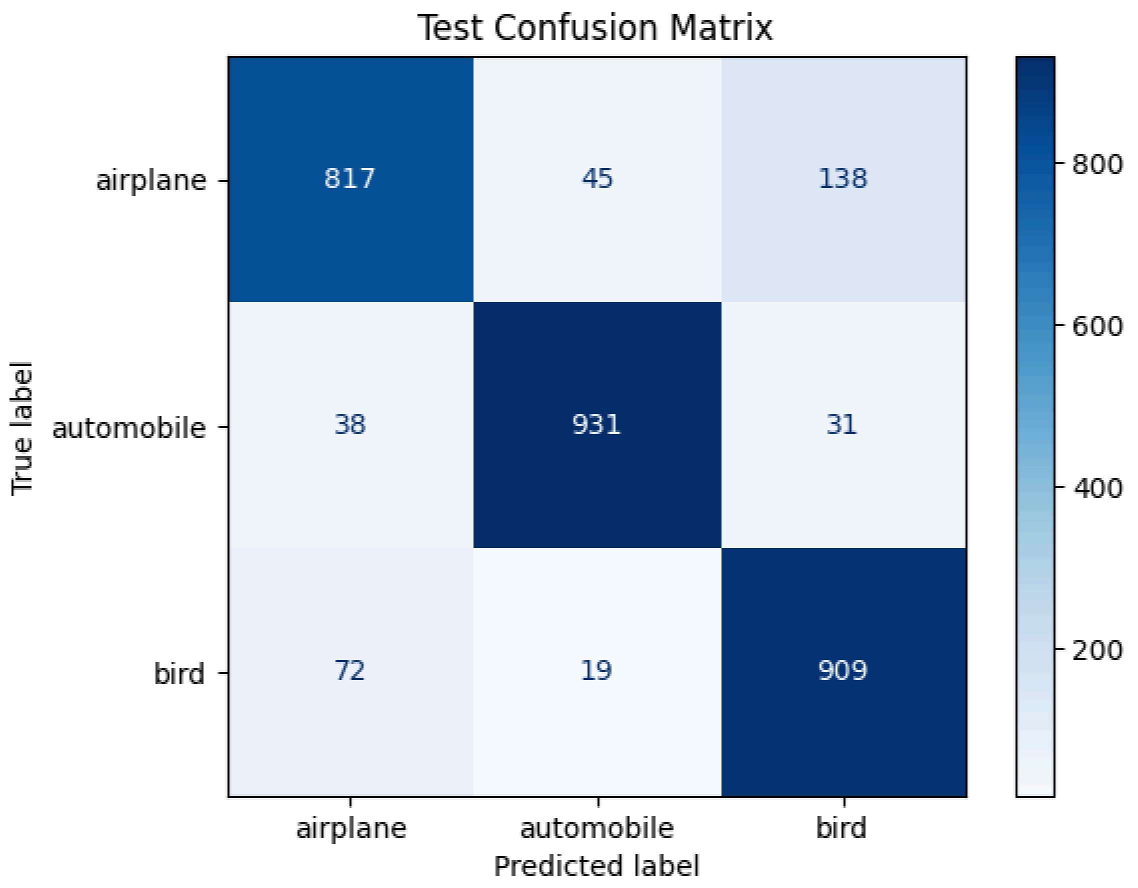
val-

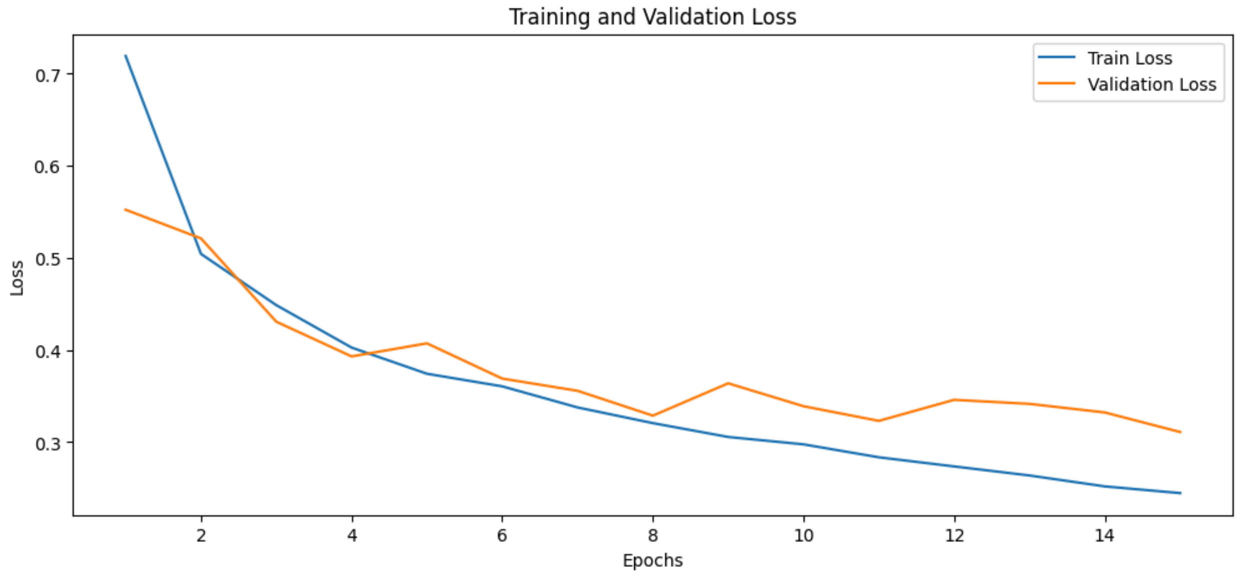


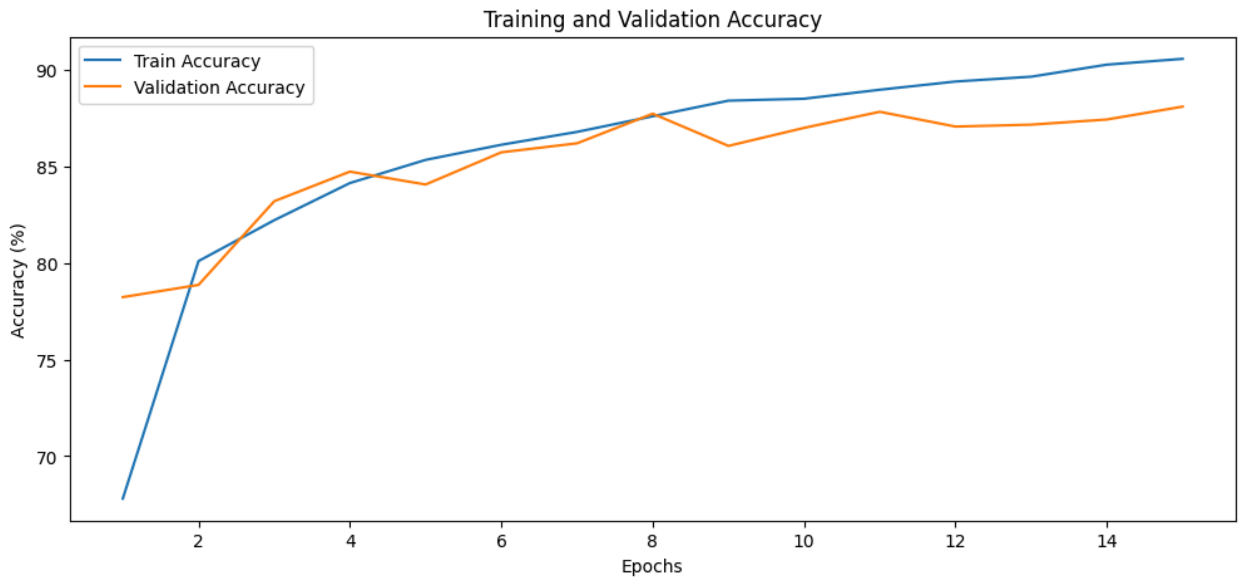
5)
FIndings-
Train Accuracy: 0.9187, F1-score: 0.9186
Validation Accuracy: 0.8810, F1-score: 0.8809
Test Accuracy: 0.8857, F1-score: 0.8854


Train Confusion Matrix
airplane
automobile
bird
True label
3405
112
440
78
3841
122
191
33
3778
airplane
automobile
bird
Predicted label
3500
3000
2500
2000
1500
1000
500


Validation Confusion Matrix
True label
Predicted label
airplane
automobile
bird
858
40
145
35
887
37
69
31
898
800
700
600
500
400
300
200
100


Test Confusion Matrix
True label
airplane
automobile
bird
817
45
138
38
931
31
72
19
909
Predicted label
800
600
400
200


Training and Validation Loss
Train Loss
Validation Loss
Loss
Epochs
0.7
0.6
0.5
0.4
0.3
2
4
6
8
10
12
14



Training and Validation Loss/Accuracy:

- **Training Accuracy:** 91.87%
- **Validation Accuracy:** 88.10%
- **Test Accuracy:** 88.57%

From the training and validation accuracy plots, we can observe that the model's training accuracy is consistently higher than the validation accuracy. This is a typical sign of slight **overfitting**, where the model learns the training data very well but struggles a bit when generalizing to unseen validation data. However, the validation accuracy is still quite high, which suggests that the model is generalizing well overall, with no severe overfitting.

The **loss curves** for both the training and validation sets decrease steadily, indicating that the model is effectively learning over the epochs. The gap between the training and validation loss is small, indicating that overfitting is not too pronounced, although some minor overfitting is visible due to the difference in accuracy between the two.

Test Dataset Accuracy and F1-score:

- **Test Accuracy:** 88.57%
- **Test F1-score:** 88.54%

The model performs well on the test set, achieving high accuracy and a strong F1-score. The relatively small difference between training, validation, and test accuracies suggests that the model generalizes well to unseen data. This indicates that the CNN architecture has successfully captured the relevant features of the images.

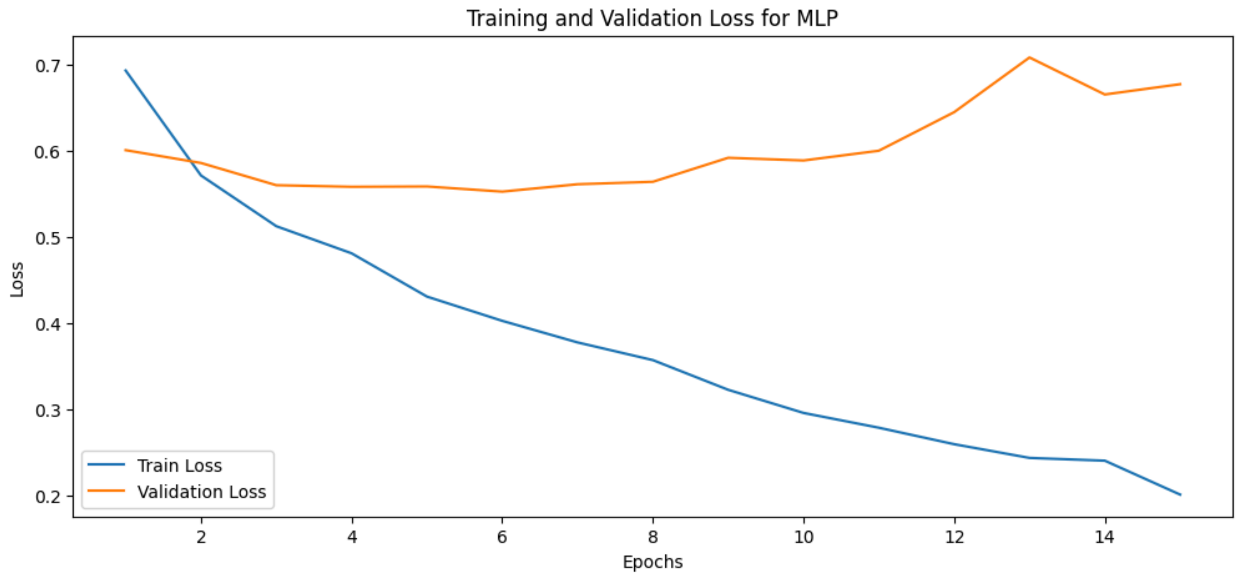
Confusion Matrix:

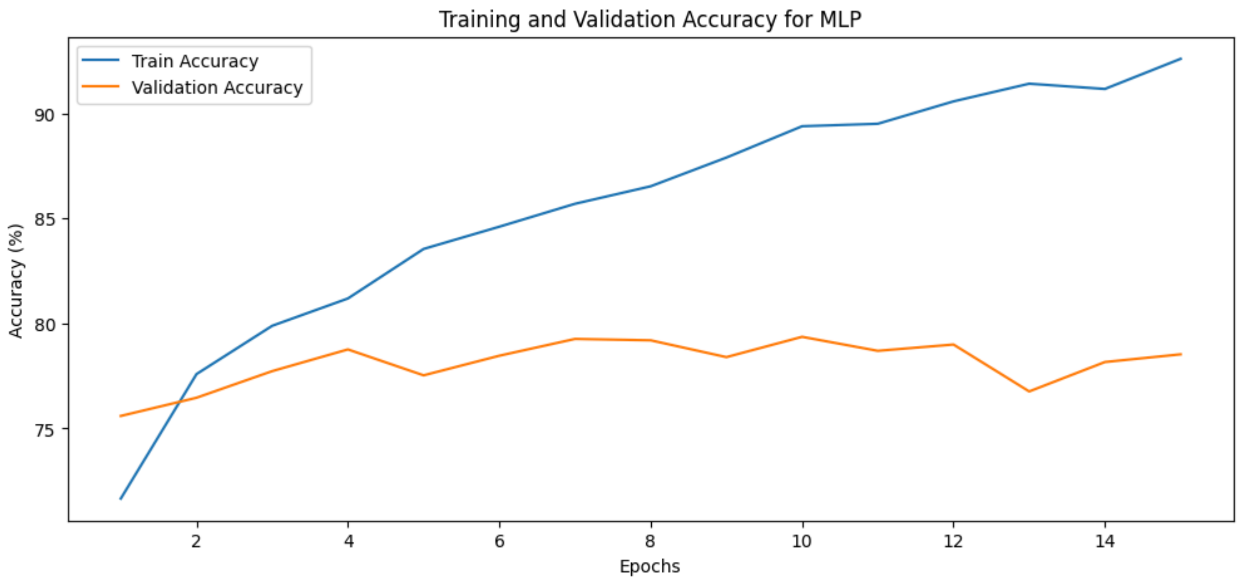
The **confusion matrix** for the dataset shows how well the model classifies each of the 3 classes . In our case in the confusion matrix, the diagonal elements (true positives) are dominating, with few misclassifications.

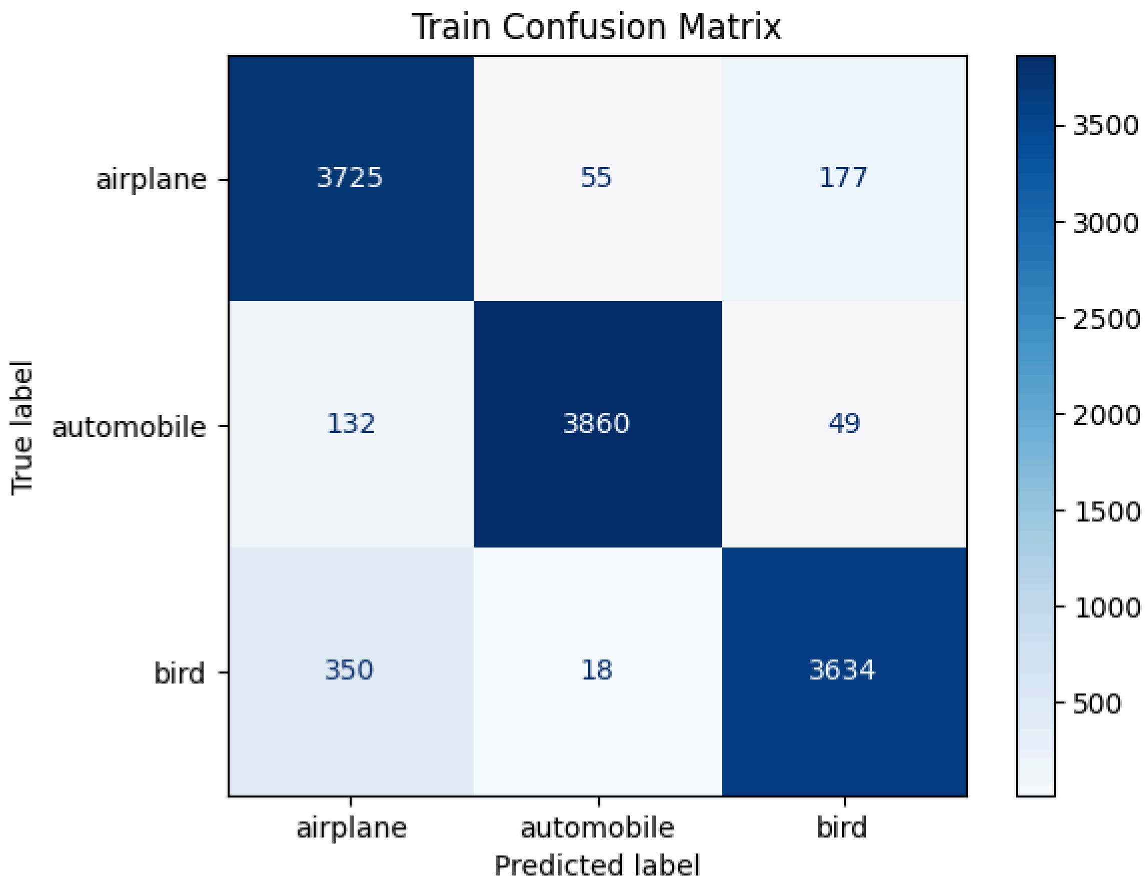
7)
Train Dataset - Accuracy: 93.49%, F1-Score: 0.9352

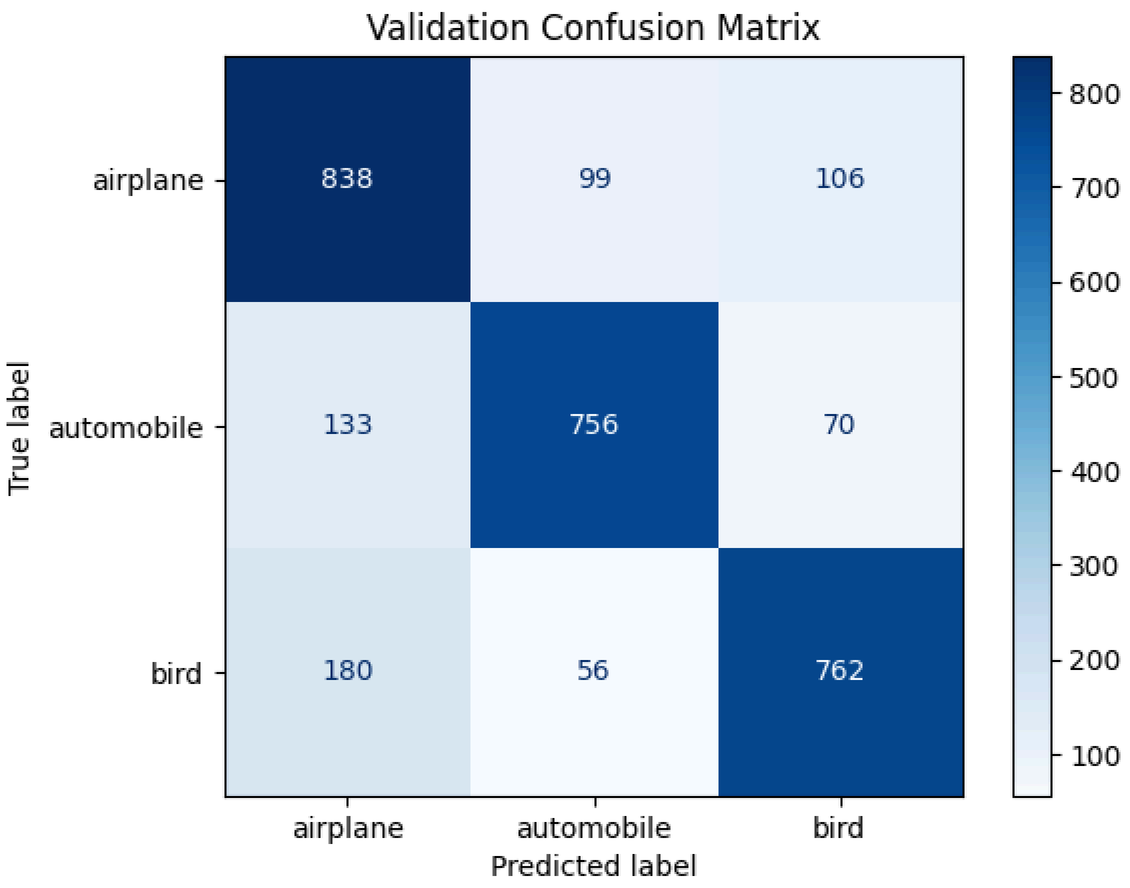
Validation Dataset - Accuracy: 78.53%, F1-Score: 0.7859

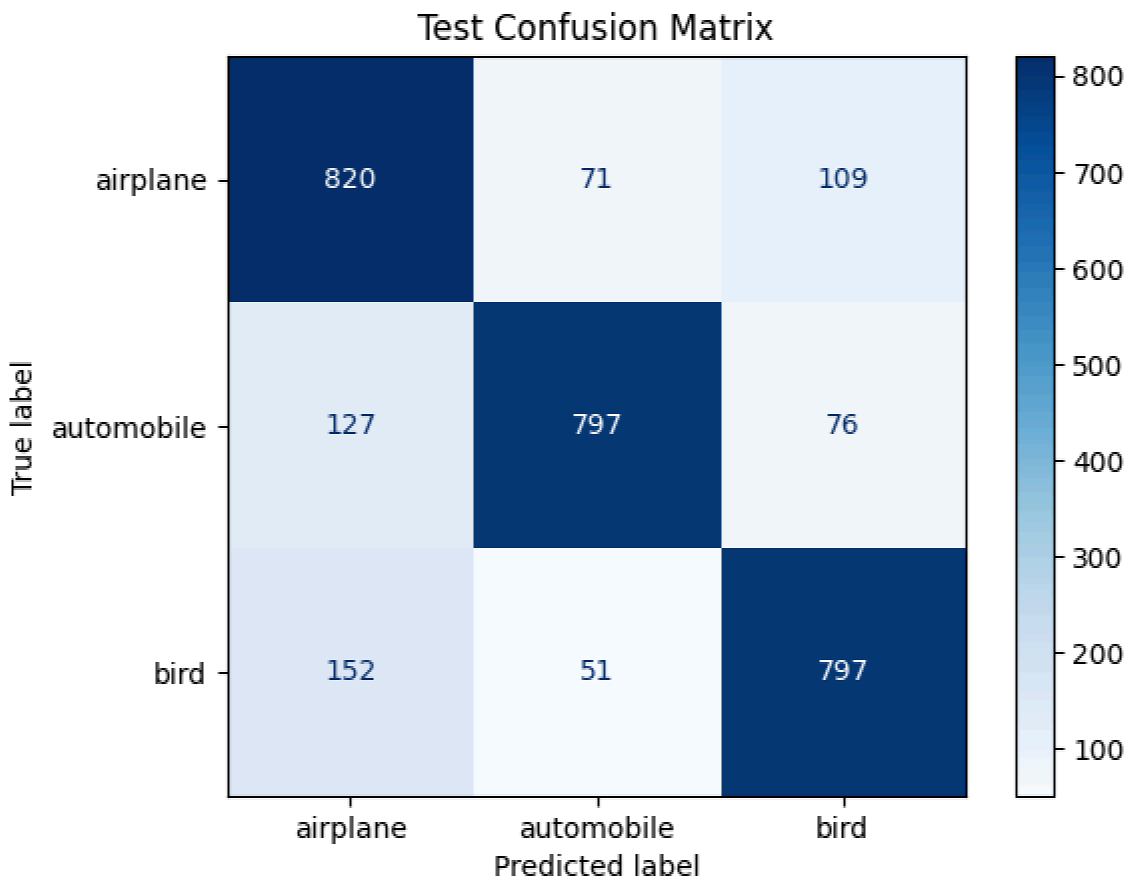
Test Dataset - Accuracy: 80.47%, F1-Score: 0.8054




Training and Validation Accuracy for MLP
Train Accuracy
Validation Accuracy
Accuracy (%)
Epochs
90
85
80
75
2
4
6
8
10
12
14


Train Confusion Matrix
True label
Predicted label
airplane
automobile
bird
3725
55
177
132
3860
49
350
18
3634
3500
3000
2500
2000
1500
1000
500


Validation Confusion Matrix
True label
Predicted label
airplane
automobile
bird
airplane
automobile
bird
838
99
106
133
756
70
180
56
762
100
200
300
400
500
600
700
800



Comparison of CNN and MLP Models:

Accuracy Comparison:

- **CNN:**
  - **Train Accuracy:** 91.87%
  - **Validation Accuracy:** 88.10%
  - **Test Accuracy:** 88.57%
- **MLP:**
  - **Train Accuracy:** 93.49%
  - **Validation Accuracy:** 78.53%
  - **Test Accuracy:** 80.47%

The **CNN model** has slightly lower training accuracy compared to the MLP but achieves **better validation and test accuracy**. The MLP model shows **overfitting** as indicated by a significant gap between its training and validation/test accuracy (93.49% vs. 78.53% on validation, 80.47% on test). In contrast, the **CNN model** generalizes better with a relatively small gap between training (91.87%), validation (88.10%), and test accuracy (88.57%).

F1-Score Comparison:

- **CNN:**
  - **Test F1-score:** 88.54%
- **MLP:**
  - **Test F1-score:** 80.54%

The **CNN model** significantly outperforms the MLP in terms of the **F1-score** on the test dataset, indicating better balance between precision and recall. The MLP struggles with generalization, leading to a much lower F1-score.

Confusion matrix Comparison:

The **confusion matrix** reveals that the **CNN model** has **fewer misclassifications** compared to the **MLP model**, which shows a higher rate of errors. This difference highlights the CNN's ability to capture spatial relationships and local patterns in images, leading to better generalization and more accurate classifications, especially for visually similar classes. In contrast, the **MLP model** struggles with generalization due to its inability to preserve spatial information, resulting in **more misclassifications**, particularly on the test data. Overall, the CNN model outperforms the MLP in both accuracy and F1-score, as it is better suited for image classification tasks.

Overfitting:

- **CNN:** The CNN shows some signs of overfitting, but it is relatively mild, with the training accuracy being only slightly higher than the validation accuracy. The test results show the model generalizes well.
- **MLP:** The MLP shows more **severe overfitting**, with a large difference between training accuracy (93.49%) and validation/test accuracy (78.53% validation, 80.47% test). This suggests that the MLP model has memorized the training data but does not generalize well to new, unseen data.

Performance on Test Set:

- The **CNN model** (88.57% accuracy, 88.54% F1-score) clearly outperforms the **MLP model** (80.47% accuracy, 80.54% F1-score) on the test dataset. CNN's ability to capture spatial features via convolutions gives it a significant edge over the MLP, which treats images as flattened vectors without considering spatial relationships.

Conclusion:

- **CNN Model:** The CNN model provides better generalization and performs significantly better on the test set. Its architecture is better suited for image classification tasks, particularly for datasets like CIFAR-10, where spatial hierarchies and local patterns are important.
- **MLP Model:** The MLP model suffers from overfitting and is less effective at generalizing to unseen data. While it performs well on the training set, it struggles with validation and test accuracy.

Reference :Campusx 100 days of machine learning series
https://www.youtube.com/playlist?list=PLKnIA16_Rmvbr7zKYQuBfsVkjoLcJgxHH
Reference :Campusx 100 days of deep learning series
https://www.youtube.com/watch?v=2dH_qjc9mFg&list=PLKnIA16_RmvYuZauWaPlRTC54KxSNLtNn