CSE343: Machine Learning
Assignment-3

REPORT
Anushka Srivastava (2022086)

SECTION A

a. Part a

Q1. a)

a) Output height = $\left[\dfrac{M-k+2 * \text{Padding}}{\text{Stride}}\right]+1$

Output width = $\left[\dfrac{N-k+2 * \text{Padding}}{\text{Stride}}\right]+1$

Given stride $=1$ and padding $\to 0$

Output height $= M-k+1$
Output width $= N-k+1$

$\therefore$ Dimensions of resulting feature map
$= (M-k+1)(N-k+1)$

b) Kernel size $= k \times k = k^2$ positions in the input image

For 1 channel,
No. of multiplication operations $= k^2$
No. of addition operations $= k^2-1$, result of k^2 multiplication summed up to 1 value.

For P channels,
Total operations $= P(k^2 + k^2 - 1)$
$\Rightarrow P(2k^2-1)$

c) Total number of output features in feature map:

$$(M-k+1) * (N-k+1)$$

Cost to compute 1 pixel for 1 kernel

$$= P(2k^2-1)$$

Total cost, for Q Kernels

$$\Rightarrow Q \times P(2k^2-1)(M-k+1)(N-k+1)$$

$$\Rightarrow O(Q \times P \times k^2 \times (M-k+1)(N-k+1))$$

Since $\min(M, N) >> k$

Final complexity:

$$\Rightarrow O(Q \cdot P \cdot k^2 \cdot M \cdot N)$$

b. B part

Q1 b.) Assignment step

We assign each data point to the nearest cluster based on a chosen distance metric with respect to the centroids.

Each datapoint $x_i \in$ cluster C_j

Cluster assignment

$$= \underset{j}{\operatorname{argmin}} \|x_i - C_j\|$$

Update step.

After assignment, the centroids for each cluster are recalculated.

The new centroid is calculated as:

$$C_j = \frac{1}{|C_j|} \sum_{x_i \in j} x_i$$

We can use the Elbow method to determine the optimal number of clusters.

- Plot the WCSS (within cluster sum of squares) against k.

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

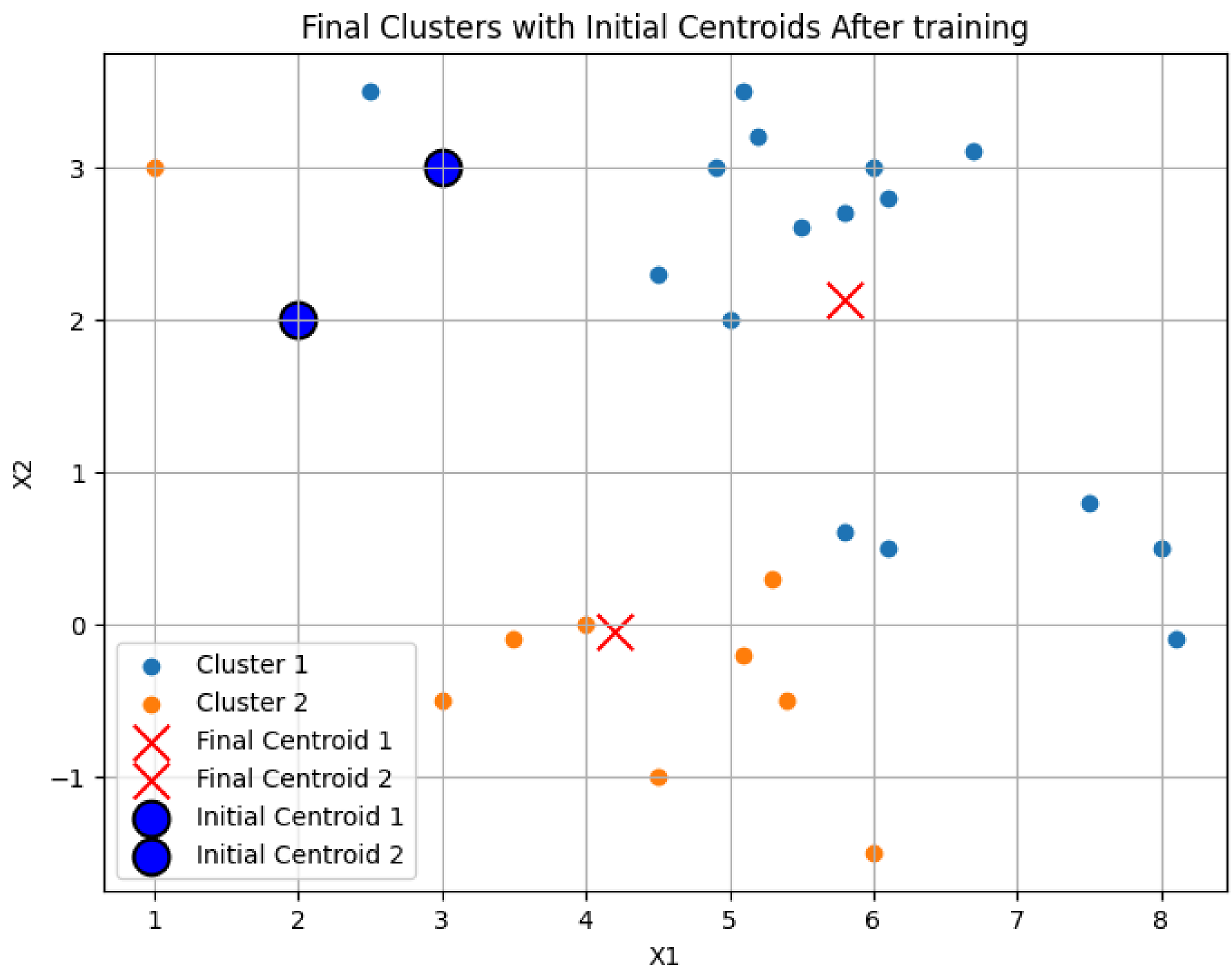
- Identify the elbow point, where the rate of decrease of WCSS slows down significantly. This represents the ~~elbo~~ optimal number of clusters.

Randomly assigning initial cluster centroids can lead to convergence but it is not guaranteed to find the global minimum. Different centroids gives different results each time and it may converge to a local minima but not a global minima.

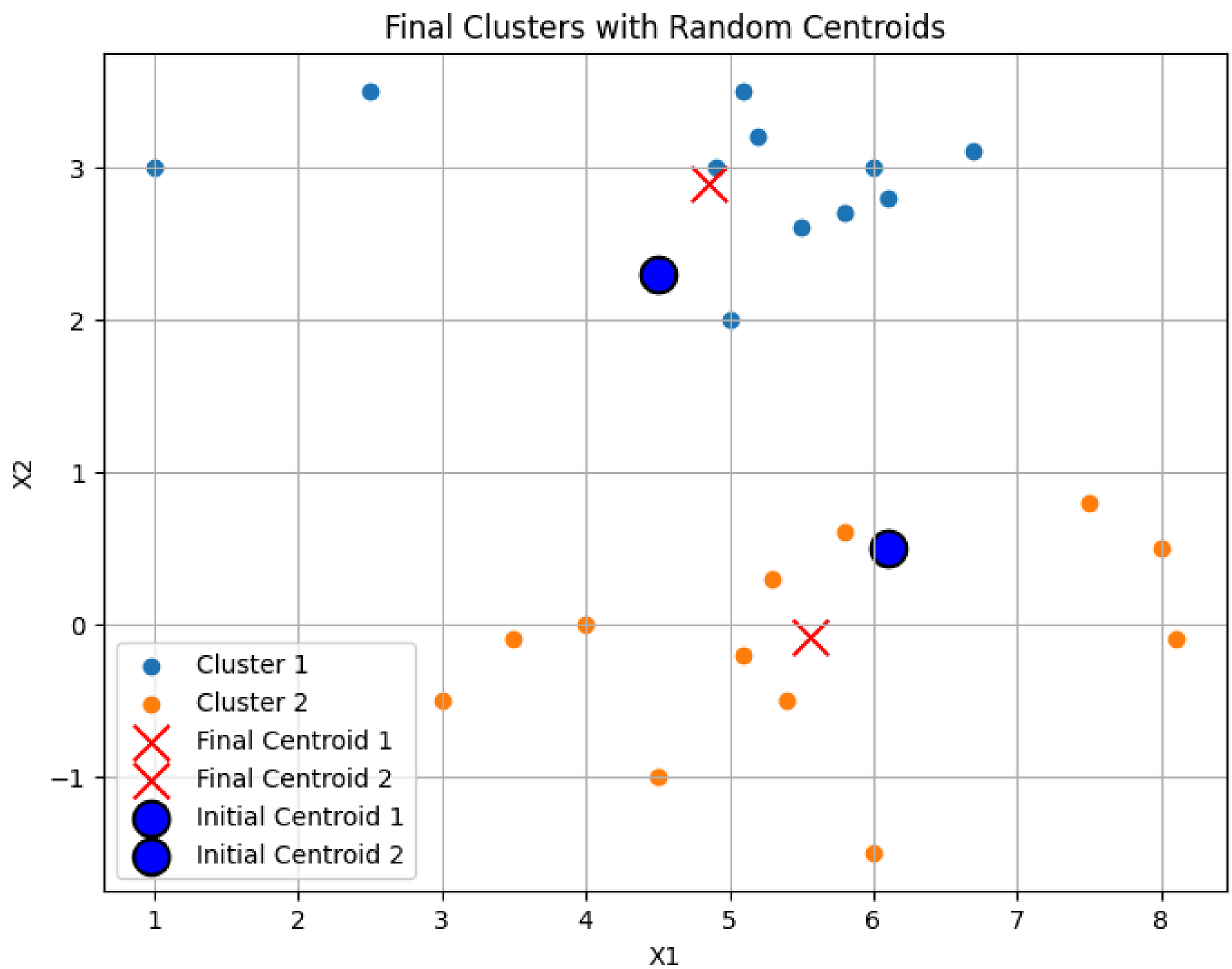
SECTION B

a. We create a class K_Means to implement the K-Means Clustering algorithm and implement methods according to the given instructions.

b. The final value of centroids is:
Centroid 1: (5.8, 2.125)
Centroid 2: (4.2, -0.056)

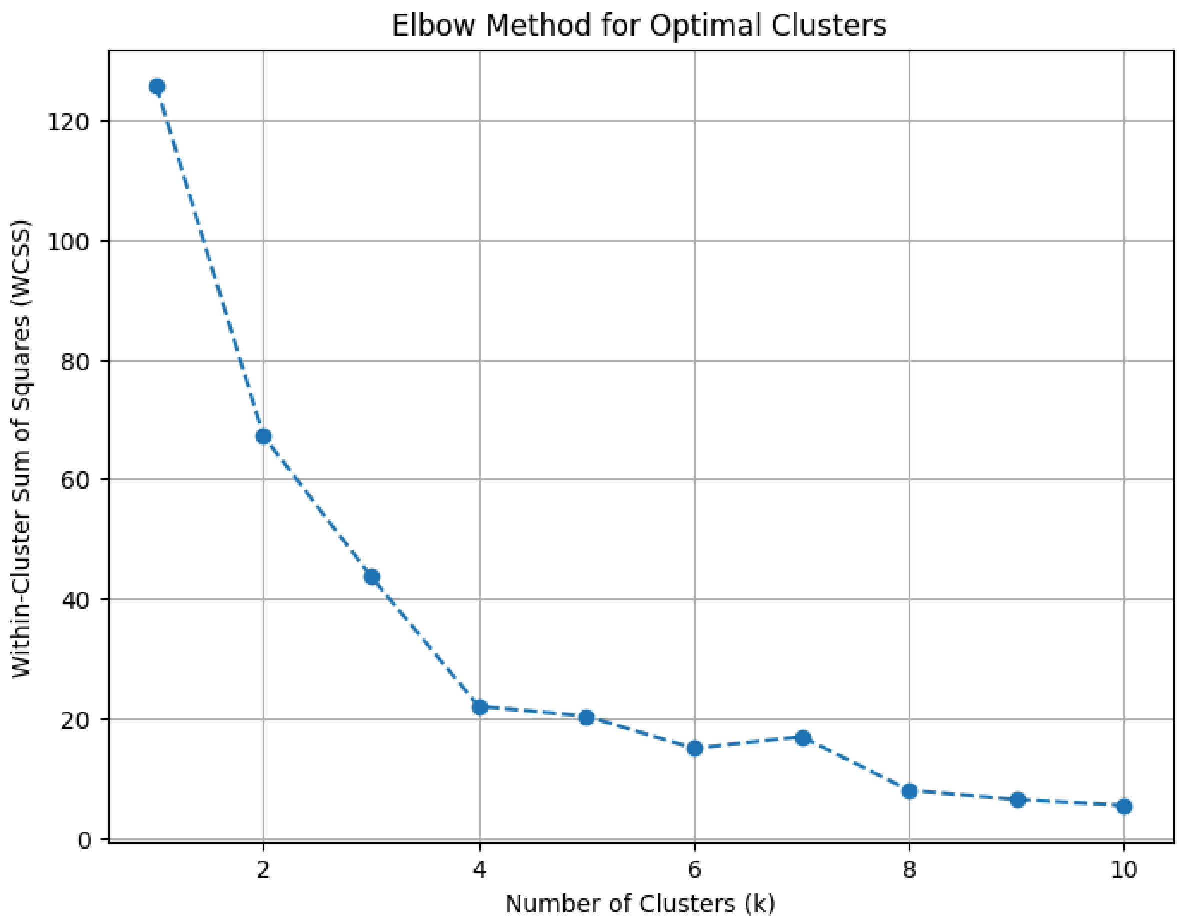


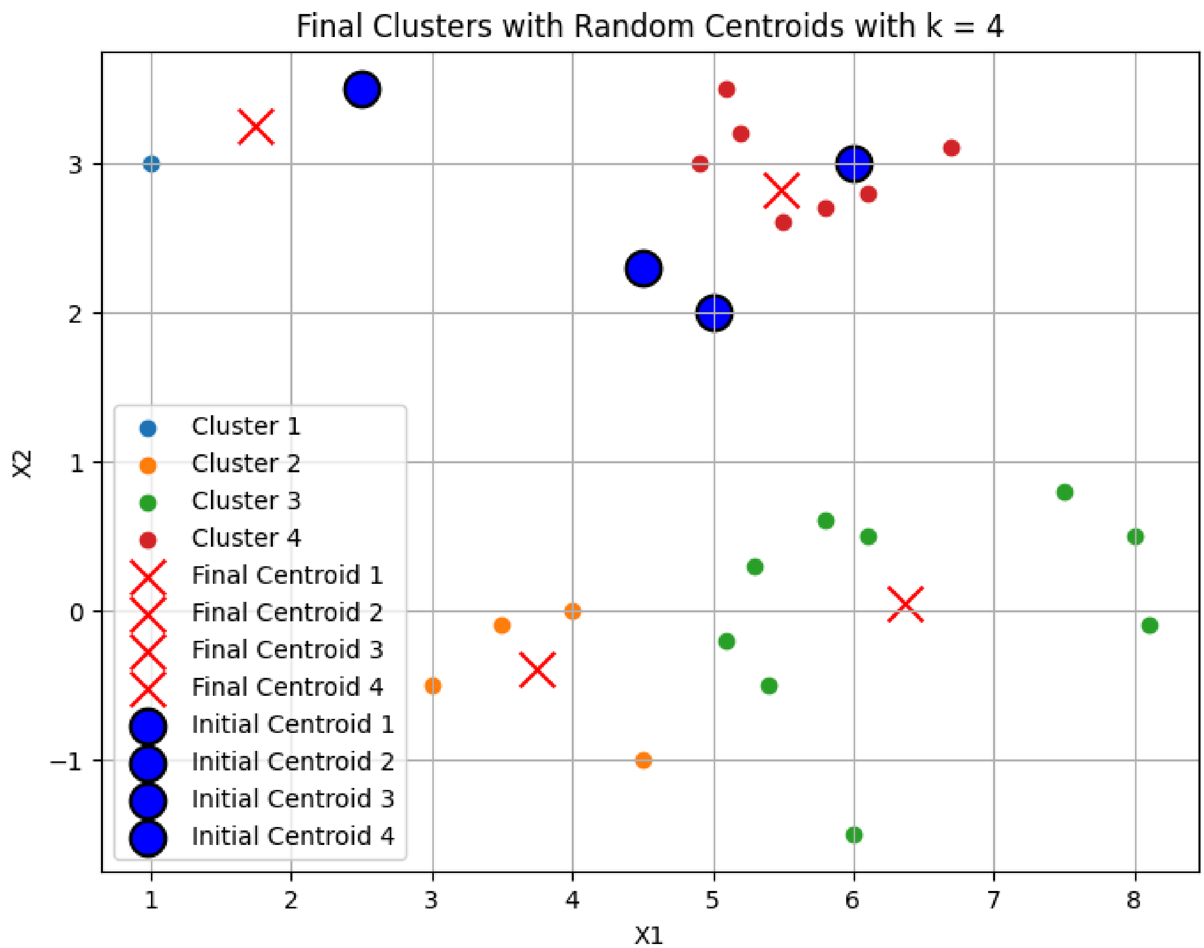
c. Random initialization of centroids



The inertia value for the graph in the b part is 83.67, and the inertia value for the graph with random centroids is 67.15. A lower inertia value indicates a more compact cluster. In this case, random centroid initialization led to better clustering. This could be because the random centroids were placed closer to the natural groupings of the data, allowing the algorithm to converge to a better solution. We cannot comment on that aspect since both algorithms converge in about three iterations.

d. Using the WCSS method, we notice that the optimal value of k is 4.

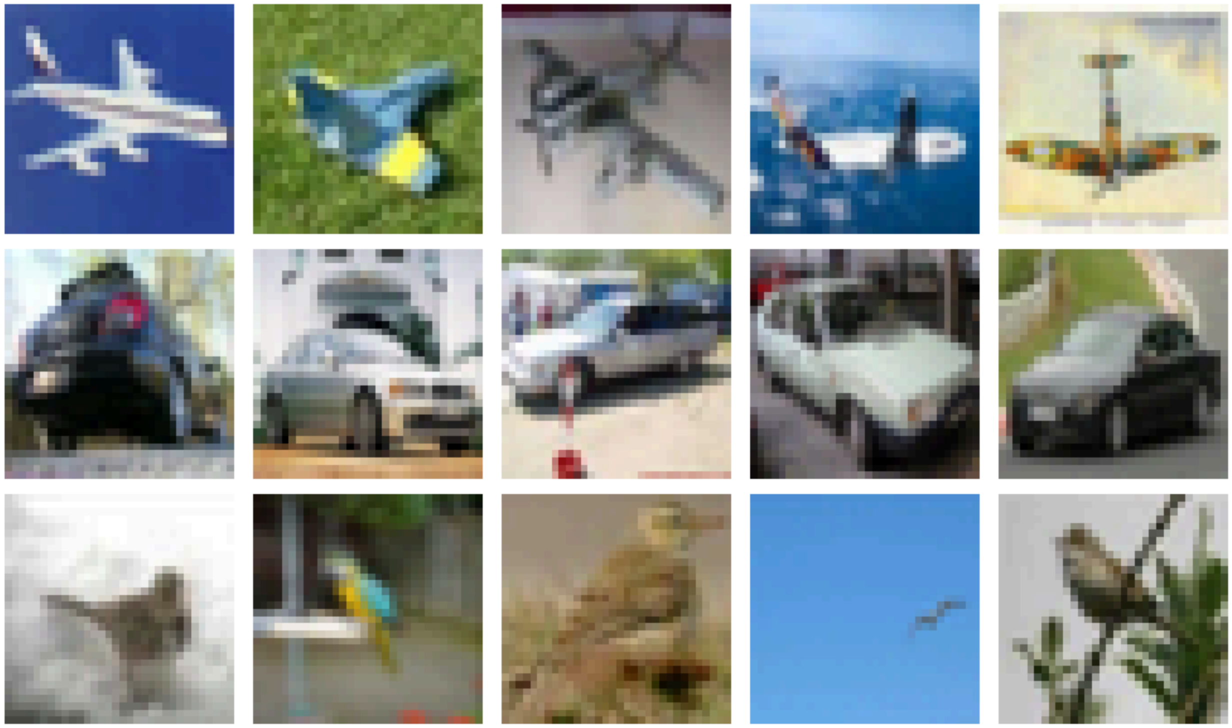

Elbow Method for Optimal Clusters
Within-Cluster Sum of Squares (WCSS)
Number of Clusters (k)
0
20
40
60
80
100
120
2
4
6
8
10



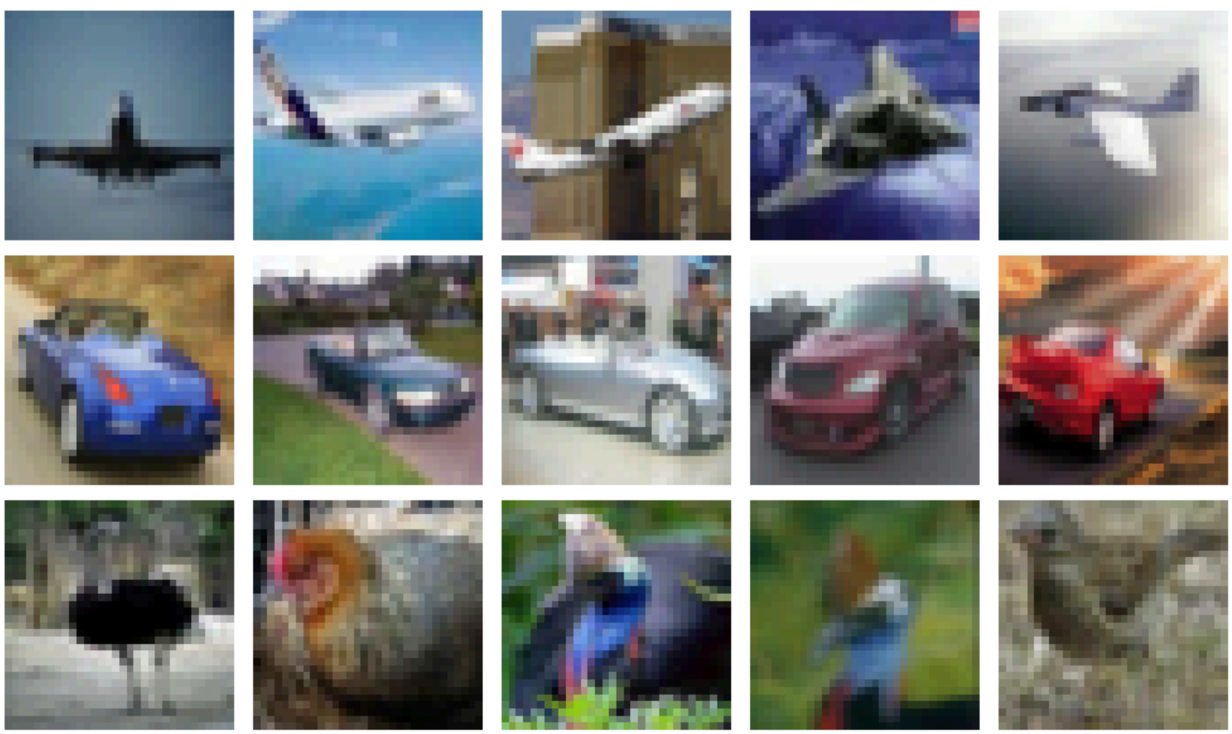
SECTION C (BONUS)

1. A custom Dataset class named CIFAR10Dataset is created, and the train, val, and test data loaders are loaded. The size of the train data loader is 12000, and the size of the val and test data loader is 3000.

2. Training dataset visualization



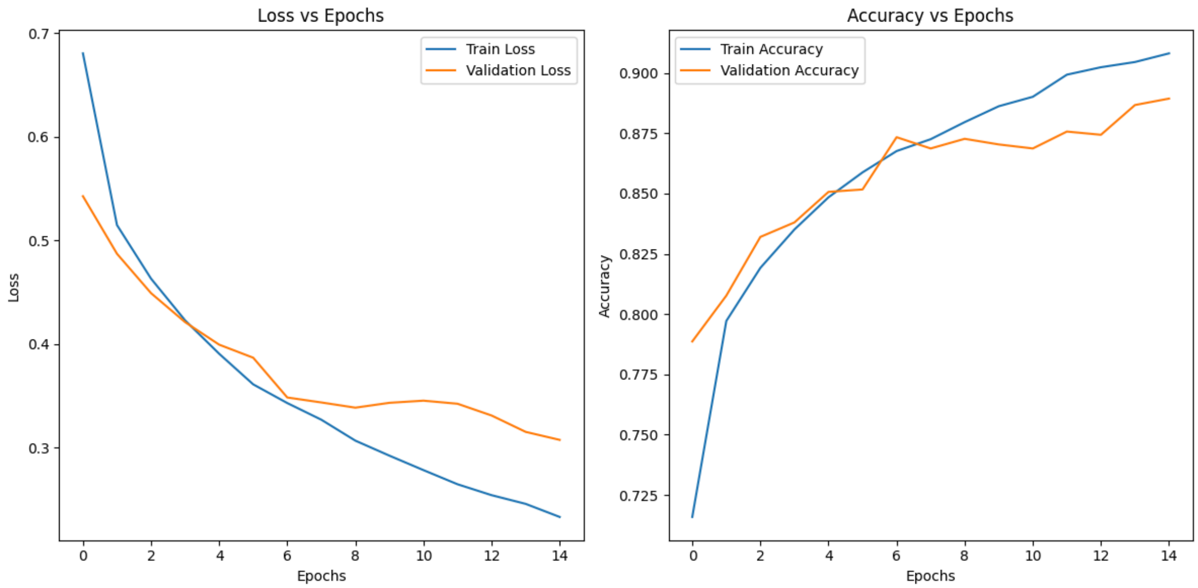
Validation dataset visualization



3. A CNN Model class is implemented, inheriting the methods from nn.Module class. It is initialized, and the forward method is created according to the specifications given.

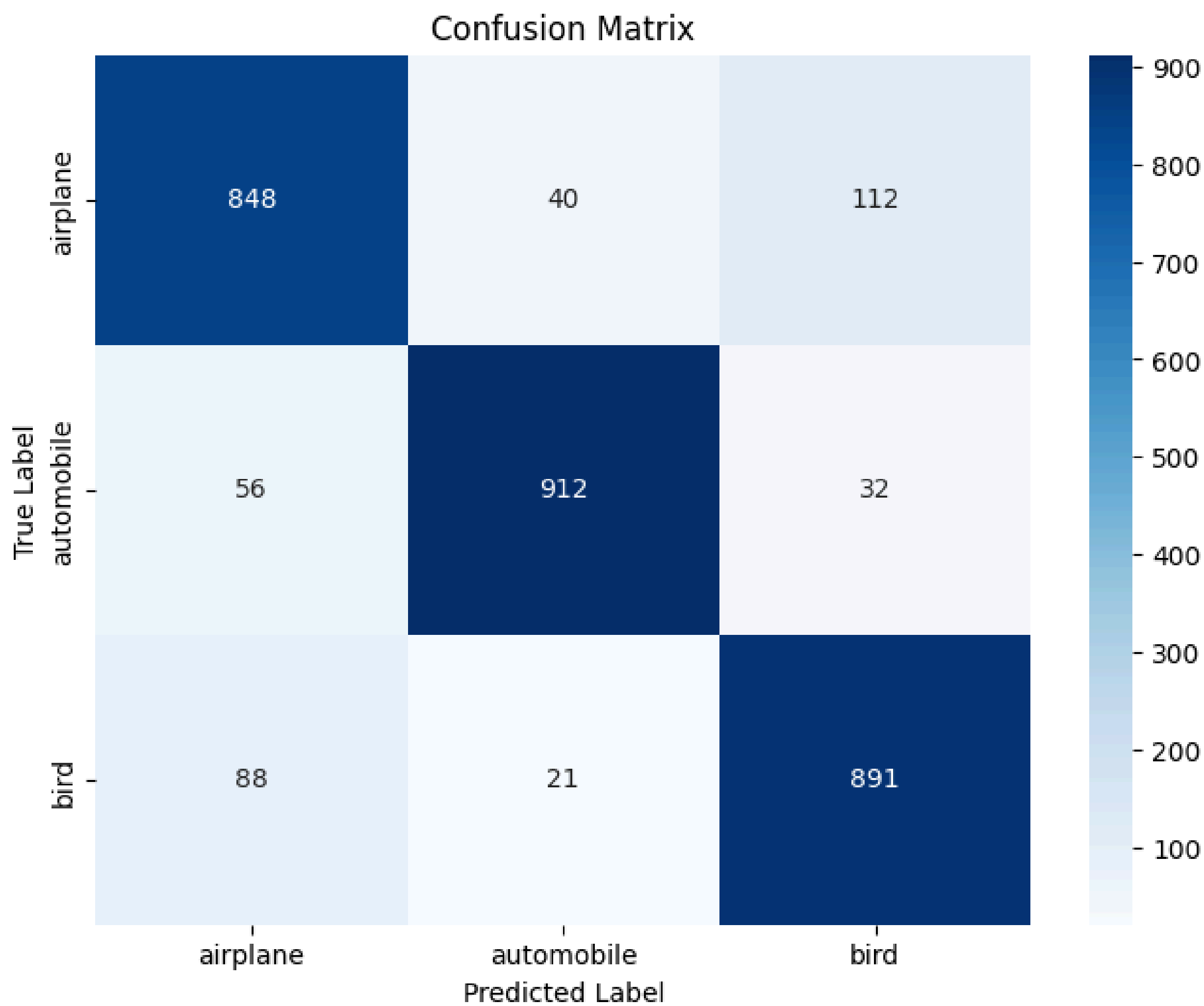
4. The model is trained for 15 epochs, and the model is saved as **cnn_model.pth**.

5. Loss and Accuracy vs Epochs Plots



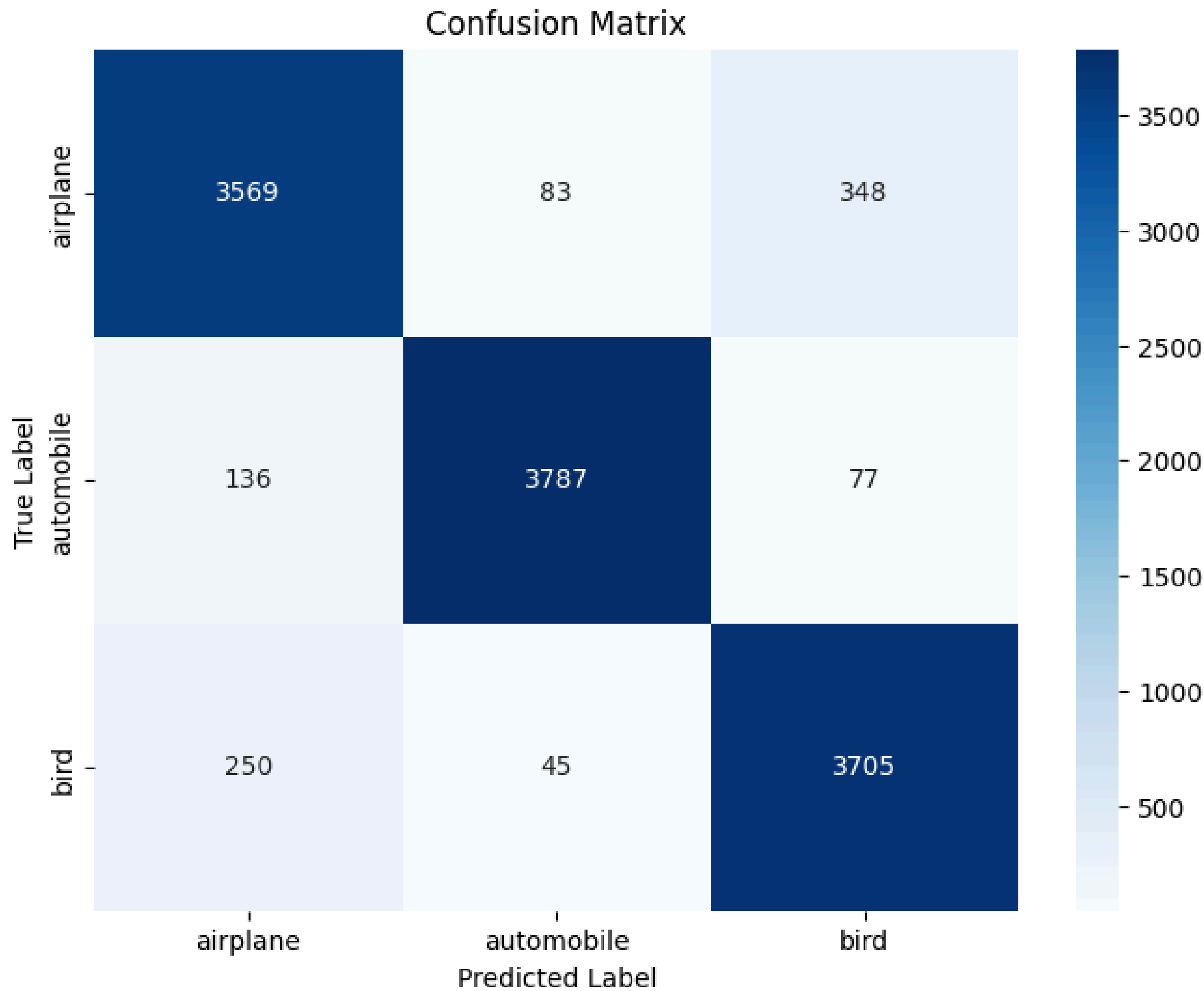
We see that the loss constantly decreases, and the accuracy increases for both the training and validation set. The validation set accuracy is slightly lower than the training set accuracy.

Test dataset:
Test Loss: 0.3079, Test Accuracy: 88.3667%, F1-Score: 0.8839

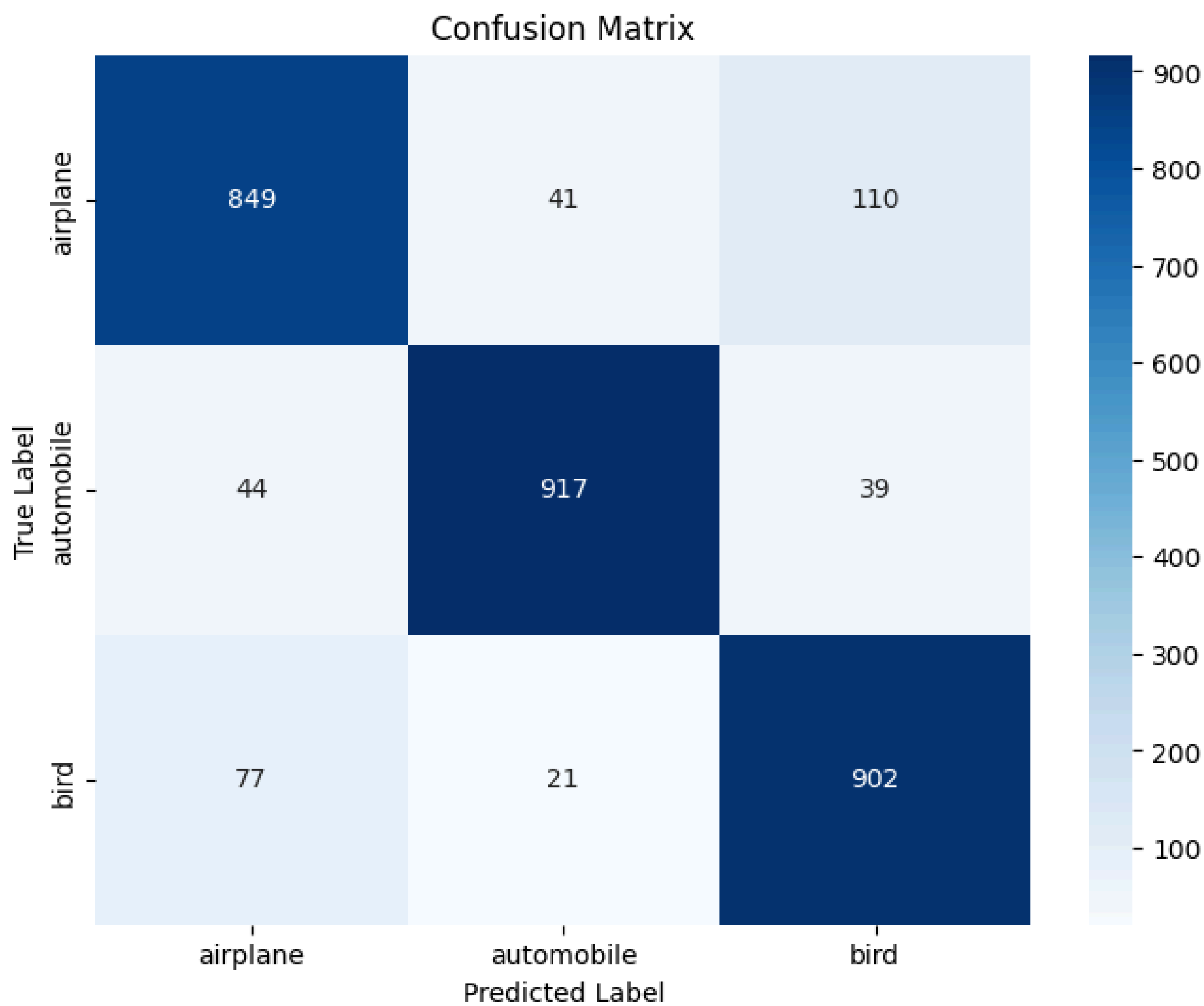


Train dataset

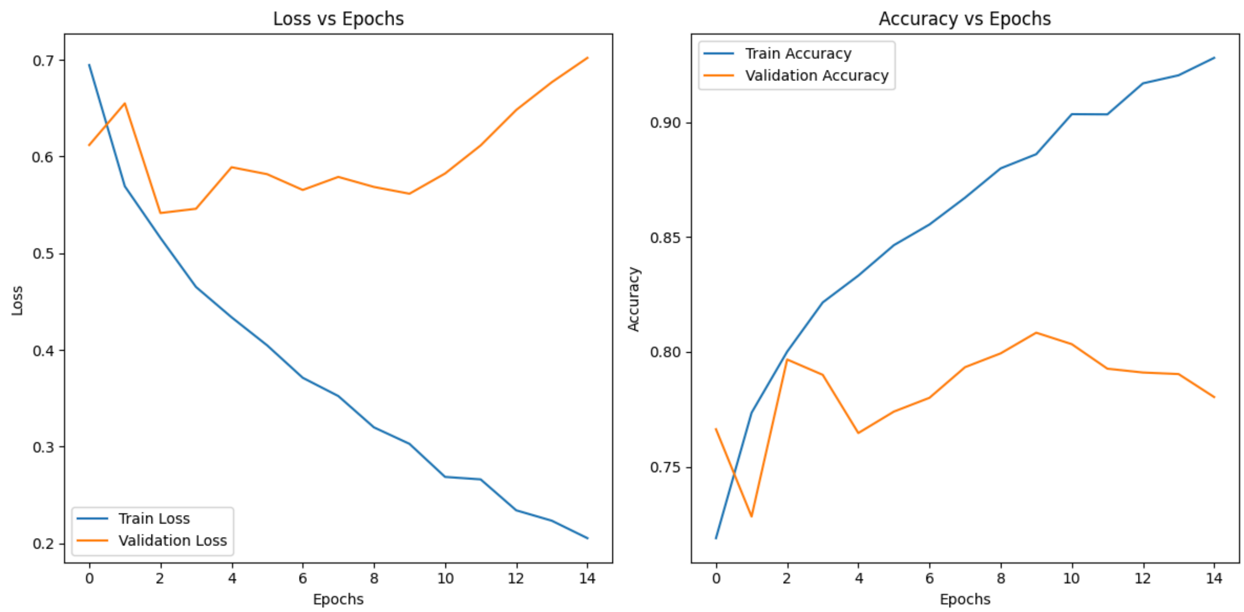
Train Loss: 0.2078, Train Accuracy: 92.1750%, F1-Score: 0.9219



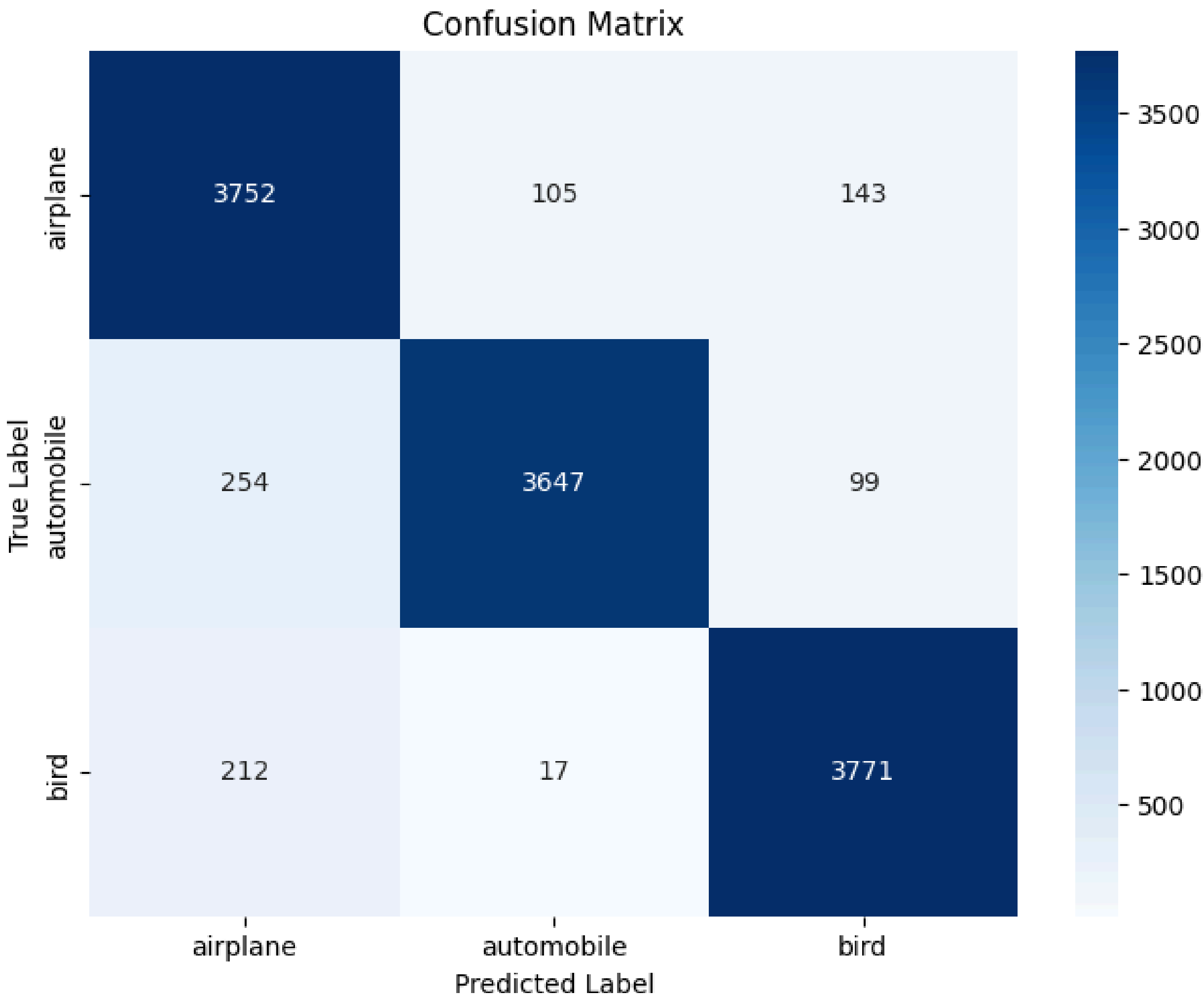
Validation dataset:
Validation Loss: 0.3071, Validation Accuracy: 88.9333%, F1-Score: 0.8894



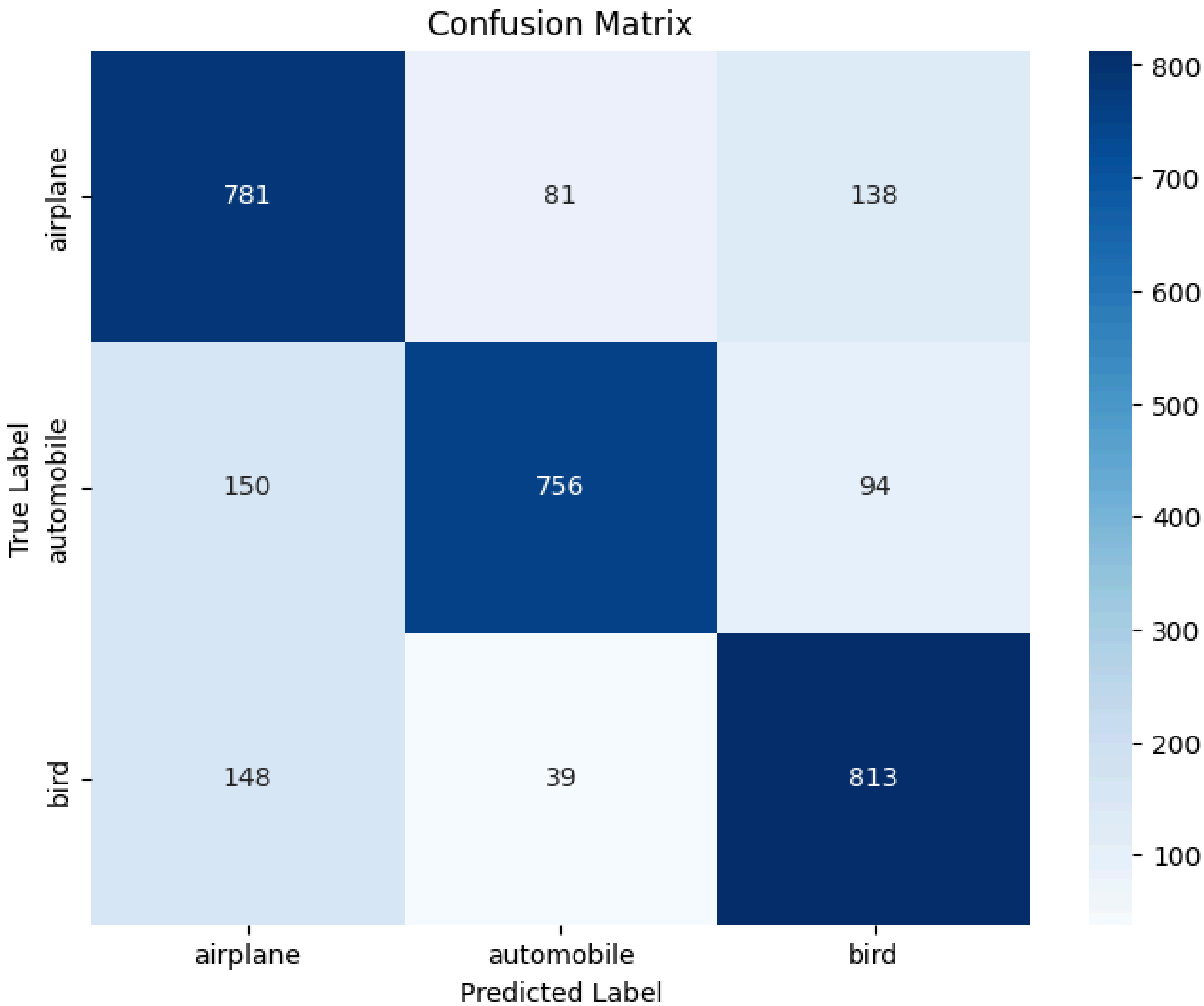
6. The MLP model is implemented according to the given specifications and saved as **mlp_model.pth** after training for 15 epochs**.**



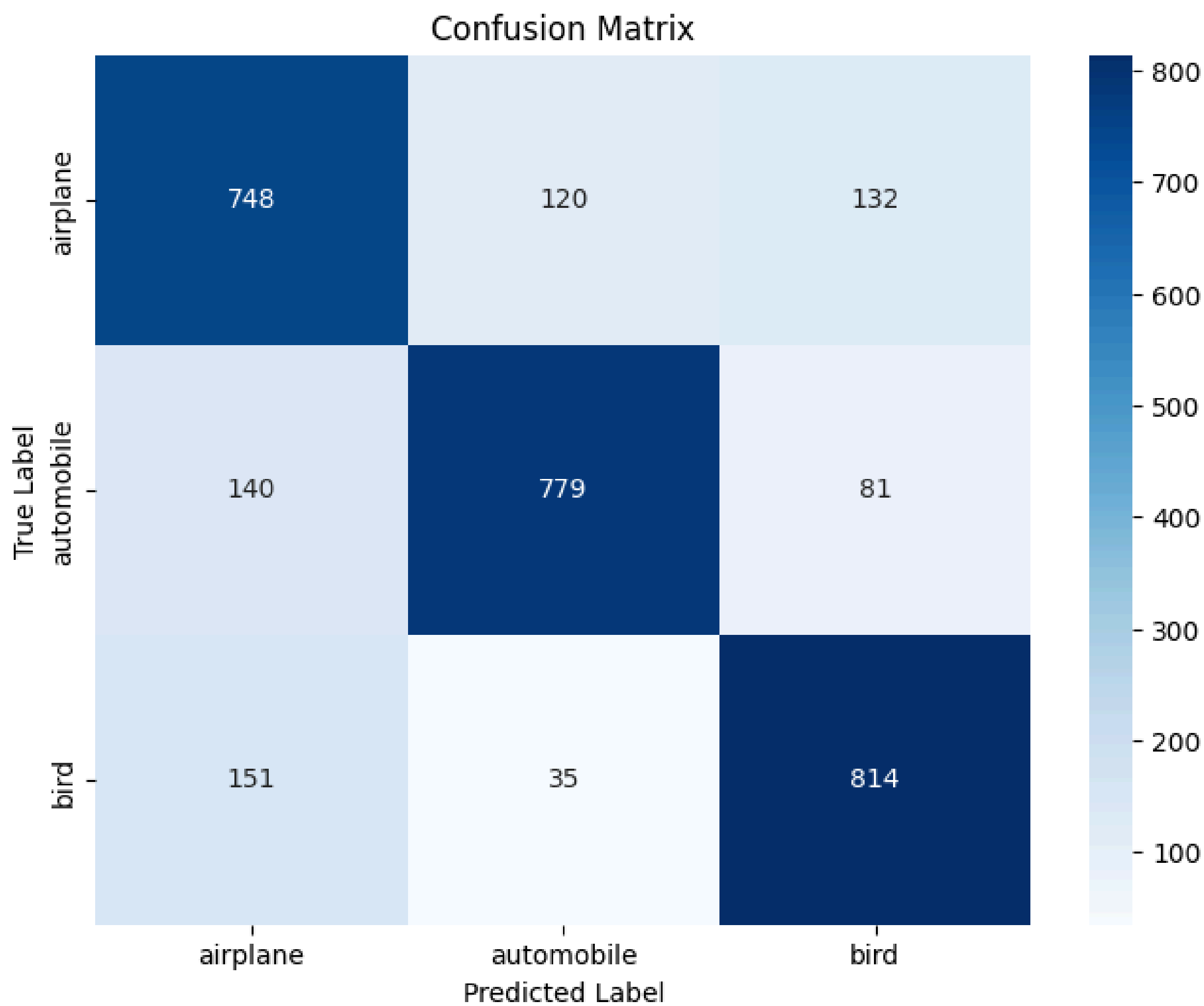
7. Training dataset:
   Training Loss: 0.2044, Training Accuracy: 93.0833%, F1-Score: 0.9311



Testing dataset:
Testing Loss: 0.6888, Testing Accuracy: 78.3333%, F1-Score: 0.7841



Validation set:
Validation Loss: 0.7024, Validation Accuracy: 78.0333%, F1-Score: 0.7808



We notice that the MLP model has higher loss and lower accuracy than the CNN model. The F1 score for the MLP model is also lower than the CNN model. This shows that the MLP model is unsuitable for tasks involving images and other similar data. Hence, it struggles to generalize the data. This could be due to the inability of the MLP to extract features efficiently compared to CNN, and hence, critical information is lost in the process.