SECTION A

Question 1

A. Output Feature Map Dimensions

To determine the dimensions of the feature map, we'll use the standard convolution output size formula when using stride = 1 and no padding:

Output Width = (Input Width - Kernel Width) + 1

Output Height = (Input Height - Kernel Height) + 1

Given:

- Input image dimensions: M × N
- Kernel size: K × K
- Stride: 1
- Padding: 0

Calculation:

- Output Width = (N - K) + 1
- Output Height = (M - K) + 1

B. Number of Elementary Operations per Output Pixel

To compute a single output pixel in the feature map, we need to:

1. Multiply each kernel element with its corresponding input image patch
2. Sum these multiplied values

Operations per output pixel:

- Multiplications: K × K × P (kernel width × kernel height × input channels)
    - We multiply each kernel element with the corresponding input channel patch
- Additions: (K × K × P) - 1
    - We need K × K × P multiplication results
    - We sum these results, which requires (K × K × P) - 1 addition operations

Total elementary operations per output pixel:

- Multiplications: $K^2 \times P$
- Additions: $(K^2 \times P) - 1$

C. Computational Complexity Analysis

To derive the time complexity, we'll count the operations for the entire forward pass:

Key Dimensions:

- Input image: $M \times N$ with P channels
- Number of kernels: Q
- Kernel size: $K \times K$
- Output feature map dimensions: $(M-K+1) \times (N-K+1)$

Computational Steps:

1. For each of the Q kernels, we'll compute a full feature map
2. Each feature map requires computing $(M-K+1) \times (N-K+1)$ output pixels
3. For each output pixel, we perform:
    - $K^2 \times P$ multiplications
    - $(K^2 \times P) - 1$ additions

Total Computational Complexity Calculation:

First, compute operations per output pixel for one kernel:

- Multiplications per pixel: $K^2 \times P$
- Additions per pixel: $K^2 \times P - 1$

Number of output pixels: $(M-K+1) \times (N-K+1)$

Total operations for one kernel:

- Multiplications: $Q \times (M-K+1) \times (N-K+1) \times K^2 \times P$
- Additions: $Q \times (M-K+1) \times (N-K+1) \times (K^2 \times P - 1)$

Big-O Notation Derivation:

1. **General Case Complexity**: Time Complexity = $O(Q \times M \times N \times K^2 \times P)$
2. **Simplified Case ($\min(M, N) \gg K$)**: When the image dimensions are much larger than the kernel size, we can approximate: Time Complexity = $O(Q \times M \times N \times K^2 \times P)$

Question 2

K-Means Algorithm: Assignment Step

The Assignment Step is fundamentally about classifying data points to their nearest cluster centroid. Let's dive deep into its mechanics:

- **Core Concept**
  - Imagine you have a collection of data points in a multidimensional space and K predefined cluster centers. The goal is to assign each point to the cluster with the closest center.
- **Mathematical Definition**
  - For each data point x_i, calculate the distance to each centroid μ_j, and assign the point to the cluster with the minimum distance. Typically, Euclidean distance is used:
  - Distance = $\sqrt{[\ \Sigma(x_{i,k} - \mu_{j,k})^2\ ]}$
  - Where:
  - $x_{i,k}$ is the k-th feature of data point i
  - $\mu_{j,k}$ is the k-th feature of centroid j

K-Means Algorithm: Update Step

The Update Step recalculates cluster centroids based on the current point assignments.

- **Core Mechanism**
  - For each cluster, compute the mean of all points currently assigned to that cluster. This new mean becomes the cluster's updated centroid.
- **Mathematical Formulation**
  - $\mu_j = (1/n_j) * \Sigma x_i$, where:
  - μ_j is the new centroid
  - n_j is the number of points in the cluster
  - Σx_i sums all points in the cluster
- **Iterative Nature**
  - The Assignment and Update steps are repeated alternately until:
  - Centroids stabilize (minimal movement)
  - A maximum number of iterations is reached
  - Points stop changing cluster assignments

Determining Optimal Number of Clusters: Elbow Method

The Elbow Method helps identify an appropriate number of clusters by plotting the variance explained against the number of clusters.

Process

1. Run K-Means for different K values (e.g., 1-10 clusters)
2. Calculate total within-cluster sum of squares (WCSS)
3. Plot WCSS against number of clusters
4. Look for the "elbow point" where the rate of variance reduction slows down

Global Minima Challenge

Randomness and Convergence

Randomly initializing centroids does NOT guarantee finding the global minimum. K-Means can:

- Converge to local optima
- Produce different results in different runs
- Miss the most representative cluster configuration

Mitigation Strategies

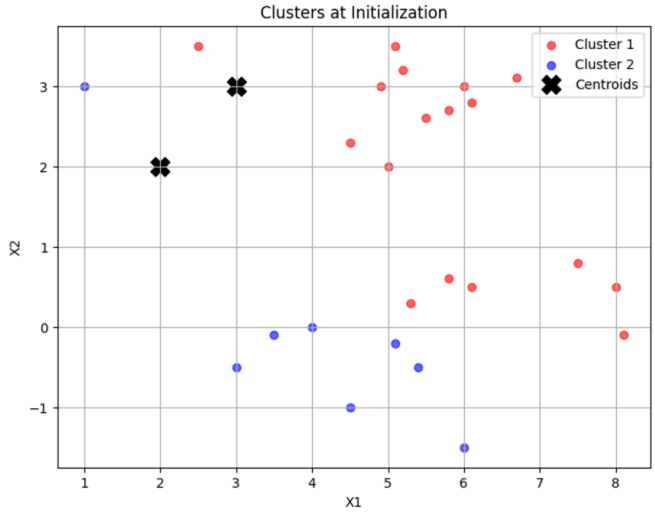
1. Multiple Random Initializations
    - Run K-Means several times
    - Choose the result with lowest total within-cluster variance
2. K-Means++ Initialization
    - Intelligently spread initial centroids
    - Reduces likelihood of poor initial configurations
3. Advanced Techniques
    - Hierarchical preprocessing
    - Specialized seeding algorithms
    - Meta-heuristic optimization approaches

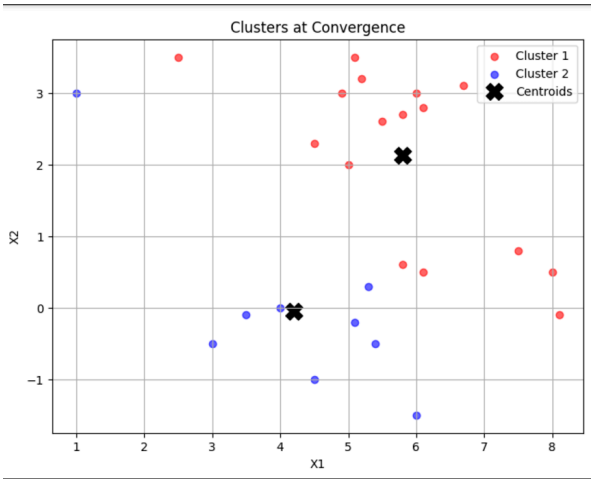
SECTION B

a. Below is the output for the part a.

- Converged in 3 iterations.
- Final centroids: [[5.8 2.125] [4.2 -0.05555556]]
- Cluster assignments: [0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 0]

B. Final Centroids: [[5.8 2.125] [4.2 -0.05555556]]



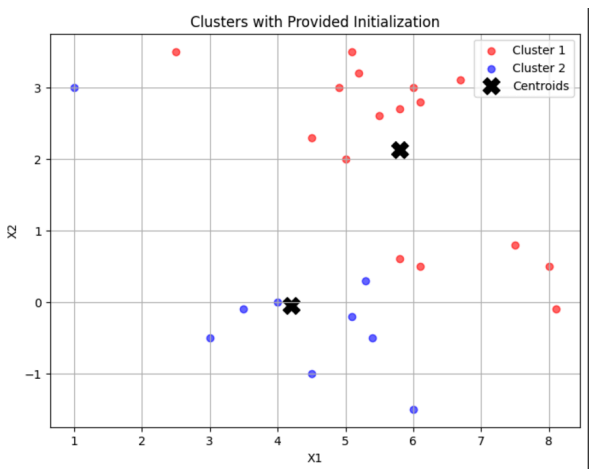


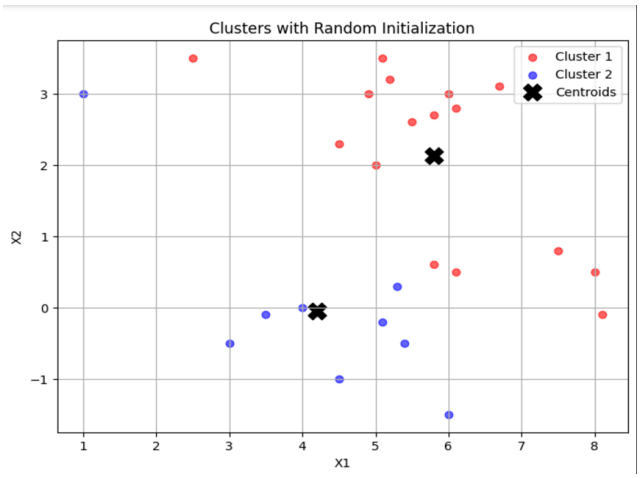
C. Randomly Chosen Initial Centroids: [[5.5 2.6] [3. -0.5]]

Converged in 3 iterations.

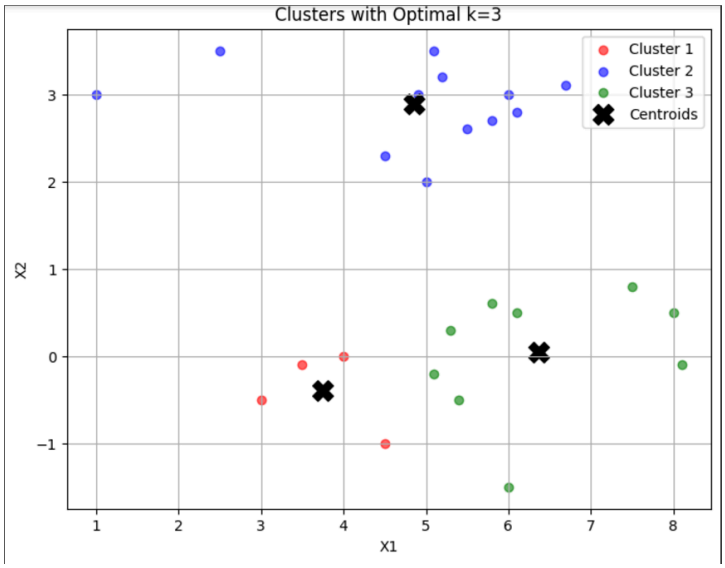
Final Centroids with Provided Initialization: [[5.8 2.125] [4.2 -0.05555556]]

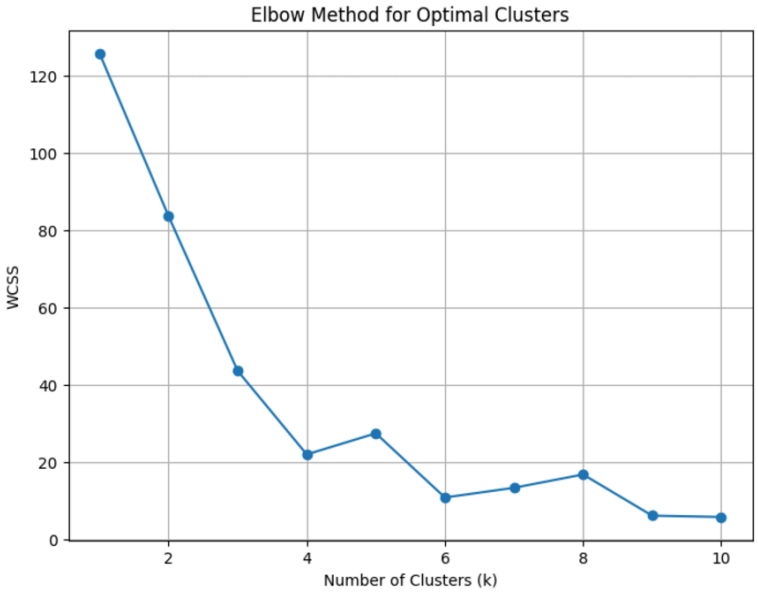
Final Centroids with Random Initialization: [[5.8 2.125] [4.2 -0.05555556]]





D.


Clusters with Optimal k=3
Cluster 1
Cluster 2
Cluster 3
Centroids
X1
X2


Elbow Method for Optimal Clusters
WCSS
Number of Clusters (k)

SECTION C

Confusion Matrix Analysis:

- **CNN:**
    1. The confusion matrix indicates a strong performance across all classes, especially for "truck," where the predictions are highly accurate.
    2. Misclassification between "dog" and "cat" is evident but less severe than in MLP.
    3. Better balance across the three classes.
- **MLP:**
    1. Significantly higher confusion between "dog" and "cat" compared to CNN.
    2. "Truck" predictions are reasonable but less accurate than CNN's.
    3. Poorer differentiation between classes, leading to reduced overall performance.

Observations and Insights:

- **CNN Outperforms MLP:**

1. The CNN consistently achieves better accuracy and F1 scores compared to the MLP across both validation and test datasets.
2. This is expected as CNNs are better suited for image data due to their ability to capture spatial and hierarchical features using convolutional layers.

- **Misclassification Trends:**

1. Both models struggle with "dog" vs. "cat" classification, likely due to similarities in features between these classes. However, CNN handles this misclassification better.

- **Model Complexity:**

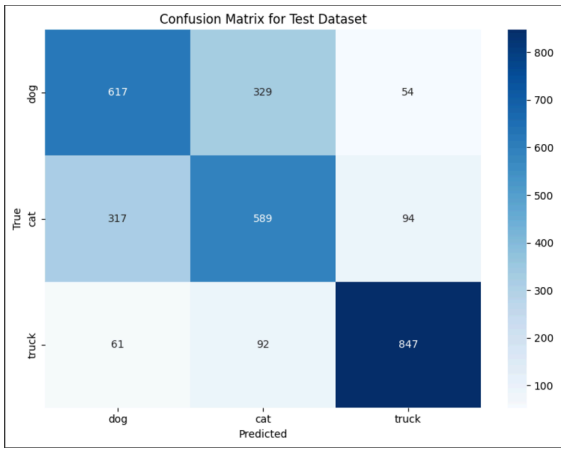
1. The CNN has a more complex architecture, allowing it to extract and learn richer features. This leads to its superior performance compared to the MLP, which uses fully connected layers and lacks the ability to process spatial relationships effectively.

Generalization:

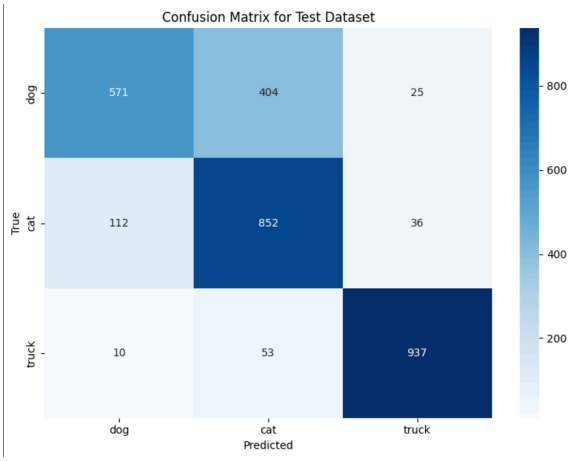
- The CNN exhibits good generalization with a minimal gap between validation and test accuracy.
- The MLP, while not as strong, may suffer from underfitting, indicating its architecture might not be complex enough for the given task.

Conclusion:

The CNN is the better-performing model for this classification problem, as evidenced by higher accuracy, F1 scores, and a better-balanced confusion matrix. It leverages its architecture to capture intricate patterns in image data, making it more effective at differentiating between classes. The MLP's performance is adequate but lacks the nuanced feature extraction required for this dataset.



(For mlp)



(for CNN)

For detailed results see the notebook