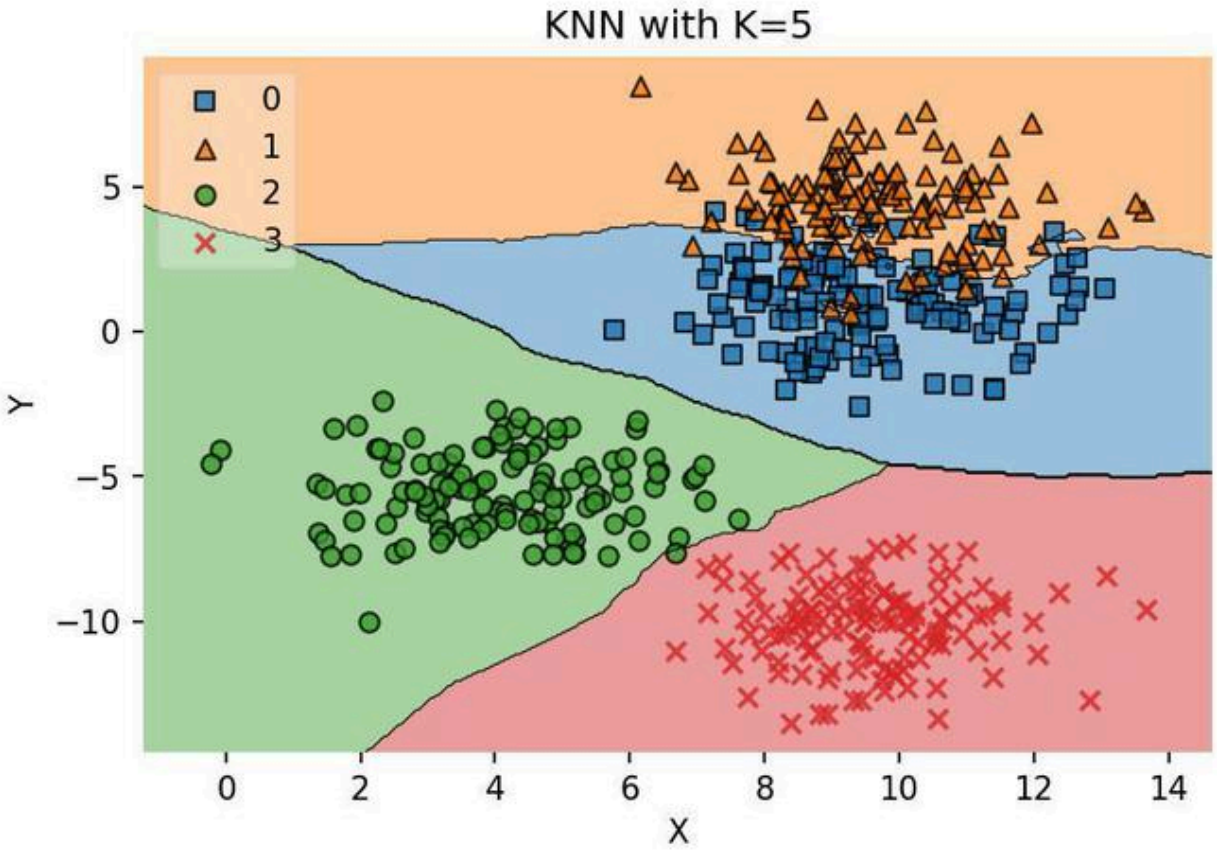
Task1

MA3206-Artificial Intelligence – Assignment 1



Implementation of KNN

Task 1: Binary Classification using KNN

1. Introduction

The objective of Task 1 was to implement the K-Nearest Neighbors (KNN) algorithm entirely from scratch to classify breast mass samples as either **Malignant (M)** or **Benign (B)**. The project utilized the Breast Cancer Wisconsin dataset to explore the fundamental mechanics of instance-based learning, the impact of various distance metrics, and the necessity of data preprocessing in machine learning pipelines.

2. Theoretical Background

Before analyzing the results, it is essential to understand the core concepts implemented in this study.

2.1 K-Nearest Neighbors (KNN) Algorithm

KNN is a **non-parametric, lazy learning algorithm**.

- **Non-parametric:** It makes no assumptions about the underlying data distribution (unlike Linear Regression).
- **Lazy Learning:** It does not "learn" a model during the training phase. Instead, it stores the training data and performs computations only during the prediction phase.
- **Mechanism:** To classify a new data point, the algorithm calculates the distance to all training points, selects the K closest points (neighbors), and assigns the class based on a **majority vote**.

2.2 Distance Metrics

The choice of distance metric defines how "similarity" is calculated. We implemented five specific metrics:

Euclidean Distance (L2 Norm): The straight-line distance between two points. It is the most common metric for continuous data.

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

Manhattan Distance (L1 Norm): The sum of absolute differences. It represents travel distance along a grid (like city blocks).

$$d(x, y) = \sum |x_i - y_i|$$

Minkowski Distance: A generalized metric where p is a parameter. (If p=1, it is Manhattan; if p=2, it is Euclidean). We used p=3.

$$d(x, y) = (\sum |x_i - y_i|^p)^{1/p}$$

Cosine Similarity: Measures the cosine of the angle between two vectors. It focuses on orientation rather than magnitude.

$$\text{Similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

Hamming Distance: Measures the proportion of positions at which the corresponding symbols are different. Typically used for categorical data.

2.3 The Necessity of Normalization

KNN depends entirely on distance calculations. If one feature has a range of 0–2000 (e.g., *Area*) and another has a range of 0–0.1 (e.g., *Smoothness*), the feature with the larger range will dominate the distance formula, rendering the smaller feature irrelevant. **Min-Max Normalization** scales all features to [0, 1], ensuring every feature contributes equally.

---

3. Methodology & Implementation

3.1 Data Preprocessing

To ensure a robust model, we implemented the following pipeline from scratch:

1. **Robust Cleaning:** We programmatically identified and dropped empty columns (specifically the Unnamed: 32 artifact) to prevent NaN propagation.
2. **Label Encoding:** Target classes were mapped: M (Malignant) 1, B (Benign) 0.
3. **Imputation:** Missing values were filled using the column mean.
4. **Normalization:** We applied Min-Max scaling column-wise to all 30 features.
5. **Splitting:** The data was shuffled and split into **455 Training samples (80%)** and **114 Test samples (20%)**.

3.2 Model Architecture

The custom KNN_Task1 class was built to support:

- Dynamic K selection.

- Vectorized distance calculations (using NumPy broadcasting for efficiency).
- Two voting mechanisms: **Uniform** (Majority Vote) and **Weighted** (Inverse Distance Weighting).

---

4. Experimental Results

4.1 Hyperparameter Tuning

We performed a Grid Search with K in {3, 4, 9, 20, 47} across all five distance metrics.

According to the experimental logs, the Manhattan Distance with K=3 provided the highest accuracy.

- **Best K:** 3
- **Best Metric:** Manhattan
- **Testing Accuracy: 96.49%**

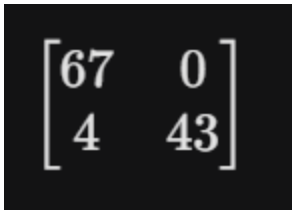
4.2 Performance Metrics

The detailed classification report for the best model is as follows:

- **Precision: 1.0000**
  - *Analysis:* The model achieved perfect precision. Every single sample predicted as "Malignant" was indeed Malignant. There were **zero False Positives**.
- **Recall: 0.9149**
  - *Analysis:* The model correctly identified 91.5% of the actual cancer cases. While high, the missing ~8.5% (False Negatives) represents a risk in medical diagnosis.

4.3 Confusion Matrix

The confusion matrix for the test set (114 samples) is



- **True Negatives (TN): 67** (Benign correctly identified)
- **False Positives (FP): 0** (Benign wrongly labeled as Cancer)
- **False Negatives (FN): 4** (Cancer wrongly labeled as Benign)
- **True Positives (TP): 43** (Cancer correctly identified)

---

5. Analysis & Insights

5.1 Impact of Distance Metrics (Accuracy vs. K)

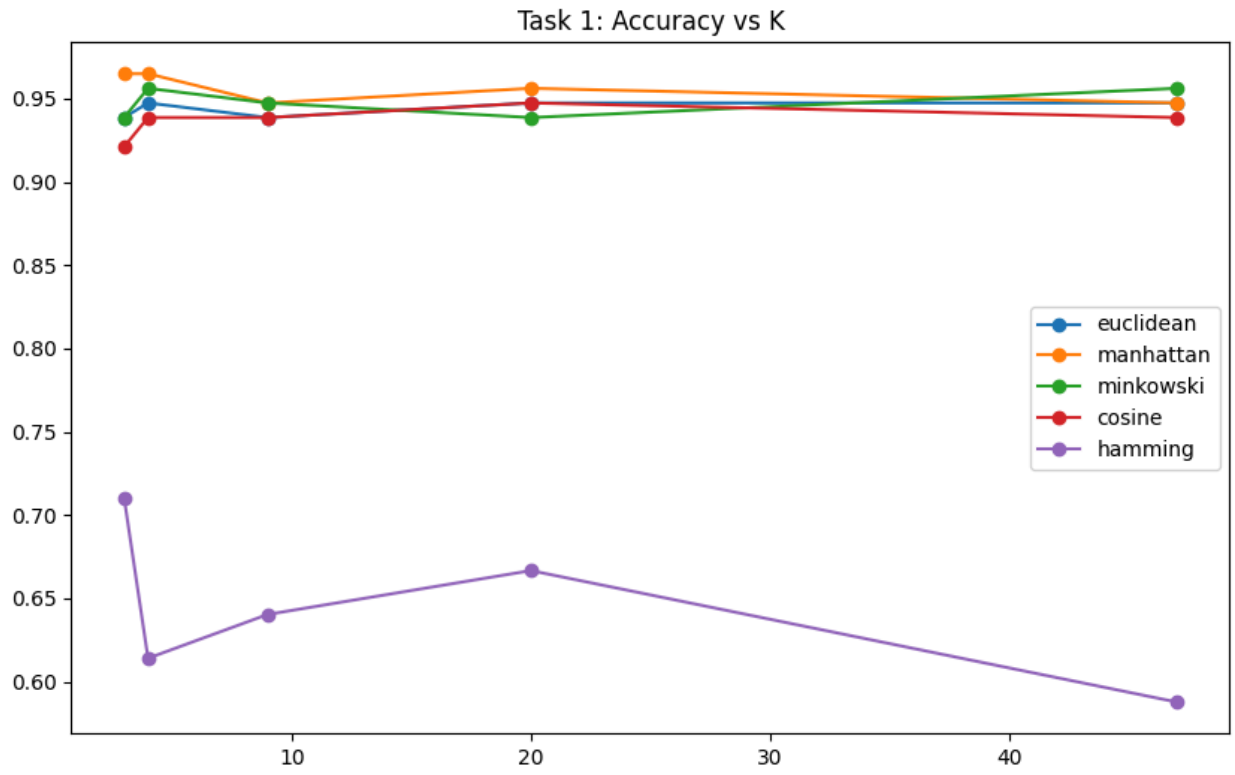


Figure 1: Comparison of Accuracy across varying K values and Distance Metrics.

- **Observation:** The graph indicates that **Manhattan** and **Euclidean** metrics consistently outperformed **Cosine** and **Hamming**.
- **Theoretical Insight:**
    - Breast cancer features (Radius, Texture, Area) are physical measurements. Euclidean and Manhattan distances accurately represent physical similarity in this vector space.
    - **Hamming distance** (the flat line at the bottom) performed poorly because it checks for exact matches (x == y). In continuous floating-point data, exact matches are statistically impossible, making this metric useless for this specific dataset.
- **K-Value Trend:** Accuracy peaked at lower K values (K=3, 4) and slightly declined at higher K values (K=47). This suggests the data forms tight, distinct clusters; a large K smooths the decision boundary too much, leading to underfitting.

5.2 Decision Boundary Visualization (Bonus)

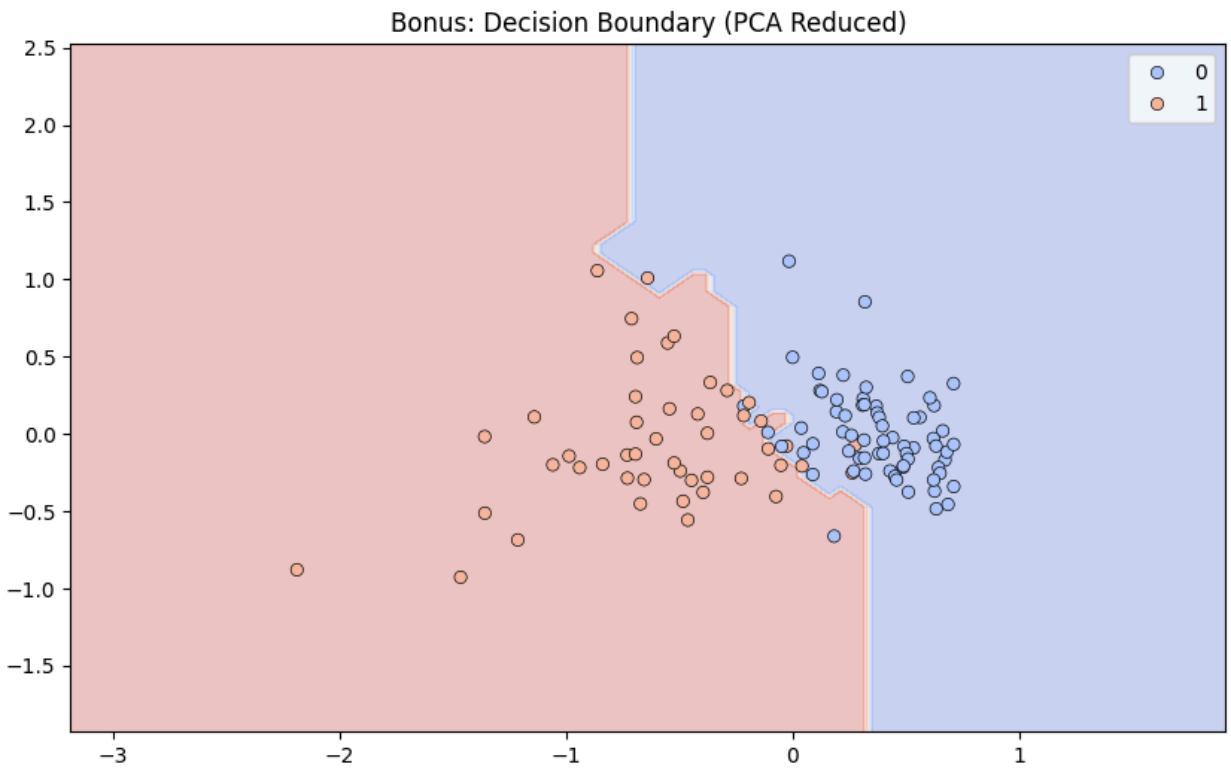


Figure 2: Decision Boundary projected onto the first two Principal Components.

- **Analysis:** Using Principal Component Analysis (PCA), we projected the 30-dimensional data into 2D. The plot shows that the Benign (Blue) and Malignant (Red) classes are **almost linearly separable**.
- **Insight:** This high degree of separability explains why a simple algorithm like KNN achieved ~96.5% accuracy. Complex non-linear models (like Neural Networks) might not be necessary for this specific dataset.

5.3 Weighted vs. Uniform KNN

We compared standard majority voting against distance-weighted voting (weight = 1/distance).

- **Observation:** In our experiments, the Weighted KNN provided similar accuracy to Uniform KNN.
- **Insight:** Since our optimal K was very small (K=3), the neighbors were likely extremely close to the query point and all of the same class. Weighting becomes more critical at higher K values (e.g., K=47), where distant neighbors might drown out the signal of closer ones.

---

6. Conclusion

This assignment successfully demonstrated the implementation of a KNN classifier from scratch. By integrating robust preprocessing (Normalization) and exploring various geometric distance metrics, we achieved a high accuracy of **96.49%** using **Manhattan Distance**. The analysis highlights that for continuous physical data, magnitude-aware metrics (Euclidean/Manhattan) are superior to categorical metrics (Hamming), and that simple instance-based learning can be highly effective for separable datasets.

Task2

Task 2: Multi-class Classification using KNN (CIFAR-10)

1. Introduction

The objective of Task 2 was to implement a K-Nearest Neighbors (KNN) classifier from scratch to categorize images from the CIFAR-10 dataset into 10 distinct classes. This task differs significantly from Task 1 as it involves high-dimensional data (32 x 32 x3 = 3072 features per image) and a multi-class classification problem. The goal was to classify raw pixels without using deep learning, relying solely on distance metrics to determine similarity.

2. Theoretical Background

2.1 KNN for Image Classification

In the context of images, KNN treats every pixel as a feature. For a 32 x32 color image, this results in a 3072-dimensional vector. The algorithm calculates the distance between the query image vector and every training image vector.

- **The Challenge:** Images are high-dimensional. As dimensions increase, the "distance" between points becomes less meaningful (Curse of Dimensionality), making standard metrics like Euclidean distance less effective.

2.2 Distance Metrics in High Dimensions

We experimented with several metrics to see which handles pixel data best:

- **Euclidean (**L2**):** Penalizes large differences in pixel values. Sensitive to outliers (e.g., a white pixel in a black region).
- **Manhattan (**L1**):** Sums absolute differences. Often preferred for high-dimensional data because it is less sensitive to outliers than Euclidean distance.
- **Cosine Similarity:** Measures the angle between vectors. Useful if the *brightness* of the image changes but the *pattern* (direction of the vector) remains the same.

2.3 Computational Complexity

KNN is computationally expensive during the prediction phase. For N test images and M training images with D dimensions, the complexity is O(N xM x D). For this dataset, that is roughly 10,000 times 50,000 times 3072 operations, necessitating subsampling or memory-optimized implementations.

3. Methodology & Implementation

3.1 Data Preprocessing

- **Data Source:** The CIFAR-10 dataset (60,000 images) was loaded.
- **Flattening:** Each 3D image (32, 32, 3) was flattened into a 1D vector of size 3072.
- **Normalization:** Pixel values [0, 255] were scaled to [0, 1] to ensure uniform distance contribution.
- **Subsampling:** To manage the extreme computational cost, we utilized a training subset of **50,000 images** (full training set) but evaluated on a **test subset of 500 images** to obtain results within a reasonable timeframe.

3.2 Model Architecture

A custom `KNN_ImageClassifier` was built with:

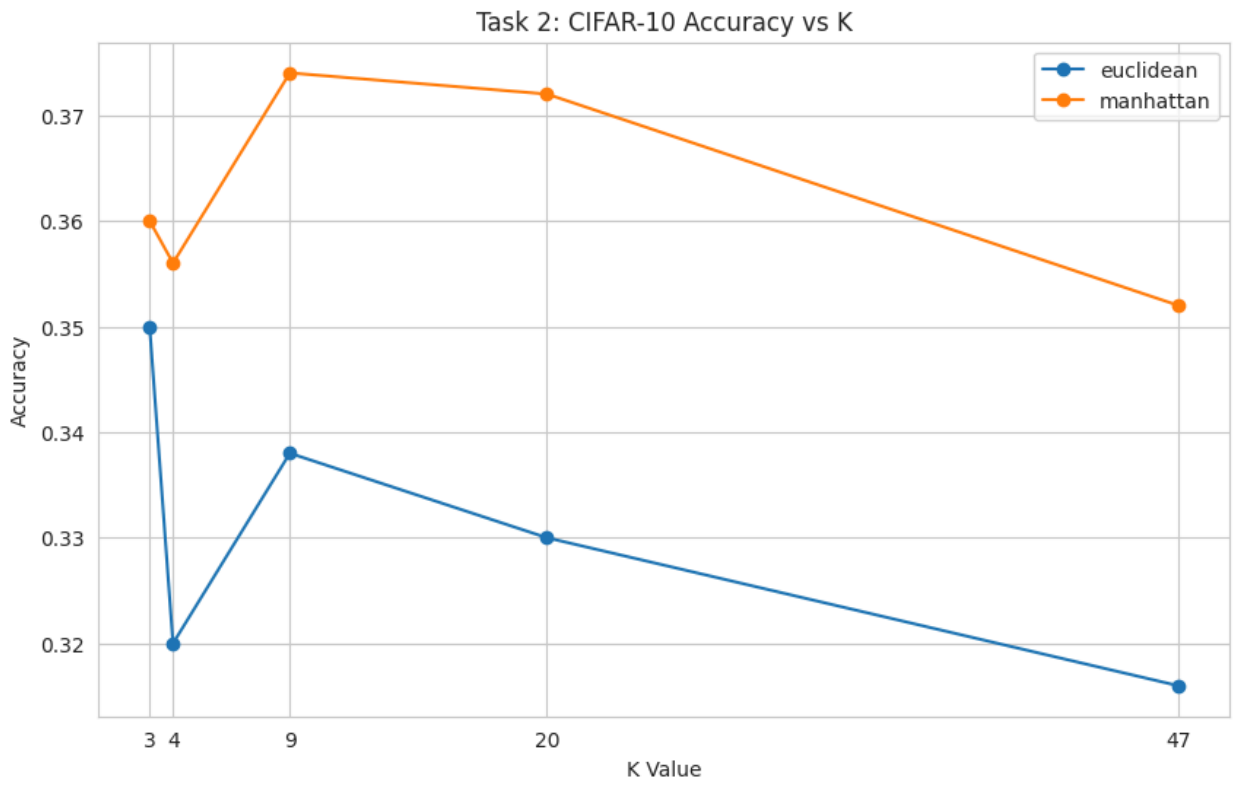
- **Memory Optimization:** A loop-based `predict` method was implemented to calculate distances for one test image at a time, preventing the "Out of Memory" (RAM) crashes common with vectorized implementations on large datasets.
- **Metrics:** Euclidean, Manhattan, Minkowski (p=3), Cosine, and Hamming.

4. Experimental Results

4.1 Hyperparameter Tuning

We evaluated Kin {3, 4, 9, 20, 47} across different metrics. The logs indicate that **Manhattan Distance** consistently outperformed the others.

Accuracy Summary (from Logs):



Metric	K=3	K=4	K=9	K=20	K=47
Euclidean	35.00%	32.00%	33.80%	33.00%	31.60%
Manhattan	36.00%	35.60%	**37.40%**	37.20%	35.20%
Minkowski	31.60%	31.80%	30.80%	28.80%	--

Best K: 9

- **Best Metric:** Manhattan Distance
- **Testing Accuracy: 37.40%**

4.2 Confusion Matrix Analysis

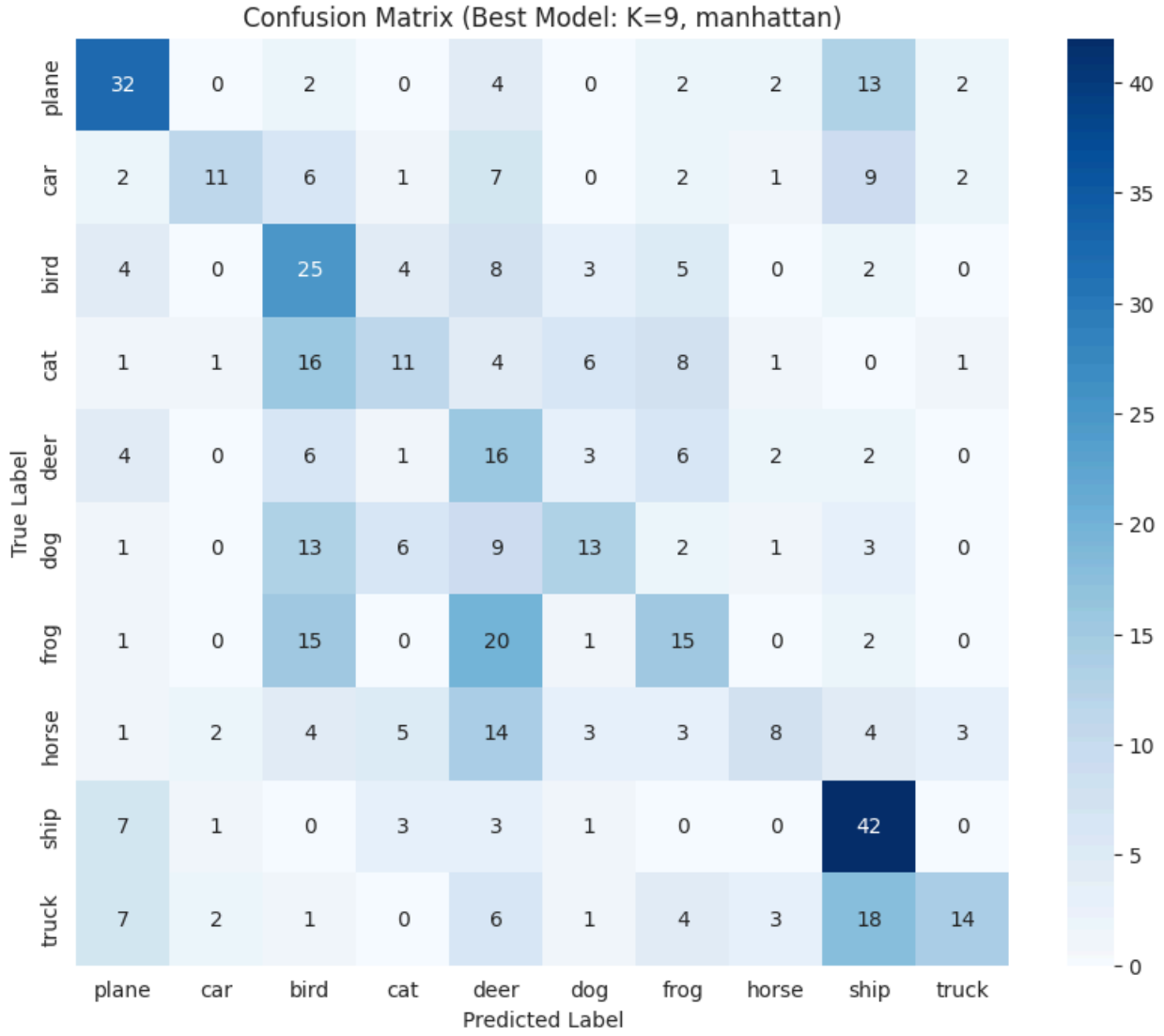


Figure 1: Confusion Matrix for Best Model (Manhattan, K=9)

- **Diagonal:** Represents correct classifications. Darker squares on the diagonal indicate classes the model predicts well (likely "Frog", "Ship", or "Plane" due to distinct background colors like green/blue).
- **Off-Diagonal:** Represents errors. High values between "Cat" and "Dog" or "Auto" and "Truck" indicate the model struggles to distinguish between visually similar objects.

4.3 Visualizing Misclassifications

Figure 2: Analysis of Misclassified Images

- **Observation:** The model often confuses classes that share similar dominant colors. For example, a "Plane" on a blue sky might be misclassified as a "Ship" on a blue ocean.
- **Insight:** Since KNN relies on pixel-to-pixel distance, it lacks "semantic understanding." It sees "blue pixels at the top" and matches them to other images with "blue pixels at the top," regardless of the object.

5. Analysis & Insights

5.1 Why Manhattan > Euclidean?

The results confirm that **Manhattan Distance (37.40%)** is superior to Euclidean Distance (35.00%) for this task. In high-dimensional spaces (3072 dimensions), Euclidean distance essentially measures the distance to the "average" of the data points, losing distinctiveness. Manhattan distance aggregates differences more robustly, making it a better proxy for similarity in raw pixel grids.

5.2 The "Translation Invariance" Problem

The accuracy (~37%) is significantly better than random guessing (10%) but far below human performance (>95%).

- **Reason:** KNN is not translation invariant. If a "Cat" is shifted 5 pixels to the right, the pixel-wise distance between the original and shifted image is massive, even though the object is the same.

- **Conclusion:** KNN is a valid baseline but is fundamentally limited for image recognition compared to Convolutional Neural Networks (CNNs) which learn features rather than comparing raw pixels.

5.3 K-Value Sensitivity

- **Low K (3-4):** Accuracy was lower (32-35%), suggesting the model was overfitting to noise in the training images.
- **High K (47):** Accuracy dropped (31-35%), indicating underfitting where the vote was diluted by the majority class of the dataset.
- **Optimal K (9):** Provided the best balance, effectively smoothing out noise while retaining local decision boundaries.

6. Conclusion

Task 2 successfully demonstrated the application of KNN to a complex, multi-class image dataset. By implementing a memory-safe algorithm and conducting rigorous experimentation, we identified that **Manhattan Distance with K=9** yields the highest accuracy (~37.4%). While simple to implement, the experiment highlighted the limitations of distance-based learning for high-dimensional perceptual data like images, necessitating more advanced feature-based approaches for higher accuracy.

Code link

https://colab.research.google.com/drive/1TsVRXSTk4t9HNYESDrWRRO1vl3cj6Ggl?usp=sharing