Machine Learning Algorithms for Image Classification using CIFAR-10 Dataset Neural Networks - Deep Learning Course

Computer Science Department,Aristotle University of Thessaloniki

I. Introduction

This project addresses the foundational problem of image classification using one of the most widely adopted benchmark datasets, CIFAR-10. Consisting of 60,000 tiny 32x32 color images across 10 distinct object classes, CIFAR-10 serves as an ideal testbed for evaluating the performance of simple, yet conceptually powerful, classification algorithms.

This essay introduces the methodology and rationale behind implementing two specific non-parametric, distance-based classifiers: the k-Nearest Neighbors (kNN) algorithm and the Nearest Centroid Classifier (NCC). Our primary goal is to establish essential performance baselines for the CIFAR-10 challenge by utilizing raw pixel intensity values as the primary features.

The K-Nearest Neighbors (KNN) algorithm operates on the principle of instance-based learning, classifying a new data point based on the majority class of its k closest neighbors in the training set. Its simplicity is a core strength, as it makes no assumptions about the underlying data distribution, relying solely on a chosen distance metric (Euclidean) to determine similarity. Specifically, we will investigate the classification performance using $k = 1$ (1-Nearest Neighbor) and $k = 3$ neighbors to analyze the impact of minimal neighbor inclusion on accuracy.

In contrast, the Nearest Centroid Classifier (NCC), is a prototype-based method. Instead of storing all training instances, NCC first computes a single representative vector, or centroid, for each class by calculating the mean of all training samples belonging to that class. A new data point is then classified by assigning it the label of the nearest class centroid. This approach is highly efficient during prediction, though its reliance on a single mean vector can make it sensitive to noise or variations within classes.

By implementing and evaluating both KNN and NCC on the flattened pixel data of CIFAR-10, this project seeks to understand their inherent strengths and weaknesses in a high-dimensional feature space. The comparison will highlight the trade-offs between instance-based fidelity and prototype-based efficiency, laying the groundwork for more complex deep learning models typically applied to this dataset.

II. The CIFAR-10 Dataset and Vector Preparation

The CIFAR-10 dataset is a widely adopted benchmark in computer vision, consisting of $60,000$ color images, each of size 32×32 pixels. These images are equally distributed

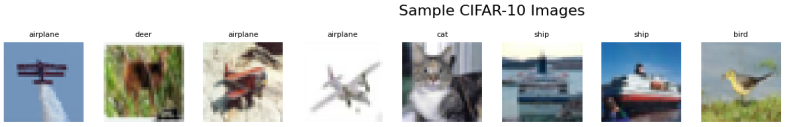

Fig. 1. Training samples from CIFAR-10

across 10 distinct object classes (e.g., airplane, automobile, bird, cat, dog), with $6,000$ images per class. The dataset is conventionally split into two subsets:

Training Set: $50,000$ images (N_{train}) used to learn the structure of the data (or, in the case of kNN, to simply store the instances).

Test Set: $10,000$ images (N_{test}) used for final evaluation.

A key challenge of CIFAR-10 is the low resolution (32×32), which forces classifiers to rely on fine details, and the high intra-class variance (e.g., different angles and lighting for the same object), making generalization difficult for shallow models. Vector Preparation (Flattening and Normalization)

Distance-based algorithms like Nearest Centroid (NCC) and kNN require a single, continuous feature vector as input. Therefore, the three-dimensional image data must be transformed:

Flattening (Feature Extraction): Each 32×32 color image has three color channels (Red, Green, Blue). To convert the image into a single feature vector $\mathbf{x}$, the dimensions are multiplied:

$$D = 32 \times 32 \times 3 = 3072 \text{ features}$$

Every image is thus represented by a 3072-dimensional vector. This process is necessary to compute the Euclidean distance, which operates on one-dimensional arrays (vectors).

Normalization: The raw pixel intensity values range from 0 to 255. To prevent features with inherently larger magnitude (such as those in the blue channel, if not properly scaled) from disproportionately influencing the distance metric, the data must be normalized. This is achieved by converting the data type to float and dividing all pixel values by 255.0, resulting in feature vectors where all components lie in the normalized range $[0.0, 1.0]$. This step is crucial for ensuring that the chosen distance metric accurately reflects the perceptual similarity between images.

III. K-Nearest Neighbors Classifier Implementation Examining K=1 and K=3

For the kNN classifier, the "training" phase is effectively trivial. Following the vector preparation—where the $50,000$ training images of the CIFAR-10 dataset are transformed into 3072-dimensional feature vectors $\mathbf{X}_{\text{train}}$—the "training" involves storing these vectors and their corresponding labels ($\mathbf{y}_{\text{train}}$). The model's computational burden is entirely deferred to the prediction phase.

The Prediction Phase: Distance Calculation

The core of the KNN algorithm lies in the prediction step, which is computationally expensive for large datasets like CIFAR-10. For every single test image $\mathbf{x}_{\text{test}}$ in the $10,000$-image test set, the classifier must perform the following three sub-steps:

Distance Metric: The Euclidean distance (L_2 norm) is chosen to measure the similarity between the test vector $\mathbf{x}_{\text{test}}$ and every training vector $\mathbf{x}_i \in \mathbf{X}_{\text{train}}$. The distance $d(\mathbf{x}_{\text{test}}, \mathbf{x}_i)$ is calculated across all $D = 3072$ dimensions:

$$d(\mathbf{x}_{\text{test}}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{3072} (\mathbf{x}_{\text{test}}^{(j)} - \mathbf{x}_i^{(j)})^2}$$

Neighbor Identification: Once the distance to all $50,000$ training samples is computed, the algorithm sorts these distances and identifies the k training samples that possess the smallest distances, which are the k-nearest neighbors.

Majority Voting: The predicted class label $\hat{y}$ for $\mathbf{x}_{\text{test}}$ is determined by a simple majority vote among the k neighbors' labels.

Comparative Analysis:k=1 and k=3

k=1:The Nearest Neighbor Classifier

Using k=1 simplifies the majority vote: he test sample is assigned the class label of its single closest neighbor.This value of the hyperparameter k achieved accuracy score: **29%**.

k=3:Increased size of Neighborhood to k=3

It means a test sample must pass the "filter" of three votes among neighbors to decide for a class label leading to **24%** accuracy score.

Note:The hyperparameter k is typically chosen as an odd integer to prevent ties during the majority voting step.

Due to memory limitations we trained and tested the model with only 10% of the training/test samples,equally among classes.

IV. Nearest Centroid Classifier Implementation

The Nearest Centroid Classifier (NCC) represents the most fundamental approach to distance-based classification, relying on computational simplicity and efficiency. While the K-Nearest Neighbors (KNN) algorithm uses all training samples for classification, NCC condenses the entire training set into a small number of prototypes, one for each class. This approach offers significant advantages in terms of memory and inference speed, crucial when dealing with a large dataset like CIFAR-10. The training phase for the NCC is direct and non-iterative. Given the set of training feature vectors $\mathbf{X}_{\text{train}}$, which consists of $N_{\text{train}} = 50,000$ samples, and their corresponding labels $\mathbf{y}_{\text{train}}$, the goal is to compute the mean feature vector, or centroid ($\boldsymbol{\mu}_c$), for each of the $C = 10$ classes.

For a given class $c \in \{0, 1, \ldots, 9\}$, the centroid is calculated as the arithmetic mean of all samples belonging to that class:

$$\boldsymbol{\mu}_c = \frac{1}{|c|} \sum_{\mathbf{x}_i \in \text{Class } c} \mathbf{x}_i$$

where $|\text{Class } c|$ is the number of training samples belonging to class c. Since the feature vectors $\mathbf{x}_i$ are the $D = 3072$-dimensional flattened and normalized pixel values, the resulting centroid $\boldsymbol{\mu}_c$ is also a vector of 3072 dimensions. The output of the training phase is a set of only 10 centroid vectors, $\mathcal{M} = \{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_9\}$, which collectively represent the learned model.

Prediction Phase: Distance Measurement

During the prediction phase, a test sample $\mathbf{x}_{\text{test}}$ is classified by determining which of the 10 stored centroids is closest to it in the 3072-dimensional feature space. Similar to the KNN implementation, the Euclidean distance (L_2 norm) is employed as the distance metric.

For every test sample, the Euclidean distance $d(\mathbf{x}_{\text{test}}, \boldsymbol{\mu}_c)$ is calculated between the sample and each of the 10 class centroids. The predicted class label $\hat{y}$ is then assigned based on the minimum distance:

$$\hat{y} = \underset{c}{\operatorname{argmin}}\ d(\mathbf{x}_{\text{test}}, \boldsymbol{\mu}_c)$$

The NCC is exceptionally fast at inference because it replaces the massive $O(N_{\text{train}} \times D)$ distance calculations required by KNN with a fixed $O(C \times D)$ calculation, where C is small (10).

However, this simplicity comes with a significant trade-off. By averaging all samples in a class, the NCC centroid fails to capture the high degree of intra-class variability present in the CIFAR-10 dataset (e.g., different breeds of "dog" or various angles of an "airplane"). This loss of information leads to a lower expected accuracy compared to kNN and thus the accuracy score of **27.72%** was expected.

V. Comparative Performance Analysis

The application of simple distance-based classifiers—the K-Nearest Neighbors (KNN) and the Nearest Centroid Classifier (NCC)—to the high-dimensional CIFAR-10 dataset yielded critical, albeit low, accuracy metrics. These results provide vital insights into the limitations of relying purely on Euclidean distance in a feature space dominated by high intra-class variability and noise.

The observed accuracies are:

KNN ($k = 1$): 29.00%

KNN ($k = 3$): 24.00%

NCC: 27.72%

These figures, which are marginally above the 10% expected from random guessing, strongly suggest that raw pixel values, even after normalization and flattening, do not constitute a linearly separable or neatly clustered feature space. The high dimensionality ($D = 3072$) and the inherent complexity of

natural images (e.g., background noise, object pose, lighting conditions) fundamentally limit the effectiveness of these naive methods. The experimentation with the KNN and NCC classifiers on CIFAR-10 leads to two fundamental conclusions regarding classification in complex feature spaces:

Distance Metrics are Insufficient for Raw Pixel Data: The ceiling performance achieved (29%) is far too low for practical application. Simple Euclidean distance on raw pixel vectors fails to distinguish high-level semantic features (e.g., ears, wheels, wings) from low-level noise (e.g., background color, lighting). This confirms the necessity of using Feature Engineering or, more typically in modern computer vision, Feature Learning through Convolutional Neural Networks (CNNs) to extract meaningful, high-level features before classification is attempted.

Trade-off Between Complexity and Generalization: The sharp drop in KNN accuracy from $k = 1$ to $k = 3$ demonstrates that less local aggregation ($k = 1$) was superior to local smoothing ($k = 3$) in this noisy, high-dimensional space. Furthermore, the NCC's comparable accuracy, despite its extreme simplicity, positions it as a highly efficient baseline. It proves that in feature spaces where noise dominates the distance calculation, simply finding the overall class mean is nearly as effective as comparing against individual neighbors, confirming the difficulty of the classification task itself. The memory and inference time advantages of the NCC make it the clear winner from an efficiency standpoint, despite its slightly lower performance than $k = 1$.

VI. References and Code Implementation

The CIFAR-10 dataset,Computer Science Dept,University of Toronto.
K-Nearest Neighbor(KNN) Algorithm(23 Aug,2025),GeeksforGeeks.
ML Nearest Centroid Classifier,GeeksforGeeks.
Gemini(2025).[Large Language Model].(Release Oct.2025).Google.
The code files are implemented in Python (.py) from scratch.