Analysis of NN, K-NN, SVM in Image Recognition: Feature and Measure.

Guan Zihan
Department of Computer and Information Science, University of Macau
Macao, 999078, China
dc22826@umac.mo

Abstract

This project is about image recognition, it measures using some proper algorithms to build classification models. In this project, several algorithms such as KNN, SVM, and NN will be used from the library, and a new NN algorithm will be designed. After that several methods will be used to analyze their accuracy. For algorithms that call directly from the library, the project will focus on using which features or measures can achieve better accuracy and recall. For the newly designed algorithm, the project will focus on why it is not as good as algorithms from the library. Hence, understand each algorithm's principles and judge their advantages and disadvantages.

1. Introduction

Image recognition involves the process of identifying and classifying objects, scenes, and features within images. In recent years, the development of autonomous vehicles has been spectacular, autonomous vehicles must recognize road signs and pedestrians, but in the real world, there are many incidents, so they may need a high-potential image recognition method. This project aims to focus on the foundational techniques of image recognition, focusing on the implementation and analysis of various algorithms, such as the Nearest Neighbor (NN), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). A new algorithm will also be designed to compare with those algorithms. This project is the basis of other more complex projects. After finishing several experiments, the results show that SVM is a better algorithm in comparison with NN and K-NN algorithms and it is better to use an edge map instead of the pixel value; use cosine similarity instead of Euclidean difference.

2. Image Recognition

Image recognition refers to the process of identifying and classifying various objects[2], scenes, and features in digital images, positioning it as a crucial element of computer vision. This capability to interpret visual information has significant implications across numerous industries. It not only enhances human abilities but also allows machines to interact more intelligently with their surroundings. One of the most compelling applications of image recognition technology is in the realm of autonomous vehicles. These cutting-edge vehicles depend heavily on sophisticated image recognition systems to detect and understand essential elements of their environment, such as road signs, lane boundaries, pedestrians, and potential obstacles. For an autonomous vehicle to navigate streets safely and efficiently, it is vital to achieve high levels of accuracy and reliability in recognizing these features in real-time. However, despite the remarkable progress in technology, real-world situations present considerable challenges. Factors like varying lighting conditions, unexpected obstacles, and occlusions can complicate the task, underscoring the need for robust image recognition methods capable of adapting to dynamic environments. As the demand for effective image recognition technologies continues to rise, it becomes spectacularly crucial to perceive the methodologies behind these advancements, as well as the challenges they face. This knowledge is critical not only for academic research but also for the practical applications that influence our everyday lives.

3. Method

3.1 K-Nearest Neighbor algorithm and Nearest Neighbor algorithm

The nearest neighbor algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point [1]. By using this algorithm, each

data in the dataset will be labeled with a class or a value as shown in figure1.

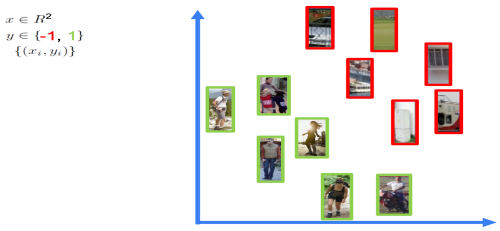


Fig. 1 Data label (Figure Credit: PPT from lecture2)

Given a test point *q,* by using this algorithm, we can search through the training point and find out the point *p* which is closest to the point *q*. This method classifies a data point by looking at the majority class among its K nearest neighbors in the feature space. The parameter K, set by the user, specifies how many neighbors to take into account. fIf K=1, the algorithm is the nearest neighbor algorithm, otherwise, it is a K-nearest neighbor algorithm. If the circumstance is about classifying the data point, KNN will assign to the label that most frequently appears among the K nearest neighbors. The distance between *p* and *q* can be measured by using features (e.g., the raw pixel values, edge map), and a measure (e.g., the average Euclidean difference, cosine similarity). Such evaluation metrics will be introduced in 4.2.

The most important advantage of the K-NN algorithm is that it can solve multi-class classification problems efficiently and simply. The disadvantage of this method is it may be computationally intensive as KNN needs to calculate all the distance between training samples. The number of K will also influence the sensitivity of noise or smoothing out some important patterns.

The Nearest Neighbor algorithm is a computationally less intensive algorithm contract with the K-nearest neighbor algorithm, given that it only considers the nearest neighbor. At the same time, this algorithm is more sensitive to noise and outliers in the data and may make this method not always representative of broader class distribution.

3.2 Support Vector Machines algorithm

Support Vector Machines (SVM) algorithm is also predominantly utilized for classification tasks. The purpose of SVM is to find a specific best hyperplane with max margin, defined as the distance between the plane and the closest data point from each class, which can separate the data points distinctly into different classes. SVM can map the original data into a higher-dimensional space where a linear separation is feasible.

The main benefits of Support Vector Machines (SVMs) are their effectiveness in dealing with complex data and their ability to use memory efficiently since they focus on key data points which are named as support vectors for making decisions. They are also good at avoiding overfitting, to be specific, they can perform well on new, unseen data if the choosing of kernels is correct. Additionally, SVMs can offer flexibility to examiners because they can use different types of kernels to adapt to different problems.

On the downside, SVMs can be slow as they require a lot of computing power, especially when working with large datasets. The efficiency of SVM heavily depends on selecting the right hyperparameters. Moreover, SVMs can have trouble with noisy data, and they are not that easy to understand compared with other algorithms.

3.3 New designed Nearest Neighbor algorithm

This algorithm is a NN algorithm. The main difference is that this algorithm is designed by myself instead of from the library. The main details of this algorithm will be introduced in 4.4.

4. Experiments

4.1 Data sets

This project utilized the dataset CIFAR10 which was a famous and widely used dataset for machine learning and computer vision. It consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 test images [2]. Moreover, the dataset is visualized, and the results are shown in figure2.

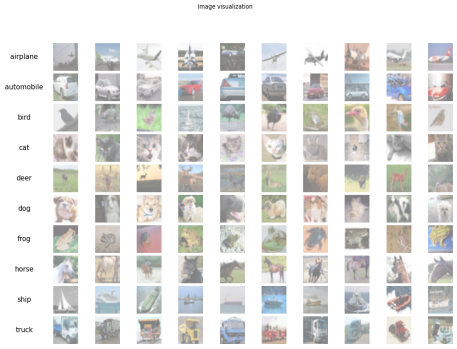


Fig. 2 Image visualization (Figure Credit: Original)

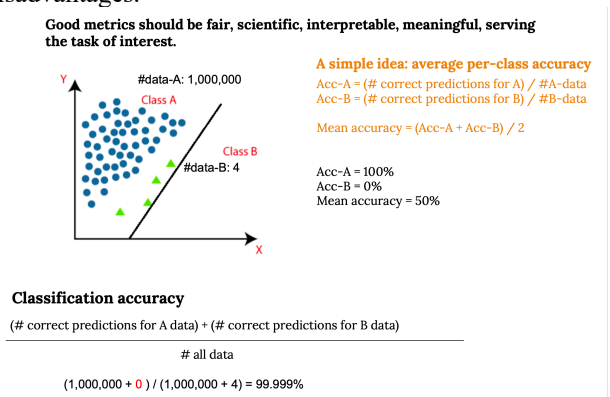
4.2 Evaluation metrics

The evaluation metrics are some metrics that can evaluate if the algorithm can achieve the goal of experiments. In this project, two kinds of evaluation metrics will be used.

4.2.1 Accuracy

Accuracy measures how often a model correctly identifies images out of the total number of images it is tested on. In

other words, it tells us the percentage of images that the model classifies correctly. As for calculating, it requires people to take the number of correct predictions the model makes, divide it by the total number of predictions, and then multiply by 100 to convert it into a percentage. While accuracy gives a good overall sense of how well the model is performing, it's important to consider it along with other metrics, especially if people are dealing with a dataset where some classes are much more common than others. In such cases, relying solely on accuracy might not fully capture how well the model is doing across all categories. Here is an explanation of why this metric may have some disadvantages.



As shown in such figure, the accuracy is high, but this algorithm can not distinguish the green triangles from the blue circle. That is the reason why sometimes people may need some other metrics.

4.2.2 Precision-recall curve

The precision-recall curve is used for evaluating classification models, especially when dealing with imbalanced datasets. It shows the relationship between precision and recall, which measures how well the model captures all the positive instances. This curve helps us see how changing decision thresholds affect these two metrics, giving a detailed picture of model performance. One of the major benefits is that it can highlight how well a model handles positive cases, which is important in fields like healthcare or spam detection. However, understanding how to interpret these curves can be hard, because it requires grasping how precision and recall move with different threshold settings.

4.3 Experiment setups

There are two parts of the experiment, the first part of the experiment is about the three algorithms from the library. In those experiments, NN, K-NN(K=10 in this project), and SVM algorithm will be called one by one, and for each algorithm, the training set will be thought of as a pool and the testing image will be thought of as a query. Features like raw pixel values and edge maps, measures like Euclidean difference, and cosine difference will be used to measure how close the two images are. For Support Vector Machines(SVM), Linear kernel and RBF kernel will be used. Metrics like accuracy and precision-recall curves are used to determine the performance of the algorithm.

To be specific, the implementation numbers and details are as follows. T means this algorithm and those metrics will be tested in the experiment Exp xx.

	NN	K-NN	SVM	Euclidean	cos	edge	pixel
Exp01	T			T			T
Exp02	T				T		T
Exp03	T			T		T	
Exp04	T				T	T	
Exp05		T		T			T
Exp06		T			T		T
Exp07		T		T		T	
Exp08		T			T	T	
Exp09			T	linear			T
Exp10			T		rbf		T
Exp11			T	linear		T	
Exp12			T		rbf	T	

Table 1 Experiment setups for algorithms from library

The second part of the experiment is about testing the newly designed algorithm, and the details are as follows. The new algorithm will be compared with the NN algorithm and analyzed for its capacity.

	NN(designed)	Euc	cos	edge	pixel
EXP13	T	T			T
EXP14	T		T		T
EXP15	T	T		T	
EXP16	T		T	T	

Table 2 Experiment setups for the new algorithm

4.4 Implementation details

As shown in Figure, this code is for Exp01 after loading the dataset and doing some basic preparation, it will initialize the K-Nearest Neighbor classifier with K=1 and Euclidean distance to fit the model on the training data and Predict the labels for the test set. After that, the code is about to calculate the accuracy, precision-recall curve, and visualize them(not shown in figure 3).

```python
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train.astype(np.float16)
x_test = x_test.astype(np.float16)
x_train /= 255.0
x_test /= 255.0
x_train_flat = x_train.reshape(x_train.shape[0], -1)
x_test_flat = x_test.reshape(x_test.shape[0], -1)
pca = PCA(n_components=0.95)  # Retain 95% of variance
x_train_pca = pca.fit_transform(x_train_flat)
x_test_pca = pca.transform(x_test_flat)
# Initialize K-Nearest Neighbor classifier with K=1 and Euclidean distance
knn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
# Fit the model on the training data
knn.fit(x_train_pca, y_train.ravel())
# Predict the labels for the test set
y_pred = knn.predict(x_test_pca)

# accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"The accuracy of the Nearest Neighbor algorithm on the CIFAR-10 test set is: {accuracy * 100:.2f}%")

# precision-recall curve
y_test_bin = label_binarize(y_test, classes=np.unique(y_train))
y_pred_bin = label_binarize(y_pred, classes=np.unique(y_train))

```

Fig. 3 Part of the code about implementation1 (Figure Credit: Original)

For other EXPs (EXP01-EXP12) from the first part of the experiment, the code is almost the same, the difference is in the algorithm and metrics. So, the details of other EXPs will not show here, but the results of the EXPs will be given as follows.

For the second part of the experiment, the details for the new method are shown as follows. Notice that this is the only difference in the experiment.

```python
class NearestNeighbor:
    def __init__(self, metric='euclidean'):
        self.x_train = None
        self.y_train = None
        self.metric = metric

    def fit(self, x_train, y_train):
        self.x_train = x_train
        self.y_train = y_train

    def predict(self, x_test):
        num_test = x_test.shape[0]
        y_pred = np.zeros(num_test, dtype=self.y_train.dtype)

        for i in range(num_test):
            distances = cdist(self.x_train, x_test[i:i+1], metric=self.metric).flatten()
            nearest_neighbor_index = np.argmin(distances)
            y_pred[i] = self.y_train[nearest_neighbor_index]

        return y_pred

```

Fig. 4 Details for class Nearest Neighbor (Figure Credit: Original)

The provided code defines a Nearest Neighbor class with the primary purpose of performing simple classification using the nearest neighbor algorithm. This class allows users to specify the distance metric, defaulting to 'Euclidean'(which will be changed in other EXPs), which is used to calculate the distance between data points. The Nearest Neighbor class first stores training data through its fit method, enabling the model to remember features and associated labels. Then, using the predict method, it classifies new input data by finding the nearest stored training samples and assigning the corresponding labels.

4.5 Results

For each test, the output will be two images and one number, take the result from EXP01 as an example. It is shown in figure 5,6,7. This kind of output for other EXPs will not be given as it can not show the difference directly. However, the data visualization of their average accuracy and average precision will be given to help measure their property. The result will be given in Figure 8-figure 12.

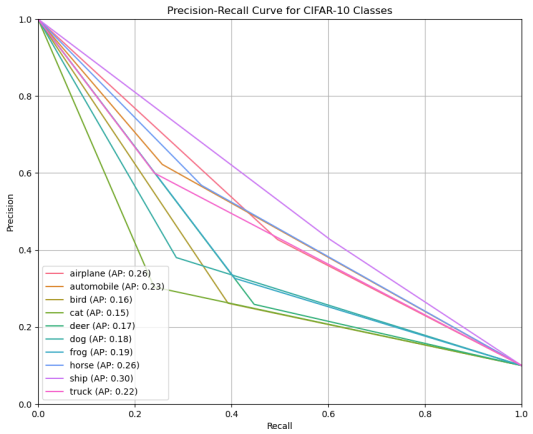


Fig. 5 Precision-Recall Curve (Figure Credit: Original)

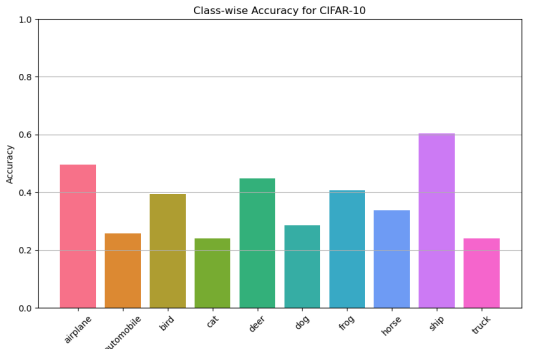


Fig. 6 Class-wise accuracy (Figure Credit: Original)

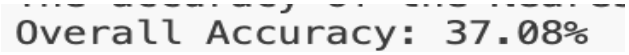


Fig. 7 Overall Accuracy (Figure Credit: Original)

The result of EXP01-04 is shown in Figure 8, the result of EXP05-08 is shown in Figure 9, and the result of EXP09-12 is shown in Figure 10, and the result of EXP13-16 is shown in Figure 11. Figure 12 is used to compare each algorithm.

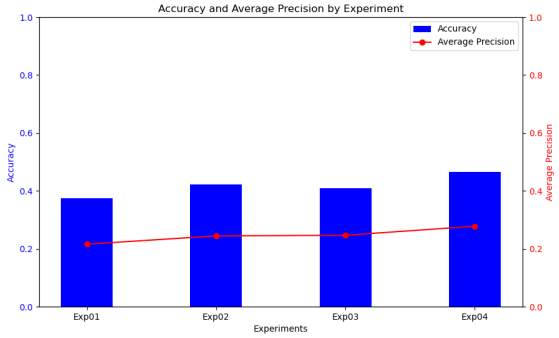


Fig. 8 Result of EXP01-EXP04 (Figure Credit: Original)

Experiment Results:

Exp01: Accuracy = 0.3708, Average Precision = 0.2157

Exp02: Accuracy = 0.4215, Average Precision = 0.2443

Exp03: Accuracy = 0.4094, Average Precision = 0.2465

Exp04: Accuracy = 0.4660, Average Precision = 0.2778

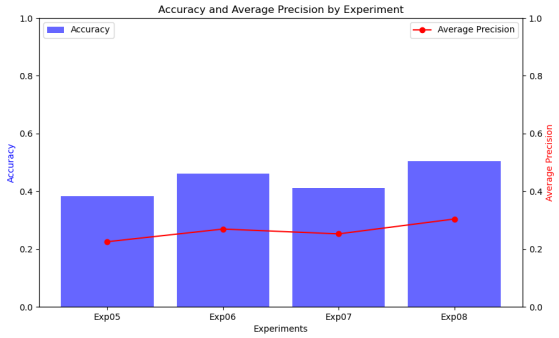


Fig. 9 Result of EXP05-EXP08 (Figure Credit: Original)

Experiment Results:

Exp05: Accuracy = 0.3834, Average Precision = 0.2252

Exp06: Accuracy = 0.4612, Average Precision = 0.2691

Exp07: Accuracy = 0.4109, Average Precision = 0.2524

Exp08: Accuracy = 0.5033, Average Precision = 0.3039

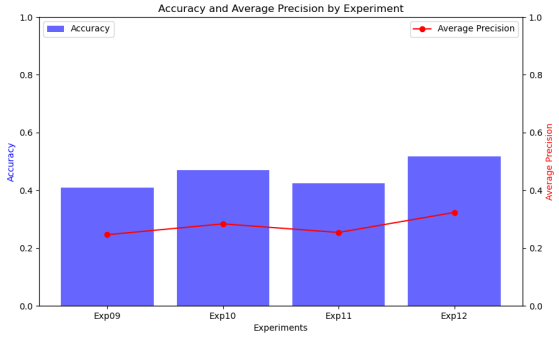


Fig. 10 Result of EXP09-EXP12 (Figure Credit: Original)

Experiment Results:

Exp09: Accuracy = 0.4092, Average Precision = 0.2460

Exp10: Accuracy = 0.4700, Average Precision = 0.2836

Exp11: Accuracy = 0.4240, Average Precision = 0.2538

Exp12: Accuracy = 0.5168, Average Precision = 0.3233

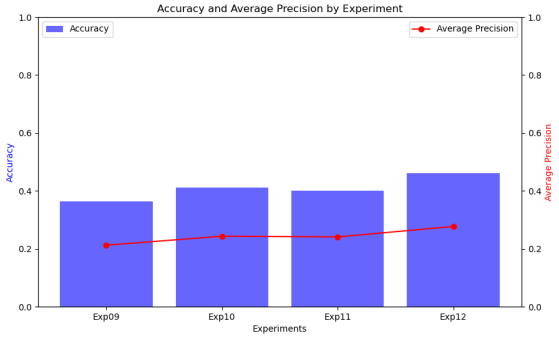


Fig. 11 Result of EXP13-EXP16 (Figure Credit: Original)

Experiment Results:

Exp13: Accuracy = 0.3643, Average Precision = 0.2123

Exp14: Accuracy = 0.4115, Average Precision = 0.2432

Exp15: Accuracy = 0.3998, Average Precision = 0.2408

Exp16: Accuracy = 0.4620, Average Precision = 0.2769

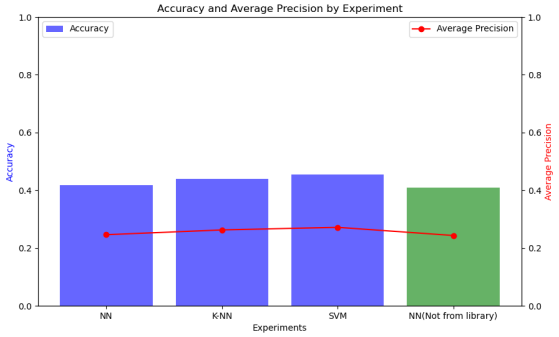


Fig. 12 Model Performance Comparison (Figure Credit: Original)

4.6 Analyses

The visualize of all the results of those experiments is in Figure, figure8, figure9. It is easy to find out that the accuracy of using the K-NN (K=10) algorithm and SVM algorithm is higher than the NN algorithm, and under the same algorithm, the use of edge map and cosine similarity will increase the accuracy as long as recall and precision. The algorithm built by myself gets the worst quality.

4.6.1 NN and K-NN(K=10)

The most important difference between those algorithms is the sensitivity to the noise, which is decided by K. NN (K=1) is very sensitive to noise and outliers. Since it only considers the nearest neighbor for classification, any noise in the dataset can significantly impact the model's decision. The decision boundary is more jagged and overly complex, potentially capturing noise as if it were a signal, leading to overfitting.

By considering more neighbors (K=10), the algorithm is more robust to noise. It incorporates more sample information, which can lead to more accurate predictions particularly when the dataset contains noise or outliers. The decision boundary is smoother and more generalized, which helps prevent overfitting by averaging out the noisy labels of individual neighbors.

4.6.2 Edge map and pixel value

Edge maps can greatly enhance image recognition in the CIFAR-10 dataset compared to using raw pixel values. By focusing on the outlines and boundaries of objects, edge maps help recognition algorithms zero in on the most important features that define each class. This not only improves the model's ability to generalize across different images but also makes it more resilient to noise and changes in lighting since it captures meaningful structures instead of pixel-level variations. Additionally, edge maps reduce the amount of data the model needs to process, making computations faster and training more efficient. Lastly, they provide a clearer way to measure similarity between images, allowing algorithms to compare shapes and patterns rather than getting distracted by varying pixel values. Overall, using edge maps makes image recognition systems more accurate and reliable when dealing with the diverse images found in the CIFAR-10 dataset.

4.6.3 Cosine similarity and Euclidean difference

Cosine similarity tends to outperform Euclidean difference when applied to the CIFAR-10 dataset for several reasons. First, cosine similarity focuses on the angle between two vectors, which means it measures the orientation rather than the magnitude of the feature representation. This is particularly beneficial in image recognition, as it helps detect similar shapes and patterns regardless of the lighting or scale variations in the images. Since the CIFAR-10 dataset features a diverse range of classes and varying conditions, this approach makes it more robust to those inconsistencies.

Moreover, using cosine similarity mitigates the impact of absolute pixel value differences, which can lead to misleading comparisons when using Euclidean distance. For example, two images that are visually similar but differ in brightness may have a high Euclidean distance, while their shapes remain closely aligned. Cosine similarity emphasizes structural similarities over pixel-level discrepancies, providing a clearer and more reliable measure for classification tasks in the CIFAR-10 dataset. Overall, when it comes to assessing similarity in image features, cosine similarity proves to be a more effective and resilient choice.

4.6.4 SVM algorithm and K-NN algorithm

The Support Vector Machine (SVM) algorithm can often do better than the K-NN(K=10) algorithm due to its ability to handle high-dimensional data more effectively. CIFAR-10 consists of complex and varied image data, where each image is typically represented by thousands of features. SVMs are designed to find the optimal hyperplane that maximizes the margin between different classes, making them particularly adept at creating robust decision boundaries even in high-dimensional spaces. In contrast, K-NN relies on local information from the nearest neighbors and can struggle with large datasets due to scalability issues and sensitivity to irrelevant features or noise. Additionally, SVMs can be enhanced with kernel tricks, which makes it able to model complex, non-linear relationships, this is the reason why the SVM algorithm can do better in capturing the intricate patterns present in CIFAR-10 images.

4.6.5 The newly designed algorithm and algorithms called directly from the library

This part is about no analysis of why the algorithm used in EXP13-EXP16 is the worst in comparison with other algorithms. The first reason may be that scikit-learn's implementation is highly optimized for performance and efficiency, benefiting from extensive development and fine-tuning over the years. Secondly, I use the '*cdist*' function from '*Scipy.spatial.distance'* for distance computation, which can be less efficient than optimized methods used by Scikit-learn. Scikit-learn can leverage specialized libraries like BLAS or SIMD instruction for faster computations. Thirdly, the Scikit-learn implementation might be more efficient in terms of memory usage. Our implementation loads the entire training set into memory and computes distances for each test instance separately, which can be more memory-intensive and slower.

5. Conclusion

In this project, I explored the effectiveness of different image recognition algorithms applied to the CIFAR-10 dataset, gaining important insights into their performance and suitability for complex visual data tasks. Through a series of experiments, we found that Support Vector Machines (SVM) generally outperformed other tested

algorithms. The strength of SVMs lies in their ability to efficiently handle high-dimensional data and establish clear decision boundaries, particularly when using non-linear kernels, such as the Radial Basis Function (RBF). This capability is crucial for accurately classifying the diverse image classes present in datasets like CIFAR-10.

A key finding from our experiments is the significant advantage of using edge maps over raw pixel values as features. Edge maps concentrate on capturing the structural essence of images, making recognition systems less susceptible to variations caused by changes in lighting or noise. This form of feature extraction enhances the performance of machine learning algorithms by focusing on the most informative image components and reducing the overall data complexity processed during classification tasks.

Additionally, I found out that cosine similarity is a more robust metric for measuring similarity in image recognition than Euclidean distance. By focusing on the angular orientation between feature vectors rather than just magnitude differences, cosine similarity better accommodates variations in scale and lighting. This method helps in accurately identifying patterns and shapes, which are crucial for successful image classification.

Our study also compared the K-Nearest Neighbor (K-NN) approach with the simple Nearest Neighbor (NN) method. While K-NN showed some improvements in handling noise, it was still less efficient than SVM in terms of computational demands and performance quality. The custom nearest neighbor algorithm we developed did not perform as effectively as the libraries' implementations like those in Scikit-learn, which are finely tuned for both efficiency and accuracy. This finding emphasizes the benefits of using established machine learning libraries, which include optimized routines and are crucial for achieving high-performance levels in image recognition tasks.

References

[1] Alex Krizhevsky. Convolutional Deep Belief Networks on CIFAR-10, 2010

[2] Xi Wang. Research on logistics optimization and development strategy of e-commerce platform based on NSGA-II, 2023 International Conference on Power, Electrical Engineering, Electronics and Control (PEEEC)

[3] Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part II

[4] D. Cire¸san, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. Arxiv preprint arXiv:1202.2745, 2012.

[5] D.C. Cire¸san, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classification. Arxiv preprint arXiv:1102.0183, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

Acknowledgment

1. I acknowledge that I had discussed my project with Prof. Shu Kong.

2. I acknowledge the use of Copilot to search for resources about the basics of features and help me debug my code to visualize the result of experiments.

3. I acknowledge the use of Grammarly to check grammar.

4. I acknowledge the use of Turnitin to check repeatability (less than 14%).