Naive 1NN Classifier and Feature Extractor With CIFAR10 Dataset

Jawad Aziz Khan, ***ID - 1530457642*****,**
Ishfaq Zaman, ***ID - 1530173642*****,**
Kazi Habibur Rahman Dipto, ***ID - 1612823642*****,**
Sabit Ibn Ali Khan, ***ID - 1610377042***

North South University
CSE468 Computer Vision

1. Introduction

For Assignment 1, we have implemented 4 different but naive methods for extracting features and performing image classification using the CIFAR 10 image dataset. The techniques, or models, used are given below:

- A naive classifier that matches based solely on pixel distance.
- A naive classifier that uses color histograms instead of pixel values.
- A model based around the Histogram of Oriented Gradients method
- A model based around the state of the art Scale Invariant Feature Transform method

Using the 4 techniques mentioned above, we used 1 Nearest Neighbour classifier to compare features of images. When comparing the features, we found the nearest neighbor to each class and assumed the label of the nearest neighbor to be the label of the class as well, and we achieved an accuracy ranging from about 20 to 50 percent using the various methods.

2. Literature Review

We have read several papers regarding image classification methods and feature extraction to better our understanding of complex computer vision algorithms. Below is a detailed review of what we have learned:

2.1. One Nearest Neighbour

1-NN is a unification of the K Nearest Neighbour method and it is highly intuitive. In simple terms, to identify a certain class, we need to find its nearest neighbor among all the training points, and then assign a single label to the class we are trying to identify the label of its nearest neighbor, rather than multiple possible labels. 1-NN is conceptually very simplistic and it performs rather well in low dimensions for complicated decision surfaces. The metrics that have been used to measure the success of the 1-NN classifier are:

- **Confusion Matrix** Given a data set whose true values are known, a confusion matrix can be used to describe the performance of any classification methods, in this case 1-NN, on said data set.
- **Accuracy:** Accuracy calculates how often our classifier predicts correctly.
  (Accuracy = True Positive + True Negative) / Total
- **Precision:** Precision is simply a measure of how many correct ”yes” predictions our classification method gives us.
  Precision = True Positive / (True Positive + False Positive)

- **Recall:** Also known as sensitivity, is a measure of how many classes were identified correctly. Recall= True Positive / (True Positive + False Negative)
- **ROC Curve:** It is a curve that visualizes the performance of any classifier method over all possible threshold values. The y-axis of the curve is True Positive rate and the x-axis is False Positive rate.
- **F1 Score:** The F1 score is used to measure the accuracy of any test. It takes into account the recall and precision values that we have talked about earlier. It is the weighted average of precision and recall. The best F1 score is 1, indicating perfect precision and recall, and the worst F1 sore is 0.

2.2. HOG

Histograms of Oriented Gradients for Human Detection is a paper written by Navneet Dalal and Bill Triggs. The purpose of the paper was to find a novel way of detecting the human form in a given image. It shows that HOG descriptors perform much better than other existing methods when it comes to human detection. The paper comes to a conclusion that factors such as good orientation binning and contrast normalization all play a big role in determining how good the results are.

The principle of the methodology is that the shape and appearance of an object in an image can be identified to a good enough extent by observing the gradient distribution of the intensity and by identifying the edges. In the HOG method, the image to be classified is taken and the gradient in a certain region is calculated and the gradients are categorized into various bins based on their orientations. To achieve invariance towards factors such as differences in lighting, shadows and so on, contrast normalization is carried out on each block.

One advantage of the HOG method of classification is that it is able to achieve an acceptable amount of rotational and scale invariance as well as other forms of invariance. The paper used two data sets, one being the MIT pedestrian database (200 test images and 509 training images). The range of different poses of the pedestrians in this data set were very limited and as a result the detectors gave almost flawless results when using this data set. To make it more challenging for the detectors, a more complicated data set, INRIA (1805 images) was used. The people in these photos were cropped from a diverse amount of personal photos and the poses and background the people were standing against largely varied. [1]

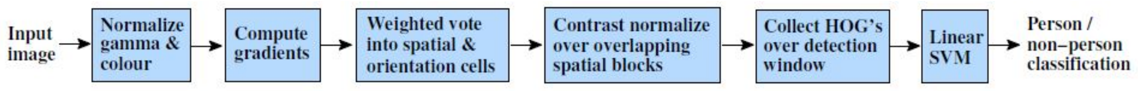
Input image
Normalize gamma & colour
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Linear SVM
Person / non–person classification

Fig. 1. HOG algorithm flowchart

2.3. SIFT

Object Recognition from Local Scale-Invariant Features is a paper authored by David G. Lowe. The detector system mentioned in the paper relies on image features that are unaffected by mane factors.The system uses local factors that are invariant to factors such as rotation and transformation. The features are identified by detecting certain stable points in the image. Keys are made from the images and said keys are sent to a nearest-neighbor detecting system that identifies keys that match to candidates. Previous methods of object detection such as color histograms, while having shown efficiency on isolated objects , are not very effective on images that are cluttered or occluded due to their features being much more global in nature

The SIFT method breaks down an image into a number of local features that are invariant to certain factors. This system is more efficient when compared to previous systems because image features generated by previous systems were variant to changes in scale and rotation. SIFT detects the local image features by initially identifying that are either the maximum or a minimum of a Difference-of-Gaussian function. Each local image feature becomes partially variant to affine or three dimensional projection by blurring the locations of the gradients in the image. The vectors produced are called SIFT keys. A single image will produces on an order of 1000 keys, needing only 1 second of computation time to do so.

The generated keys are used as the input of a nearest-neighbor method to match the keys to candidate objects. A minimum of 3 keys must match with candidate objects for the conclusion to be drawn that the object might possibly be present there. [2]

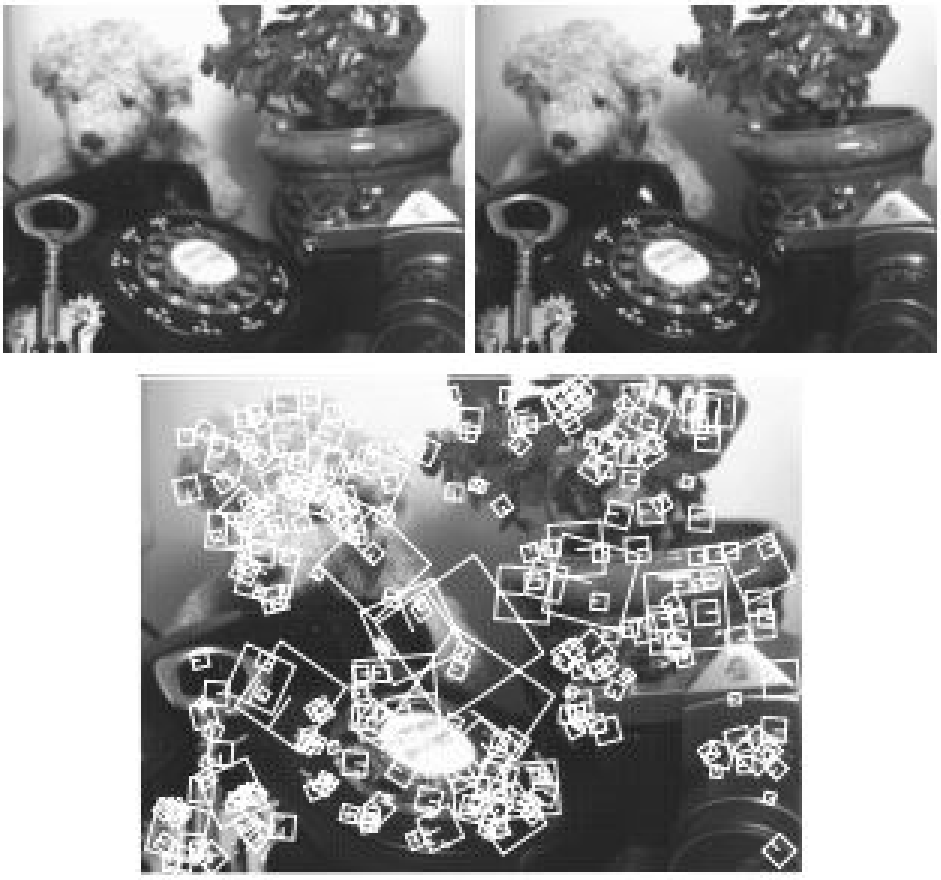

Fig. 2. SIFT descriptor finding matching keypoints

3. Experimental Setup

3.1. Dataset and Preprocessing

Before we could start extracting features from our dataset we needed to preprocess the image data into a workable format such that our scripts could take them as parameters and return coherent results

3.1.1. Dataset: CIFAR10

The CIFAR10 dataset is one of the most frequently used datasets when it comes to machine learning purposes. It contains color images that are 32*32 in size and have 3 color channels (RGB). There are 60,000 images divided in 10 classes, which means there are 6000 images in each class. we found it in a binary batch files, This means that there are 5 batches of training images and 1 batch one batch for testing. The images in the test set are randomly selected from every class and the rest of the images are in the training batches in random order, and one class in the training batch may have more images than another

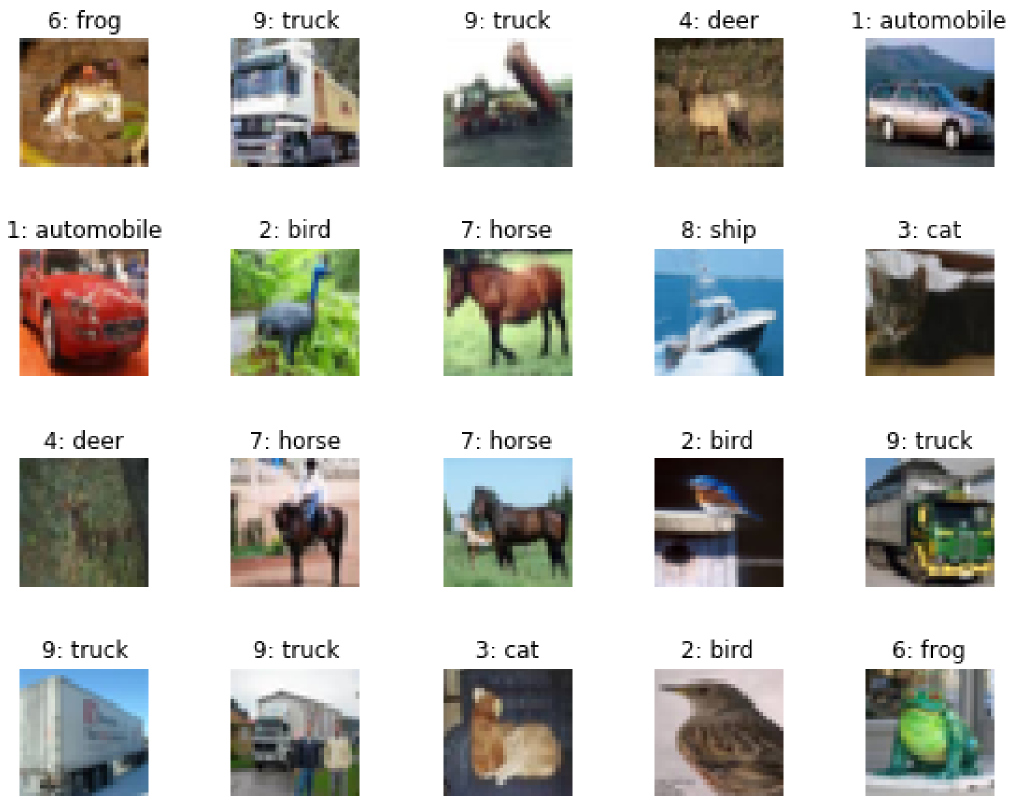

Fig. 3. Some samples from the CIFAR10 dataset

3.1.2. Train Test Division

The dataset consists of a total of 60,000 images that are divided into a train set of 50,000 that will be used by the NN model to look for a match and a test set of 10,000. It is to be noted that images in the test set are not presemt in the train set as our NN classifier would find a perfect match in that case.

3.1.3. Pickling and Vector Dimensions

Since the dataset was acquired in a binary format they required pickling to be loaded into memory. The image vectors were found already flattened into a long 1D vector of size 3072. We had to use numpy's "reshape" function to wrangle it into a suitable RGB form. Once loaded the new RGB vectors were saved as an "NPY" file for easy access.

3.2. Classifier and Extractor

3.2.1. Features

The features that are to be extracted have been selected with the pretense of providing a better understanding of how computer vision behaves and the varying degrees of results acquired from said features. The features used are:

- **Raw Flat Pixel** This is simply a long flattened 1D version of an RGB image vector
- **Color Histogram** A color image vector is read in terms of the HSV (Hue, Saturation, Value) cylindrical coordinate color model and its Hue values are binned into a histogram

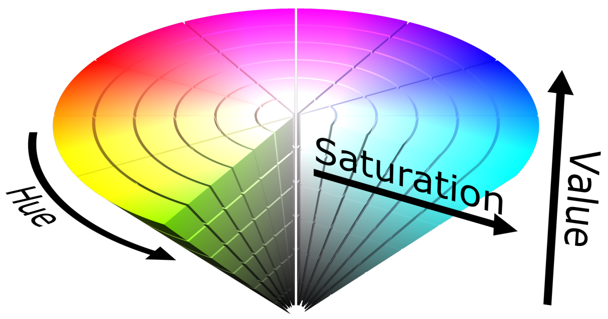

Fig. 4. HSV Color Model

- **Histogram of Oriented Gradients** A feature histogram containing the frequency of particularly angled edges using the state of the art technique from the paper written by
- **Scale Invariant Feature Transform** A descriptor vector that represents a "Bag of Visual Words" which contains features that are localized and are resistant to scale deformations

3.2.2. Extractor

The extractor functions had to accommodate the [3,32,32] dimension vector into algorithm calculations that need [32,32,3] inputs and is reshaped as seen fit.

3.2.3. Classifier

For the classifier, a naive but simple model that calculates the least Euclidean distance between features from the train set is used to match the test image's feature vectors against every feature vector of the images in the train set, then picks the label from the matched element.

4. Results and Discussions

We have run our classifier and feature extractor on a computer with an ’AMD Ryzen 5 1600’. Each feature extraction process took a different amount of time, since they have varying degrees of complexity.

The classifier matches euclidean distance with every training entry and therefore the prediction process took quite a lot time.

4.1. Raw Flat Pixel

The size of the vector was 3,072 which is why, even the calculations were fairly simple ones, it took more time than expected. The large vector size contributed to the long computation time which, on the aforementioned device, was exactly **24 minutes**.

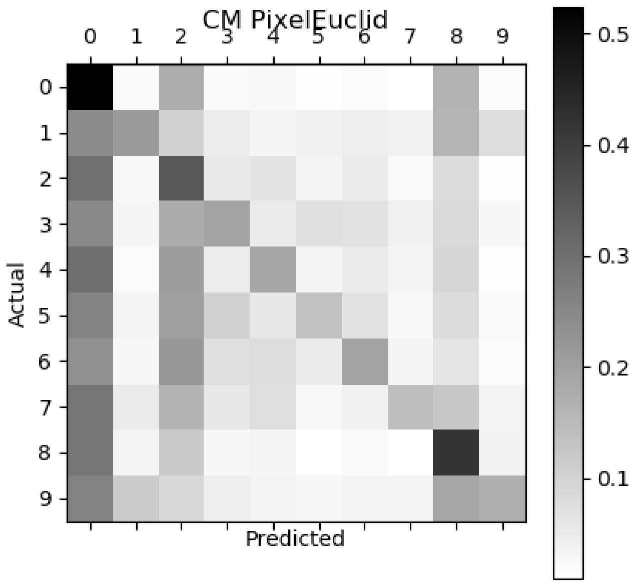

Fig. 5. Confusion Matrix for Pixel Euclidean method

The classification method that we ran gave us an accuracy of **25 percent** using this process. The label 0 (airplane) was heavily predicted incorrectly and 8 (ship) was predicted correctly almost half the time.

4.2. Color Histogram

The bin size for the color histograms was set as 10 by default, which resulted in a small vector size. This lead to much faster computation. The feature extraction aspect of it took only 7 minutes, while the prediction took a further 5 minutes, a total of **12 minutes**. However, the label results were very spread out as can be seen by the confusion matrix.

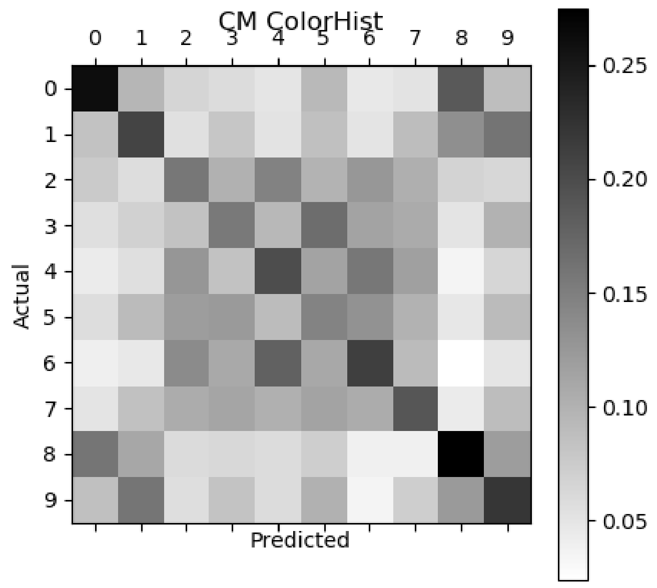

Fig. 6. Color Histogram

The accuracy that we got using color histograms was only **20 percent**. A huge number of labels were classified incorrectly because color histograms are not a very good description of an image.

4.3. HOG

We selected a kernel of 4x4 and converted all 4 by 4 areas of images into a histogram of oriented gradients. The vector size was 324. The histogram contained 9 slots binned based on rotation and each contained the magnitude of the gradient. We selected a smaller kernel because our images are very small and hence we didn't use 8*8 kernel as was used in the paper. For the purpose of gradient subtraction the kernel we used was [1,0,-1] instead of [-1,0,1] as was used in the paper as well.

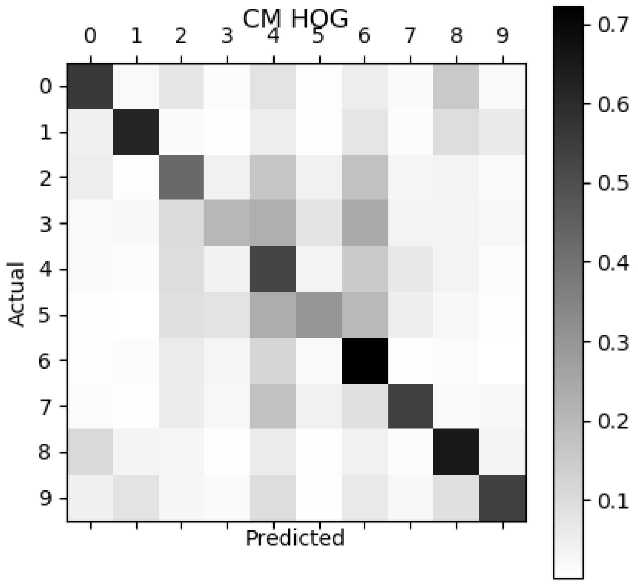

Fig. 7. HOG

Our classification gave us an accuracy of **50.8 percent** using HOG. The time to completion of the process was a whopping **5.2 hours** and the reason behind it was that our implementation used a large amount of looping for every single image. Hence it took a large amount of time to complete the entire process.

4.4. SIFT

to be completed shortly

5. Conclusion

cannot be concluded as experiment is not complete as of yet

References

[1] Navneet Dalal. *Histogram of Oriented Gradients*. University of British Columbia, 2005.
[2] David G. Lowe. *Object Recognition from Local Scale-Invariant Features*. UoBC, 2004.