# Image Classification - Final Report (Group P13)

**Kevin Gao, Rachel Chen, Srujana Marne Shiva Rao, Zunaid Sorathiya**
Department of Computer Science, North Carolina State University
Raleigh, NC 27695
{kgao, rschen, smarnes, zhsorath}@ncsu.edu

## 1 Background & Introduction

Humans can recognize and classify a wide variety of objects in images and videos into categories with minimal effort despite changes in view perspective and object scale. However, manually classifying objects across thousands of images and videos is time-consuming and inefficient. Computational systems can reduce time to perform such tasks, but struggle to accurately predict image classes [5].

Image classification is a very popular problem. Solutions range from traditional Machine Learning models to newer Computer Vision techniques. Popular solutions involve using neural networks, especially Convolution Neural Nets [6]. Though we will mention these in our paper, we were interested in finding other ways to tackle this problem. We will present alternate methods, such as Naive Bayes and K-means Clustering, that aren't commonly associated or used for image classification. Our goal is to exemplify that big problems often require many different solutions or perspectives. We hope to provide a wider scope with our presented models.

## 2 Method

We started by selecting a few classifiers that we believed would perform well with correctly pre-processed image data. After some trial-and-error, we managed to reduce our selection of models and have narrowed down our focus onto the best performing classifiers. Within the time period since the Midway Report, we've added adjustments and small changes to our chosen classifiers to improve the performance and results of our models. In this paper, we will present each model along with their corresponding methods and techniques, explain all the changes we've added to them, and compare the results of all of the models to see how they performed relative to each other.

**Naive Bayes:** Because our image data type is continuous, we will be using the Gaussian Naive Bayes Model. For reference, Naive Bayes Classifiers assume that there is the strong independence between the feature, that is, the value of a particular feature is independent of the value of any other feature. The Gaussian Naive Bayes assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. In our original design of the Naive Bayes Classifier, we used a very simple setup that involved brute forcing multiple classes into one classifier model. We used no re-sampling or ensemble methods, and our original model did not factor in any sort of hidden biases or correlations. The end product was a very sub-par classifier with low accuracy rates and inconsistent results.

In the new version of the Naive Bayes Model, we address many of the problems with our previous design in the following ways:

- For feature extraction, we keep our original image processing method, converting all images into gray-scale channels for each of the RGB layers, then calculating the mean pixel values and the standard deviations for all the channels. These features also lend themselves very well to the Gaussian Naive Bayes, and they can be slotted into the calculations smoothly.


**Figure 1:** Grayscale Conversion

- Previously, we've concluded that Naive Bayes is best used for binary classification. To extend Binary to Multiclass comparisons, we borrow a widely used technique called 1A1, or 'One Against One', which we will elaborate on later.
- We use stratified Sampling to deal with categories of different sizes.
- We added k-fold and Leave One Out Cross Validation to stabilize the randomness of our results and provide more consistency in the accuracy scores.
- Lastly, we do our best to remove hidden biases and correlations between certain class groups, and we remove miscellaneous classes that our model struggles with. We will cover these details later as well.

### 1A1 Classification

In One Against One Classification, the classifier executes multi-category classification by comparing each class against every other class; given N classes, we end up with N(N-1)/2 unique pairs of classes. Thus, our model creates N(N-1)/2 machines, where each one performs binary classification on a unique pair of image classes [1]. Each machine/round of classification votes on whether an image belongs to the ground truth class or not. Each image gets voted on N-1 times, and the final overall model selects either the ground truth class or not the ground truth class, in which all votes for the latter are combined into a "mixed class" category for a given image.

### Removing Hidden Biases/ Correlations

One way to improve the performance of Naive Bayes is to remove hidden correlations. In the context of our problem, we found that there are certain categories whose images are highly correlated based on their features. One example of this case can be seen in the 'Dolphins' and 'Killer-Whales' pair

**Figure 2:** A pair of highly similar image classes

In both of the image sets above, we found that a majority of the dolphin images look very similar to many of the killer whale pictures in terms of color composition and pixel values. Isolating and training our model on these types of category pairs resulted in a very poor performance and lower than average accuracy score. We denote these as 'Type A' image similarities.

A different kind of classification bias occurs when we end up with 2 classes whose images look very distinct between the 2 categories, but within each class, the images look similar and uniform. An example of this is the 'Motorbikes-101' and 'Hibiscus' pair.

**Figure 3:** A pair of highly distinct image classes

In both of the image sets above, we found that a large portion of the motorbike images looked very similar to each other, and likewise with the hibiscus images. When we trained the Naive Bayes

classifier on this pair of categories, our model ended up doing very well, resulting in a higher than normal accuracy score. We denote these as 'Type B' image similarities.

As a preliminary step in our setup, we need to make sure the dataset is void of as many Type A and Type B biases/correlations; we remove Type A to avoid cases of under-fitting, and removing Type B to avoid cases of over-fitting. In practice, this is done by splitting the categories into different groups, separating the undesired pairs in the process. We can use tools that make this easier, such as a "Correlation Classifier", a model that, given an input class, determines which categories are most likely to be confused with the input category. We took some inspiration from Griffin et al., who implemented this classifier, in choosing how to split apart our Type A and B category pairs [2].

**Rationale:** Naive Bayes isn't typically used for Image Classification problems. Naive Bayes is based on the fundamental, or 'Naive' assumption, of independence between every pair of features, i.e all input variables are stochastically independent of each other. We know that problems can occur if these assumptions aren't held in the dataset, and in fact, this was a part of our issue with our previous model design. What we've described in the methodology is our attempt to stay as close to the independence assumption as possible, hence the reason for multiple, serial binary classification rounds. Naive Bayes is also referred to as the maximum a posteriori decision rule, or MAP rule, and different studies have shown that even if the independence assumption does not always hold, training models that are based on the MAP rule can still yield acceptable results. [4]

**K-Means clustering:**

- **Approach**: We used multiple statistical methods for obtaining final clusters. After extracting features using Pre-trained CNN Model, we used Principal Component Analysis (PCA) for dimensionality reduction to reduce number of features to be processed. Finally, these reduced number of features were passed to K-Means algorithm to cluster each image.
  - **Principal Component Analysis (PCA)**: For the image data, we had a large feature vector which cannot be processed directly by the clustering algorithms. In order to extract important features from these large feature set, we used PCA. In other words, we are making data easy to explore and visualize. Mathematically, we are transforming the data into new coordinate system so that the greatest variance is captured by the first principal component. Less Dimension means less time to train and test the model.

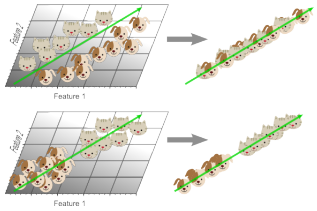


**Figure 4:** Working of PCA for Image Feature Extraction

  - **K-Means Clustering Algorithm:** A clustering algorithm recognizes patterns without specific labels and clusters the data according to the feature. K-Means algorithm assigns number of centroids based on the number clusters given as an input. Each data point is assigned a cluster whose centroid is the nearest. We used K-Means algorithm to distribute visually similar images in the same cluster. Each cluster represents a unique class of image. Mathematically, K-Means aims at minimizing the squared Euclidean distance between the data and centroid of the cluster to which it belongs.

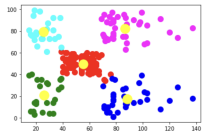

**Figure 5:** K-Means Clustering Example

- **Rationale**: Although it is uncommon to use unsupervised learning for the image classification, we implemented the K-Means clustering with limited online resources and articles available especially for the clustering of images. In the image classification, generally labeled images are passed for training the model and output labels of the test images are

predicted by the model. The idea of using an unsupervised learning algorithm came from the suggested project ideas in our coursework Project Overview Document. Moreover, Manually labelling of images is a cumbersome task, it's better to have an algorithm which classify images and identify the relationship between the images. As a result, We were able to achieve image class results closer to its actual class.

**K-Nearest Neighbors:**

- **Approach**: In addition to the Naive Bayes classifier and K-Means clustering, we also wanted to try a simpler classification method. We chose the K-Nearest Neighbors classifier. Before training our classifier, we resized all images to 32 x 32 pixels to account for differences in image size during calculations.
  - **K-Nearest Neighbors Algorithm:** K-Nearest Neighbors is a supervised learning method that takes a test data set and outputs the class each data point is expected to belong to as the majority class for the nearest K training data points.
  - **Rationale**: K-Nearest Neighbors is a very simple but versatile and has a relatively higher accuracy than many classification algorithms. We chose to use K-Nearest Neighbors because of its ability to perform well with multi-modal classes, especially since our data set has many classification categories.

## 3 Experiment Setup

We are using the Caltech 256 Image Dataset on Kaggle. It contains over 30,600 images from Google Images in 257 categories, averaging 100 pictures per category. These categories span a wide range of objects, including animals, plants, food, sports, vehicles, tools, utensils, etc.

**Naive Bayes:** We took a subset of the categories and split them into smaller groups of 20-30 classes each. For this section, we highlight one of our groups that our model trained exceptionally well on. However, we operate with the same experiment setup for any group of categories. Our example group contains 24 categories, split into 4 subgroups of 6 categories each:

- Group 1:{'Desk-Globe', 'Fighter-Jet', 'Hammock', 'Hummingbird', 'Motorbike', 'Steering-Wheel'}
- Group 2:{'Bear', 'Coffee-Mug', 'Fireworks', 'Frying-Pan', 'Leopard', 'Radio-Telescope'}
- Group 3:{'Binoculars', 'Cannon', 'Cormorant', 'Cowboy-Hat', 'Minaret', 'Skunk'}
- Group 4:{'Chopsticks', 'Grapes', 'Horseshoe-Crab', 'Knife', 'Sunflower', 'Swan'}

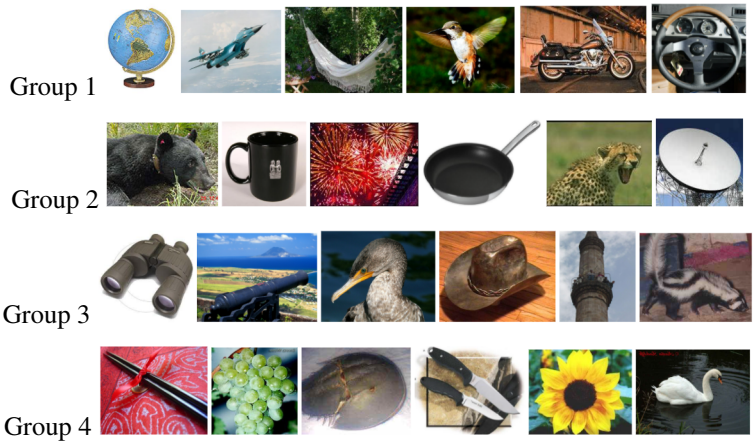


**Figure 6:** Sample Images from each Group

We tested our model using training/testing split ratios of 3:1 and 4:1 (or 75% and 80% training data). Per subgroup, we tested our model with sizes of 60 and 80 images per category. To get more coverage, we adjusted the algorithm for our K-fold Leave One Out CV, slightly tweaking it each trial to produce

different partitions on the data each time. For each combination of parameters, we run 15 trials each and average the accuracy scores to produce our results.

**K-Means clustering:** We learned in our class that Cluster analysis is the difficult part as there no fixed parameters to judge whether a cluster is good or not. We performed Silhouette analysis to identify how well the formed clusters are. We sequentially performed the below steps to achieve final clusters:

- **Experimental Design:**
  - **Data Selection:** Out of 256 different objects from the Caltech 256 image Dataset, we selected 20 different objects to cluster them into 20 different clusters of clusters of images. We performed clustering on total 2614 images. All the images are passed together at once and splitted into 80/20 Train-Test split. Below objects were used as input:

    {'Backpack', 'Beer Mug', 'Cake', 'Calculator', 'Chopsticks', 'Coin', 'Computer Mouse', 'Eyeglasses', 'Flashlights', 'Frying Pan', 'Hamburger', 'Headphones', 'Knife', 'Laptop', 'LightBulb', 'Mattress', 'People', 'Sneaker', 'Spoon', 'Watch'}
  - **Data Pre-processing:** We loaded all images and converted them into PIL object of Size 224 X 224 using `load_img` function in the Keras and further into numpy array. All numpy array images were reshaped into 3 dimensions (rows, columns and channels) array and the `preprocess_input` in the keras was used to convert reshaped numpy array to a format understandable to CNN Model.
  - **Feature Extraction:** We used Pre-trained VGG16 Convolution Neural Network for feature extraction. The 16 layer VGG16 itself is used in visual object recognition software, but we removed the output layer to obtain last second feature vector that helped the K-Means algorithm to easily separate images with different classes.
  - **Data Transformation:** We got a feature vector of size 4096 from the VGG16 Model. It is computationally very expensive if we use images with 4000+ dimensions for clustering. We transformed the feature vector from 4096 Dimensions to 100 Dimensions using Principal Component Analysis (PCA). These 100 Dimensions preserves much of the important information to distinguish each image.

**K-Nearest Neighbors:**

- **Data Selection** We selected 4 categories for preliminary testing: Fighter Jets, Grapes, Hummingbirds, and Steering Wheels.
- **Data Pre-processing** Since images in our dataset differ in dimension size, we resized the images before testing and training the classifier. We ran trials on various resizing parameters, but 32 x 32 pixels performed best, and we will only include its results.
- **Trials** We ran 15 trials each for each combination of parameters as mentioned below. We used stratified sampling to split images within each category into training and testing sets.
  - Training Set/Test Set Split Ratio: 0.75 and 0.8 (for test set)
  - Number of Nearest Neighbors: 1, 3, 5, 9

## 4 Results

**Naive Bayes:** Below are the results from our Naive Bayes Experiment Trials:

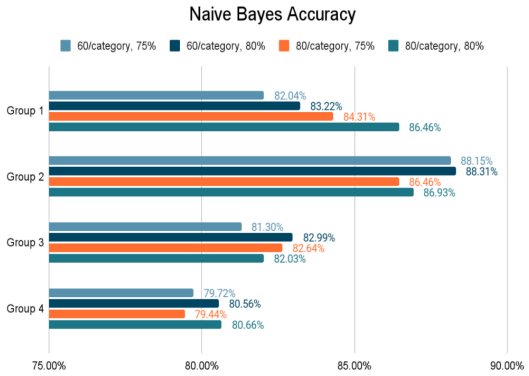


**Figure 7:** Accuracy Scores for each group of categories. Group names are the same names as given in the previous section; each label is in the format (size per category, training set size)

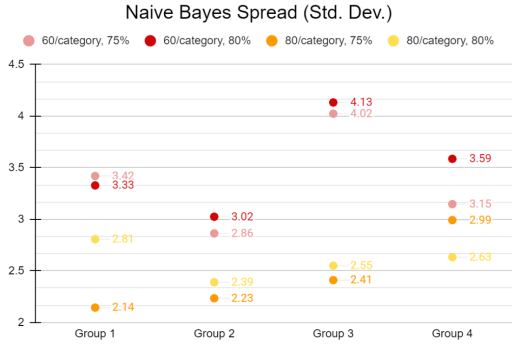


**Figure 8:** Spread/Standard Deviation among the 4 classes' accuracy scores

The results don't indicate any 'best' set of parameters for the Naive Bayes model. We know that the images we've trained the model with were selected based on our previous method criteria. With that in mind, the results show that, with carefully selected groups of categories, the Naive Bayes classifier can achieve average to above average accuracy scores on our image sets; we've even accounted for classes that slightly under-perform (Group 4) and those that slightly over-perform (Group 2). We get some more information by observing the spread of all the trials for each group (standard deviation of accuracy scores). As we move along the different sets of parameters, the results become more consistent, with the most stable parameters being the 80 images/category, 75% training set size. This is more expected, and given that Caltech 256 does not supply a consistent number of images for each category, more data or more images would certainly improve the results of our classifier.

**K-Means Results:** After performing the K-Means clustering, we got pretty good results. As we can see in Figure 9, nearly all visually similar images are classified into same cluster without a need to explicitly pass the label images.

**Figure 9:** Each cluster represents a different set of images. For example, Cluster 16 indicates a cluster of knife images.

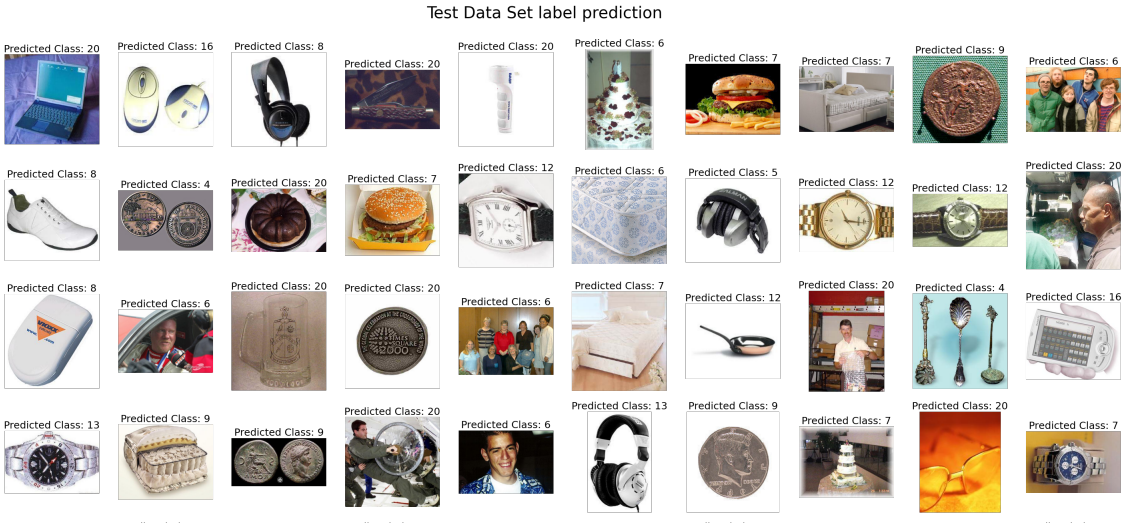


**Figure 10:** Test images snapshot with their predicted classes

We also passed the test images which weren't known to the model before to identify their predicted classes or cluster to which it belongs. From Figures 9 and 10, we can clearly see that almost all test images gets correctly classified. As an example, Predicted class of test knife images in Figure 9 is Cluster 16. Along with correct classification, there were some misclassification of images as well. The reason could be insufficient number of images per class in CalTech 256 Dataset or a need of more training data with varied angle image shots. Figure 11 and 12 shows an example of misclassification in some clusters.


**Figure 11:** Some Eyeglasses and headphones are misclassified into the same cluster


**Figure 12:** Some Hamburgers, Shoes and Cakes are misclassified as same cluster object//

**K-Nearest Neighbors:**

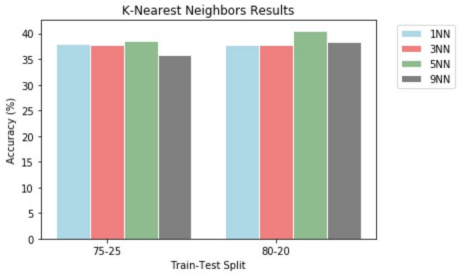


**Figure 13:** Accuracy Scores for K-Nearest Neighbors classifier.

The overall accuracy for K-Nearest Neighbors is low compared to other implemented methods. The 80-20 train-test split yields higher accuracy, especially for 5 nearest neighbors, but this difference is negligible. This is likely because the classifier makes no assumptions in the data and ignores RGB value groupings within pixels. We chose to not run further trials and focus on other classifiers.

## 5 Conclusion

There are some other methods that we wanted to test for image classification. For instance, one such model we found was an interesting mix of Naive Bayes and Nearest Neighbors: Naive Bayes Nearest Neighbors (NBNN). One paper mentioned the process of using local neighborhoods, which reduces the search space for any class, finding nearest neighbors from a subset of classes (a marked improvement on algorithm run time) [7]. Complex and intricate systems are generally preferred to achieve near-perfect accuracy. Our work can be further expanded upon, using different sampling methods, parameter tunings, etc. Better forms feature extraction, ensembles, and more data could all play a role in improving the results. Overall, we have demonstrated that using simple model designs to build classifiers, along with the right parameters and data sets, can still achieve relatively good performance, even with models that are generally not recommended for certain tasks.

## 6 References and Citations


1. Gidudu, A., Hulley, G. and Marwala, T. (2007) “Classification of Images Using Support Vector Machines“. Department of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, Private Bag X3, Wits, 2050, South Africa. https://arxiv.org/ftp/arxiv/papers/0709/0709.3967.pdf
2. Griffin, G., Holub, A. and Perona, P. (2007) “Caltech-256 Object Category Dataset”. California Institute of Technology. https://authors.library.caltech.edu/7694/1/CNS-TR-2007-001.pdf
3. Simonyan, K. and Zisserman, A. “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. Visual Geometry Group, Department of Engineering Science, University of Oxford. https://arxiv.org/pdf/1409.1556.pdf
4. Park, D. C. (2016) "Image Classification Using Naïve Bayes Classifier". International Journal of Computer Science and Electronics Engineering, MyongJi University, Korea. http://www.isaet.org/images/extraimages/P1216004.pdf
5. Scheidegger, F. and Istrate, R. (2020) "Efficient image dataset classification difficulty estimation for predicting deep-learning accuracy" https://link.springer.com/content/pdf/10.1007/s00371-020-01922-5.pdf
6. Jmour, N., Zayen, S. and Abdelkrim, A. (2018) "Convolutional neural networks for image classification". National Engineering School of Carthage Tunis, Tunisia. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8379889
7. McCann, S. and Lowe, David G. (2011) "Local Naive Bayes Nearest Neighbor for Image Classification". Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver, BC, Canada V6T 1Z4. https://arxiv.org/pdf/1112.0059.pdf


Code repository: https://github.ncsu.edu/rschen/engr-ALDA-fall2021-P13