Project 2: Report and Code Output

Alejandro J. Rigau

March 24, 2021

Abstract:

In this project I show my results of image classification using a bag-of-words model using SIFT descriptors. First, I find features in all of the training images, cluster them using KMeans and then form histograms that will later be used for training. In the training part, I used the algorithms: Nearest Neighbors, Linear Support Vector Machine and a Kernel Support Vector Machine.

3.1 Find SIFT Features

To accomplish this part I created the function ***generate_descriptors()*** function. The function takes in the path for the training or testing images and it returns a 2D array where each inner array contains the descriptors for each image.

IMPORTANT: To easily identify which features belonged to which training class, I decided to keep everything in the same order as the dataset. The dataset is ordered numerically and all the images of the same class appear together. Therefore when I do ***os.listdir(train_path)*** I will get all the names of all the images in order. This makes it easy to create labels because we can just extract the first three numbers of the image name to create the label. The first few items in the list belong to label 024, the ones around the middle of the list belong to 051 and the ones at the end of the list belong to 251.

Sample Output:

```
>>> generate_descriptors(train_path)
[array([[  6.,  31.,  34., ...,   0.,   0.,   5.],
       [  2.,  19., 129., ...,   0.,   2.,   9.],
       [ 58.,  41.,  15., ...,   0.,   0.,   1.],
       ...,
       [  1.,  14.,  50., ...,   0.,   1., 133.],
       [ 47.,  18.,   3., ...,   0.,   1.,   9.],
       [ 44.,  33.,  14., ...,   0.,   1.,   4.]], dtype=float32), array([[ 1., 14.,  4., ...,  2.,  3., 12.],
```

3.2 Clustering

Here I simply passed the returned array from ***generate_descriptors()*** and passed it to the KMeans model. I set up 100 clusters and limited the amount of iterations to 100 and restarts to 3. This gave a good balance between speed and performance. The default values are slightly higher but making them lower did not seem to make a significant difference.

3.3 Form Histograms

Now that we have the cluster and our image descriptors, I created the function ***generate_histograms()*** that takes in the path to our dataset and our descriptors. I run this function twice: Once with the training data and once with the testing data. Our output from this function is an array of histograms where each histogram belongs to an image. The array is ordered in the same way as the dataset. Each histogram is a list of numbers where each number is the number of descriptors that matched that centroid. I also normalized the values of the histogram by dividing by the sum of all the values. This makes the sum of all values equal to 1. I also created a ***generate_histograms_plot()*** function that generate a visualization of a selected histogram.

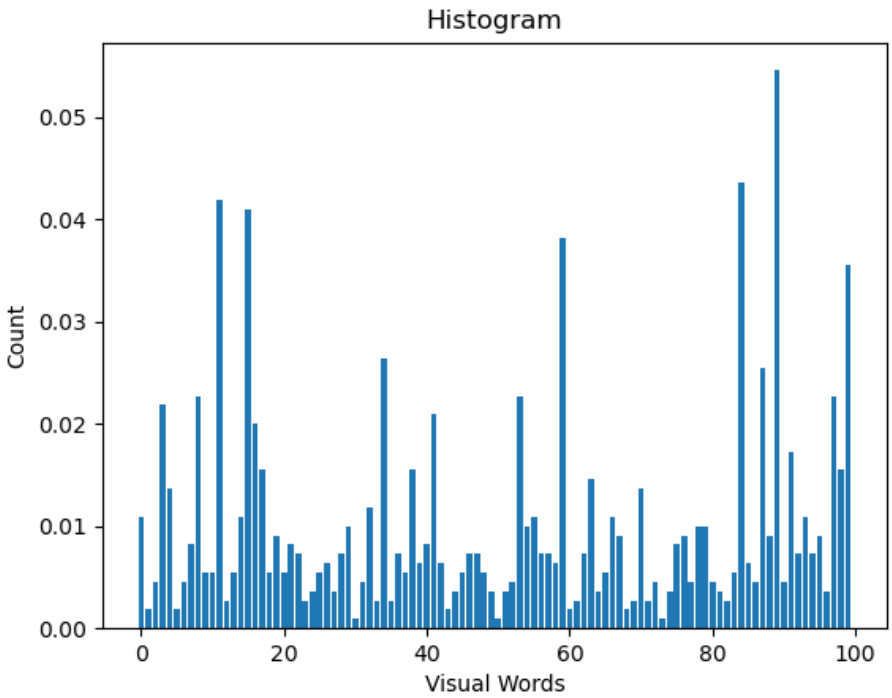

Figure 1: Example histogram from the training set

3.4 Prepare for Classification

In this section I ran all the functions I created but with the testing dataset. The functions were created in such a way that I can just pass the new testing dataset path and I can generate all the same things that I did with the training set. I also have the function ***generate_labels(path)*** that takes in a path to the dataset and returns the labels for the images. The labels are extracted from the first 3 digits in the image name.

3.5 Classification

Here I utilized the sklearn K Nearest Neighbor classifier to fit our data. My data labels follow the same convention as the dataset where 024 denotes butterflies, 051 denotes cowboy hats, and 251 denotes airplanes. I used the default parameters for the model and set the K=1. After fitting the model with the training set, I tested it out with the testing set and received an accuracy of 75%. The confusion matrix is as follows:

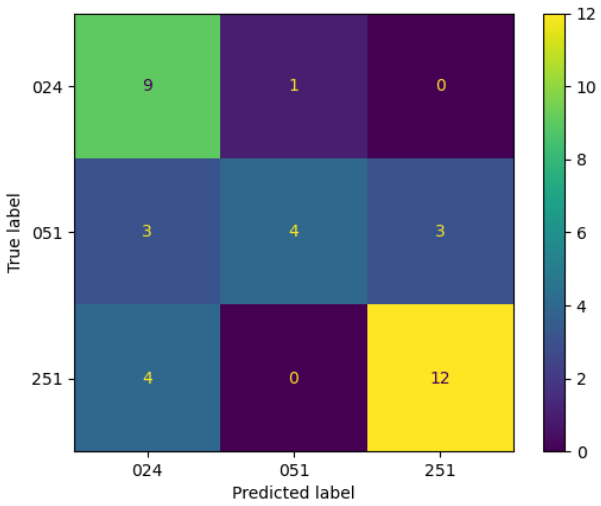

Figure 2: Confusion Matrix for the K Nearest Neighbor classifier

From the output we can interpret that the model tends to have difficulties with false positives with cowboy hats while it performs significantly better with butterflies.

3.6 Linear Support Vector Machine

In this section I utilized the sklearn linear Support Vector Machine classifier to train in my training set. I used the default parameters for the model while making sure the kernel was set to linear. My data labels follow the same convention as the dataset where 024 denotes butterflies, 051 denotes cowboy hats, and 251 denotes airplanes. I tested it out with the testing set and received an accuracy of 47%. The confusion matrix is as follows:

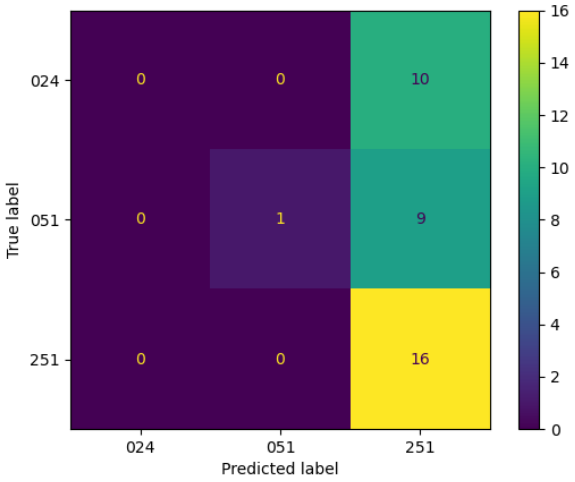

Figure 3: Confusion Matrix for the Linear Support Vector Machine classifier

From the results, we can see that the model does not seem to be able to fit the data properly. We can clearly see that these histograms are not linearly separable.

3.7 Kernel Support Vector Machine

For my last model, I utilized the sklearn Support Vector Machine classifier to train in my training set. I utilized the default parameters and only changed the kernel to rbf which is a commonly utilized kernel. My data labels follow the same convention as the dataset where 024 denotes butterflies, 051 denotes cowboy hats, and 251 denotes airplanes. I tested it out with the testing set and received an accuracy of 83%, achieving the best accuracy out of all the models. The confusion matrix is as follows:

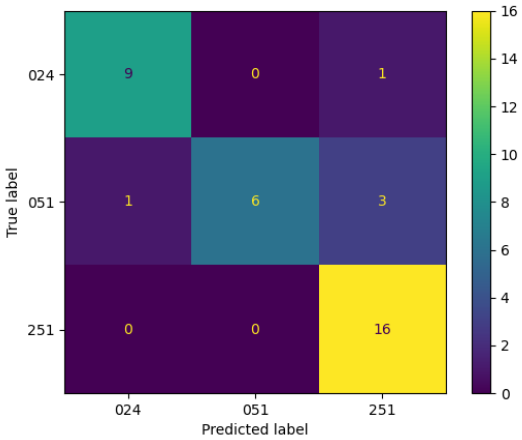

Figure 4: Confusion Matrix for the Kernel Support Vector Machine classifier

From the results we can see that it did a great job at classifying both butterflies and cowboy hats. The label that it most struggled with was the airplanes where it commonly misclassified it with a cowboy hat.

Possible Improvements

Some possible improvements to these models are parameter search and amount of data. These models have a lot of different values that can be tweaked to achieve better results tailored to our dataset. There are many search techniques that can be used to automate the parameter optimization process. Another possible improvement is the amount of data. Currently we have a very small sample size that limits the learning ability of our models. Adding more data will allow my model to generalize better to different images.