TTIC31040 Texture Classification Project

Qingyuan Liu

June 8, 2021

1 Description

1.1 Provided Resources

The goal of this project is to build a classifier that could classify textures. An image dataset called Describable Textures Dataset (DTD) with a total of 47 different classes and each class contains 120 images(a total of 5640 images) is used in this project. Within the DTD dataset, a provided split that equally split the dataset into three parts with each part denote as training, validating, and testing is also followed in this project.

1.2 Preprocessing

The project is started with encoding the categories into numerical values. All the classes are sorted alphabetically first to ensure the mapping between text and the number is consistent and easy to track.

After loading the image, all the image is converted to the gray-scale first before applying any filters. Due to the limited computational resources, every image is cropped to keep only the center part of the image with a size of 200 pixels * 200 pixels.

1.3 Applying Filter

The Leung/Malik filter bank's filter is used in this project. The filter bank consists of 48 different filters with each filter covering 49 pixels * 49 pixels. Only the pixels located within the region where the filter box would not exceed the image would be considered. Therefore, there is no padding of the image to counteract the cases of edge pixels. The edge pixels are not considered in this project for two reasons. First, padding may affect the original image thus influence the later classification. Second, unless the pattern of the texture is specifically about the edge, disregard the edge pixels will not affect the overall patterns. Additionally, this could marginally facilitate less computational resources and hardware storage as well. To further reduce the size of the data and computational time, the filter is only going to be applied to every 2nd pixel in both directions (stride equals 1).

1.3.1 Normalization of correlated output

For every pixel that applied the filters, since every filter has a filter box of 49 pixels * 49 pixels in this project, it would generate a matrix of size 49*49. The values of the matrix are summed up as the output of a filter on this pixel. Since a total of 48 filters is applied to the image, every pixel would have an output of a vector of size 48. To prevent the case that some filter may naturally output big values with high variance, thus it will dominate other filters output when calculating distance, a normalization step is applied. The output of every filter of that image is normalized to have a standard Gaussian distribution. In other words, re-scaling the output to have a unit Gaussian distribution for each filter.

1.4 k-means clustering

After applying the filters to the images, every image is now turned into a convolved image with a dimension equals to the number of filters applied. For all the convolved pixels in the training data, a k-means clustering method is used to find clusters. For every image, each pixel point is assigned to one of the clusters and the histogram of how many pixels is assigned to different clusters is now considered as the new feature of the image. The histogram is normalized so that the sum of the histogram values is equals to 1. In this implementation, 470 number of clusters is used.

Due to the high computational resources required for this step, a K-means using mini-batches is used. The idea is this: Instead of went through every data points per iteration, randomly sample a number of points from the

data and do k-means to find clusters. The batch size implemented is 1 million data points per iteration because that is the hardware limit.

1.5 Training a Classifier

A k nearest neighbor (KNN) classifier is used in this project to classify images into different texture categories. In this implementation, five nearest neighbor(k=5) for each training sample is used. The voting of the clusters is affected by distances. i.e. neighbor close by have a higher weight in voting. The distance between two samples, which in this case is the difference between two image histograms is calculated via χ^2 distance.

1.6 Tuning and Evaluation Performance

Tuning the hyperparameters using the accuracy output from validation dataset. Evaluate the model performance on the testing dataset.

2 Discussion on the implementation

2.1 Number of Clusters for K-means

A total of 10 different values is experimented with, which are 47, 47*2, 47*3 ... 47*10. The reason it is on the scale of 47 is that there are a total of 47 categories in the dataset. If there exists a perfect set of filters, all the pixel points coming from the same category would have a very close output and would form a cluster whereas all the pixel points from a different category would be far away and not joining the clusters. However, taking into accounts that all the pixel points from a single image, or the same category, can be clustered into different clusters as well, with an average of n clusters per category, a total of n*47 clusters for all the pixel points is more reasonable. This is the reason why these numbers are chosen as the number of clusters. From the accuracy plots of the validation dataset, see Figure 1, number of cluster equals to 470 have the best precision and recall values.

2.2 Selecting Classifier

There are all sorts of classifier available. Since the dataset is not big, a simple classifier such as KNN or linear classifier based on distance is recommended. The big difference between KNN and linear classifier is that KNN is non-parametric and sort of memorize all the data points and linear classifier are mostly parametric. Since all the classification depends on the distance between histograms, distance between every two samples of the training set of different texton size is plotted. Figure 2 And because the training set is not shuffled, every 40 consecutive samples are belongs to the same category. The distance between samples from same category is not that different comparing to samples from different classes. A non-parametric KNN method with voting affected by neighbor distance would less impacted because as long as there is one neighbor that is really close by, the prediction does not affected from other samples.

3 Evaluation

3.1 Performance

Out of the testing dataset, a figure of actual category vs. predicted categroy is plotted. See Figure 3. In the plot, top 5 most wrongly predicted, or most confused has also been marked. which is dotted vs. polka dotted, crystalline vs. marbled, knitted vs. pitted, perforated vs. dotted and polka dotted vs. dotted respectively.

Precision and accuracy for each class is plotted as well, Figure 4. Some of the class did better than others. The average precision is 0.25 and average recall is 0.20.

The averaged confusion matrix is as Figure 5

3.2 Potential Improvement

3.2.1 Normalization

One thing that could be improved, is the normalization step I implemented after applying filters to the image. The normalization is conducted with the base of every image. Considering a case, under the same filter, an image produced a very high value vs. an image produced a very low value, if their standard deviation is the same,

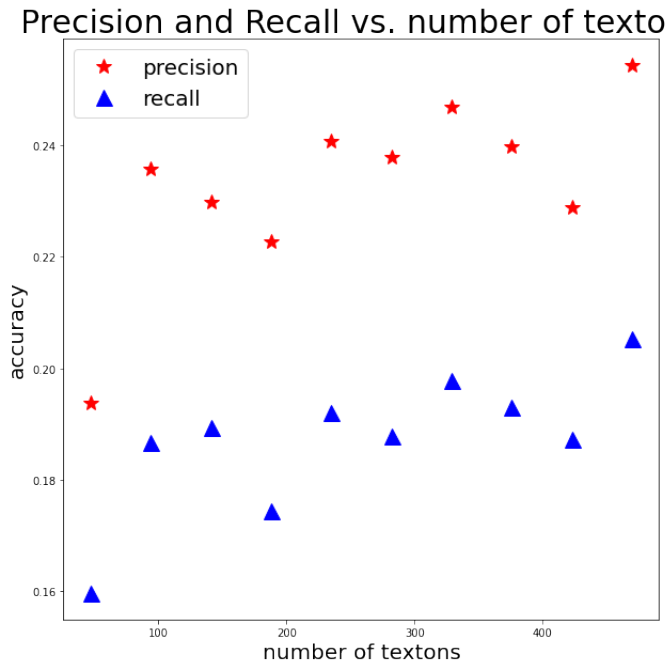

Figure 1: Precision and Recall of different texton size of validation dataset. Number of textons is denoted by how many clusters used in K means

after normalization, these two images would have similar pixel points after correlated. A better normalization step would includes outputs from all the image points for a specific filter instead of one image. This will reserve more information on the mean and possible the variance.

3.2.2 number of clusters for kmeans

From Figure 1, it seems the precision and recall value is still increasing with a higher number of clusters for kmeans. Maybe a even higher number of clusters for kmeans my still help.

3.2.3 speeding

Most of the times spent in this project is to apply filter to every pixel points. Using GPU to take advantage of parallel computing might speed up the process.

4 Limitation and Further Direction

The implementation did not take into account such as orientation, scale. Because the filter size is fixed, a different scaled picture will drastically produce a different output after filter applied. The most confused labels is dotted vs. polka dotted. Using extra information such as gradient information, scale information, location information will definitely help the classification. The solution is to add additional features such as the sift descriptor for every pixel of the image.

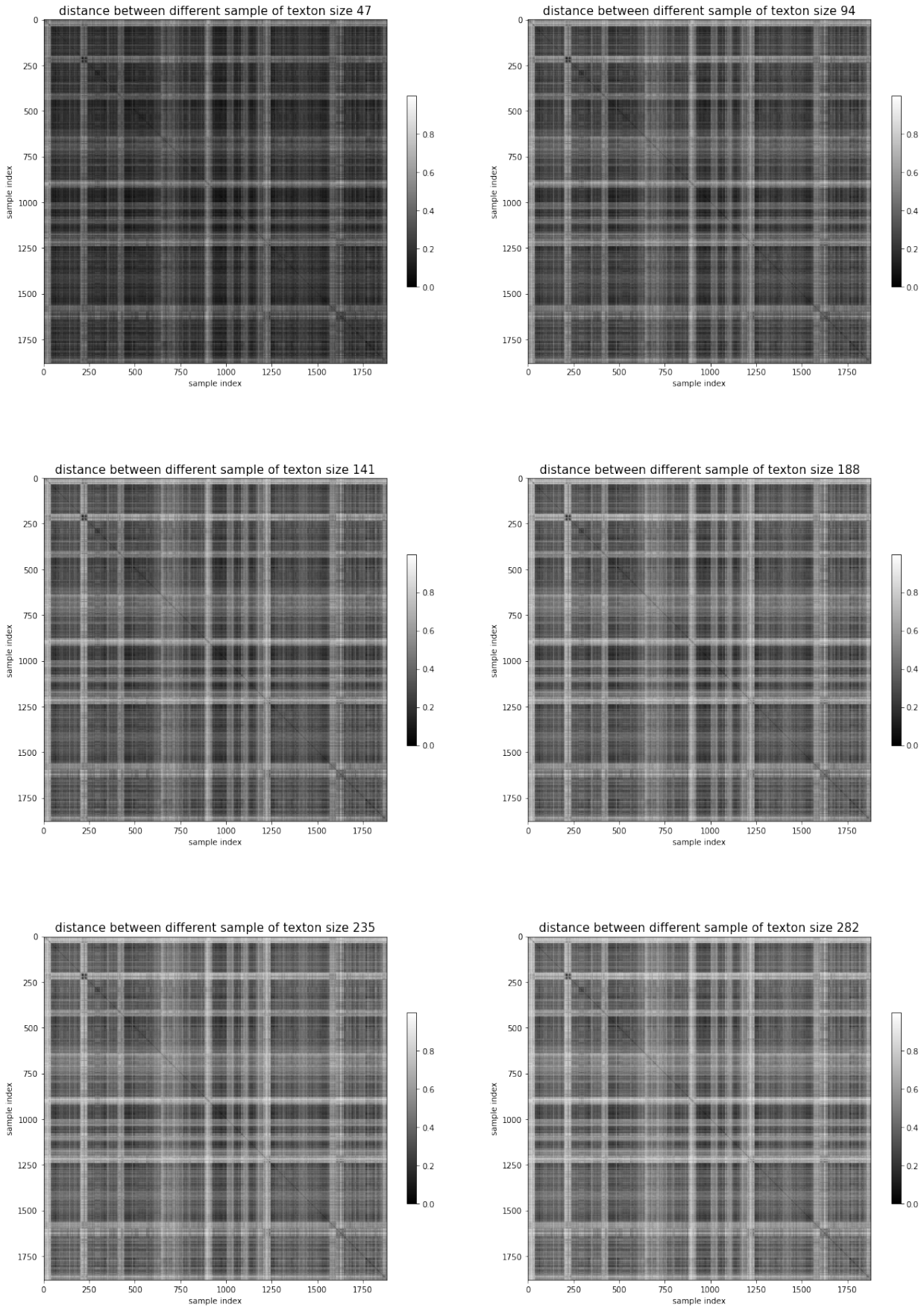

Figure 2: distance between two training samples. Every 40 samples is from same category, a block of 40*40 along the diagonal line is expected and can be observed but not by much

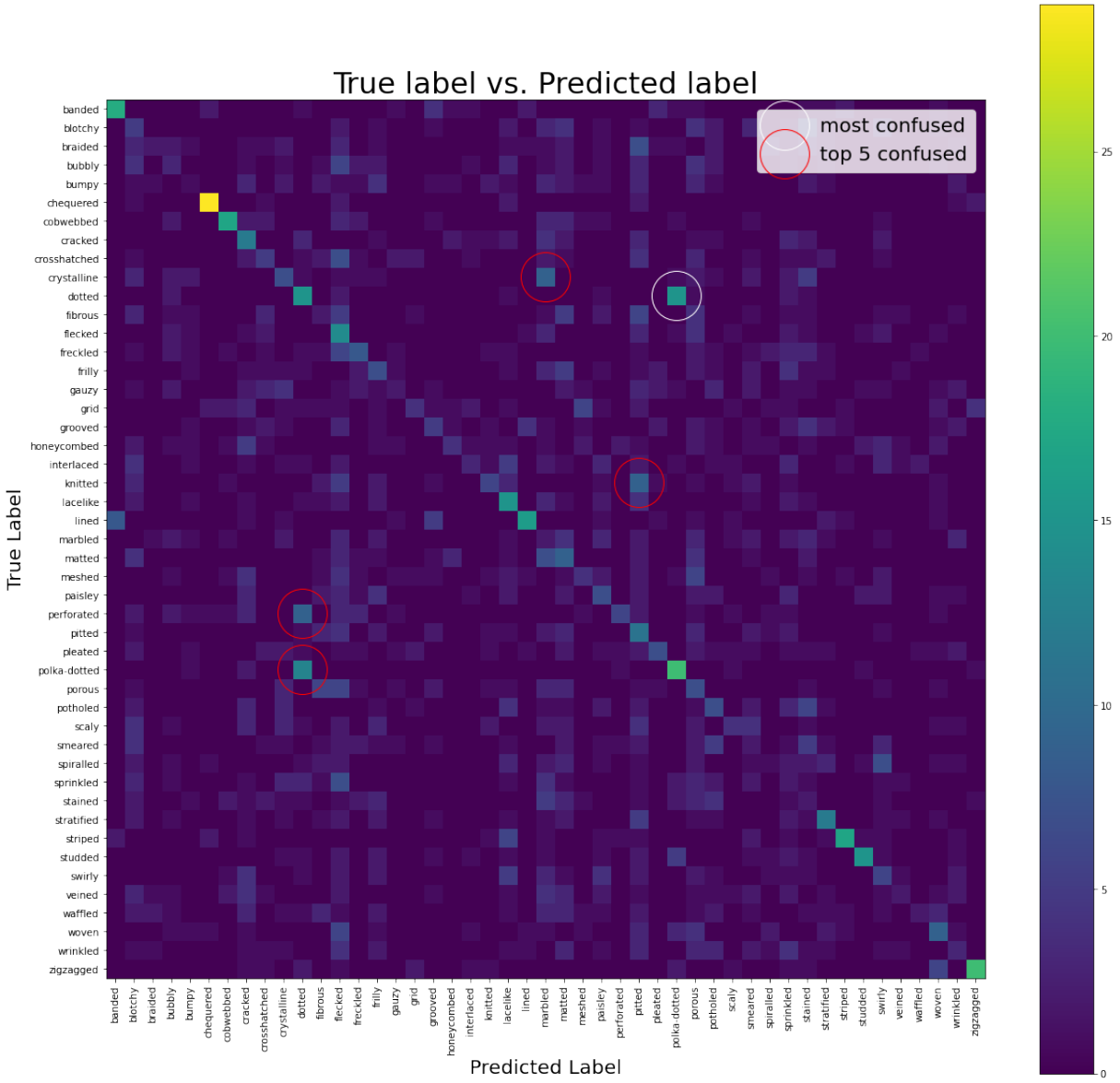

Figure 3: True label vs. predicted label. Most confused classification has been marked. The lighter the color is, the more occurrence of this case happened

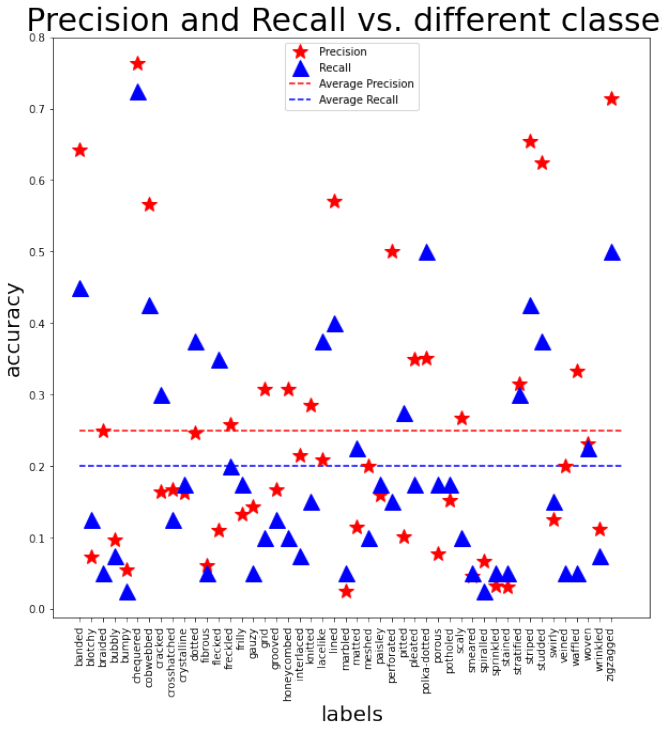

Figure 4: Precision and Recall for each classes.

Table 1: Averaged Confusion Matrix

True Positive: 8	False Negative: 32
False Positive: 32	True Negative: 1808

Figure 5: Confusion Table