Computer vision coursework 3: scene recognition

Iustina Ivanova, *email: ii1n17@soton.ac.uk,* David Guest, *email: dg4n17@soton.ac.uk*

***Abstract*—Image classification refers to the task of extracting information classes from a multiband raster image. Depending on the interaction between the analyst and the computer during classification, there are two types of classification: supervised and unsupervised. In this paper we described different types of classification algorithms.**

***Index Terms*—image classification, knn classifier, kmeans classifier, neural network.**

I. Introduction

For this coursework, we were tasked with implementing three separate classifiers for scene recognition. The first classifer created was a simple K-Nearest neighbour classifier; the second was a series of linear classifiers using a visual bag of words feature; and for the third we attempted to implement a self-trained Convolutional Neural Network. The data set consisted of grayscale images of no more than 70,000 pixels from the 15-scene category dataset. In all cases we were given a training set of 100 images for each of the 15 categories. The testing set consisted of the remaining 2985 images, but could not be used for any validation purposes, meaning that one of the main challenges we faced was a lack of availability of training data. The following sections outline the results we were able to achieve with the different classifiers.

A. K-nearest-neighbour classifier

For the first run of scene classification, we created a K-Nearest Neighbour classifier from scratch. The classifier was simple, storing the values of all training images, then evaluating the distance between each training image and the input test images. We used Manhattan distance as a metric for this, as using distance metrics which penalised outliers even further decreased the precision of our classifier. In order to decide upon which label should be assigned from the nearest images, a simple mode was used. The mode function implemented selects at random if there are equal numbers of closest neighbours from two or more classes. The coursework specification suggested that we crop the images to squares and resize them to a maximum of 256 pixels, while normalising them to have a mean of 0 and a length of 1. This modifications improved the average precision by 1-2 ppts on average.

In order to choose the best value of K for the classifier, we used a 5-fold validation approach to test the best K-value in the [1,100] range. We created 5 random train-test splits of 1200-300 from the training data set, The output of this can be seen in Figure 1. The line represents the average accuracy at that K-value. It can be seen that the highest average precision was achieved at K=24, therefore this was the value that was used for the test run, on which we would expect to maintain an accuracy of just under 20%. The output of a K-value in this range makes sense intuitively, as whilst lower numbers of K-values would normally be seen, the 15 separate categories makes it likely that low K values would end up randomly selecting between a large number of categories. Too high K values would risk drawing too wide a net and using images with a large distance to the test image to inform the category classification.

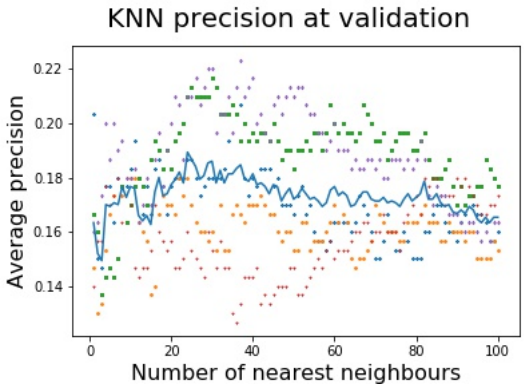

Figure 1. Average precision at validation using simple KNN classifier. Max precision achieved at K=24

In conclusion, the KNN classifier, while being a big improvement on random labelling, does not perform particularly well on the dataset - we believe that the distance metric is overly simple and will not correctly identify features, making it near impossible for the classifier to accurately distinguish between scenes with similar features, such as living rooms, bedrooms and offices.

B. A set of linear classifiers

In this part the images are represented by histograms based on a bag-of-visual-words feature.

Bag-of-visual-words works like a dictionary for the words. Basically, all the images are sliced to the pieces (patches) of the same size (*k*k*) and then from all this patches the algorithm chooses the most important. K-Means clustering is used for caricaturisation of all the patches (Figure 2).

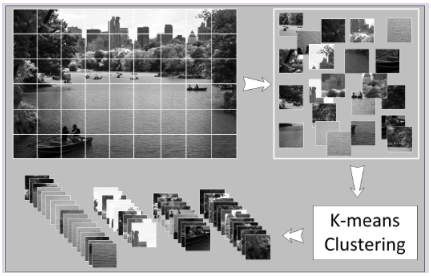

Figure 2. A grayscale image is broken down into small image patches which are then quantized into a number of visual words and the image is represented as a histogram of words[1].

There are different ways how the input images can be divided to patches. In this paper the random patches of the image is described. Random patches can work better then SIFT or just step-by-step choosing of patches.

All the patches should be assigned to 1 of N classes (N can be chosen by user). For this purpose K-Means classifier is implemented.

K-Means classifier algorithm:

In the clustering problem we are given a training set of patches $x_1, ..., x_m$ for each image, and the task is to group the data into a few cohesive 'clusters' (500 clusters). Here we are given feature vectors for each data point as usual; but no labels y_i (making this unsupervised learning problem). Our goal is to predict k centroids and a label c_i for each datapoint. The k-means clustering algorithm is as follows:

1. Initialize cluster centroids randomly:

$$\mu_1, \mu_2, ..., \mu_k \in \mathbb{R}^n \tag{1}$$

2. Repeat until convergence: (

For every *i*, set

$$c_i := arg\left(min_j \left\|x_i - \mu_j\right\|^2\right) \tag{2}$$

For each *j*, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\left\{c_i = j\right\} x_i}{\sum_{i=1}^{m} 1\left\{c_i = j\right\}} \tag{3}$$

)

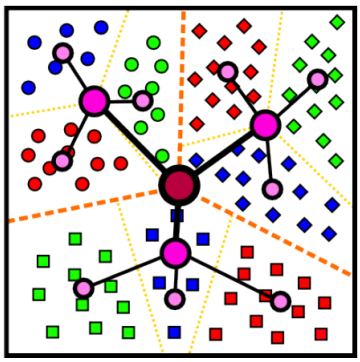

Figure 3. An example of *k*=3 means clustering. First sort the points into clusters and then recursively cluster each clustered set of points[3].

3After creation of N=500 clusters the algorithm creates histograms for all the pictures from training set. There are several ways to compute histograms, in this report the histogram was created from the 3100 different patches that were chosen randomly from the images. Each patch is 8*8 pixels and 3100 patches is the patches that algorithm has if it samples every 4 pixel. We also normalized and scaled all of the patches separately, so the patches themselves are represented more bright, each feature is more clear (figure 4). Random taking of patches sometimes can predict better that SIFT or just step-by-step taken patches. The parameter of clusters can be changed. We used python library *sklearn.cluster* for kmeans clusterisation. The parameter of maximum iteration was strict to 100.

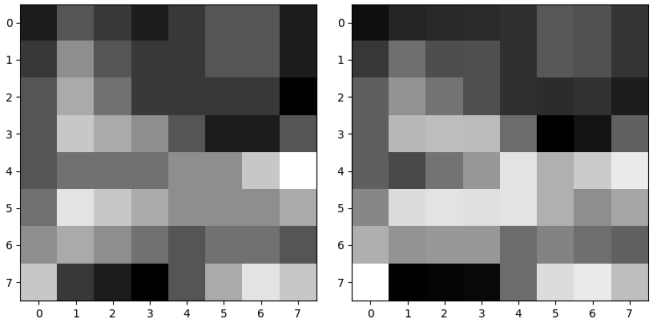

Figure 4. On the left side there is the random patch before normalization. On the right side there is the same patch after normalization. We can see features more clearly.

Next step is creating linear classifier. The algorithm creates input vector for each histograms of the images and the output vector is the name of result class. Training of the linear classifier is implemented with library *sklearn.svm*. The data is splited to training set (80%) and test set (20%). The accuracy of linear model after training is 56%.

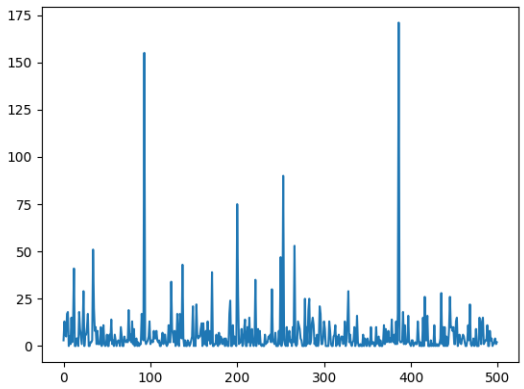

Figure 5. A histogram of 1 of the images with 500 clusters.

The last step of the algorithm is the prediction of images. The algorithm creates histograms for all the images from testing set folder. Linear model predict the output class for each histograms.

C. The best classifiers

One way of improvement of the second classifier is to create histograms for all the images for each class based on the amount of patches represented in the class. That means that we can compare the histogram of each training image summed together from each class with the histogram of each image from the testing set. The comparison in this can be made by cosine similarity:

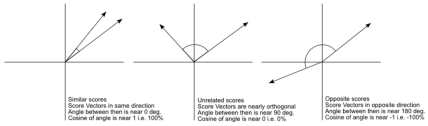

Figure 6. The cosine similarity shows how far each vector are from each other.

When the vectors have different sizes, the distance between them is the same, that means that we can measure the whole amount of histograms from the class with the histogram of 1 image.

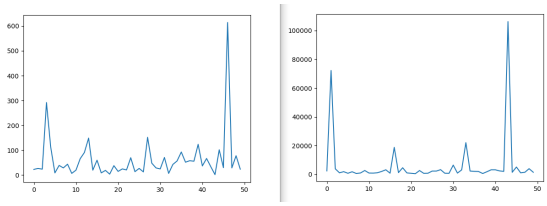

Figure 7. 2 histograms with the cosine distance=0.93. We can see the similarity, but the amount of patches in each class are different. The amount of clusters is 50.

We created program that gives the results with accuracy lower then the linear classifier.

Than we decided to improve this algorithm by scaling images from training set. The histogram representation remains the same in this case, at the same time the amount of training set improves. We had results worse than linear classifier.

D. Training a Convolutional Neural Network

To compensate for the lack of training data, we used some simple data augmentation techniques. Firstly, we altered the cropping used for earlier runs to crop the image at random. For each image, we then added the following augments: the image flipped along the vertical axis; the image put through a random small rotation; the image put through a random shift; and the image with a random scale change applied applied. After experimenting with rotation and seeing a fall in the validation accuracy, we realised that it did not make sense to train for large rotations in the context of the data set. For each batch of 2000 training iterations, we resampled the training image and re-ran, up to the total of 20,000 training iterations, meaning that the network had effectively seen 75,000 images over its training run. This pushed the average precision of the network at validation up to 40%

E. Further work to improve the classifier

The aim was to implement and train a convolutional network using the SimpleNet architecture

F. Statement of responsibilities

David completed part 1 and wrote the report for the same, Iustina completed part 2 and wrote the report for the same.

II. Conclusion

In this work we created the different image classificators, usupervised and supervised. In theory unsupervised algorithm gives better results, but for that purpose we need to use computers with high quality. One problem that we had was connected with administrator permission to ECS computer Lab.

References

[1] S. Banerji, A. Sinha and C. Liu, *A New Bag of Words LBP (BoWL) Descriptor for Scene Image Classification*, Department of Computer Scince, New Jersey Institute of Technology, USA: Newark.
[2] Harry Gifford, *Hierarchical k-Means for Unsupervised Learning*, Carnegie Mellon University.
[3] A. Coates and A. Y. Ng, *Learning Feature Representations with K-means*, Stanford University, Stanford CA 94306, USA.
[4] http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/
[5] https://arxiv.org/ftp/arxiv/papers/1608/1608.06037.pdf