COMP6223 Scene Recognition Report

Pin Wang: pw2a19 Xinyue Zhang: xz9a19
Zhengxi Shen: zs3g19 Daoyuan Lin: dl1u19

1 Introduction

This project is designed to study and explore image recognition. In this project, it uses tiny images and nearest neighbour classification to recognise scene. There are 100 images in each of the 15 scenes to train, and 2985 images to test. It also requires to implement three different classifiers against the testing images through labelled data image training and provide prediction of each image. The three different classifier used here are K-nearest-neighbour classifier,linear classiers using bag-of-visual-words with K-Means, and VGG19 framework based on deep learning. Moreover, the operating environment configuration are 64-bit python 3.7.5 for windows, Tensorflow 2.0, and CUDA10.0 and cuDNN7.6.5, respectively.

2 K-nearest-neighbour classifier

The “Tiny image” feature is one of the simplest possible image representations. It was presented in 2008 and the report shows that the tiny picture has only small resolution (32*32 in the report) but seem to contain most of the relevant information for the reliable recognition, and the performance only has a 30 percent decrease relative to full resolution[1]. So, “tiny image” can be a simple representation for the recognition work, and it is able to guarantee precision despite considerably decreasing the size. But the limitation is that it will discard all of the high frequency image content and not invariant to spatial or brightness shifts. In this assignment, the process of getting tiny image is that: we crop the image to make the height and width of image be divided by 16. And then, splitting the image to 16*16 rectangles and calculate the mean of the all pixels of ever rectangle and using this mean to represent the rectangle. At last, normalizing the new 16*16 image to have zero mean and unit length and picking it into a vector representing this image by concatenating each row.

2.1 Experiments

K-nearest neighbor algorithm (K-NN) is a non-parametric method that can be used for solving classification, regression, and search questions, and has no need to build a model and tune several parameters a. It assumes that similar things exist in close proximity, and the predicted output based on the distance between the training data and testing data. There is a disadvantage that it will be significantly slower when the feature dimensionality increases, because the classifier has no mechanism to learn which dimensions are irrelevant for the decision. In this assignment, the classifier is built and the first step is calculating the distances (Euclidean distance) between the training images and a testing image. And then, storing the distances and sorting them in a line. Furthermore, choosing the k training images that have smallest distances and counting the number of classes. At last, the class having the largest number will be class of testing image.

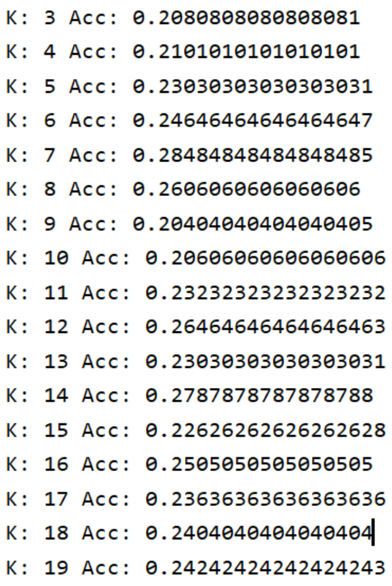
2.2 Results

After testing the k from 3 to 20, we choose k=7 and the precision is about 0.28 as shown in figure 1.

3 Linear classifiers

For this question, it is clear that linear classier, bag-of-visual-words and K-Means are required to be used.

Extracting every feature from an image is the basis in this task. As mentioned in the topic, the size of patches



```
K: 3 Acc: 0.2080808080808081
K: 4 Acc: 0.2101010101010101
K: 5 Acc: 0.23030303030303031
K: 6 Acc: 0.24646464646464647
K: 7 Acc: 0.28484848484848485
K: 8 Acc: 0.2606060606060606
K: 9 Acc: 0.20404040404040405
K: 10 Acc: 0.20606060606060606
K: 11 Acc: 0.23232323232323232
K: 12 Acc: 0.26464646464646463
K: 13 Acc: 0.23030303030303031
K: 14 Acc: 0.2787878787878788
K: 15 Acc: 0.22626262626262628
K: 16 Acc: 0.2505050505050505
K: 17 Acc: 0.23636363636363636
K: 18 Acc: 0.2404040404040404
K: 19 Acc: 0.24242424242424243
```

Figure 1: the accuracy of KNN classifiers with K

and the interval of each sample are recommended, 8 pixels and 4 pixels respectively. These patches need to be standard normalized with zero-mean and unit variance and mean-clustered with K-Means as images they belongs to, making K centroids and representing visual words. K-Means, these normalized images and visual words will be used in quantisation which is used to update bag-of-visual-words. Bagof-visual-words indicates frequencies of visual words on images. It will be normalized later to train the classier.

For images in the training set, they all would be read at the beginning to train the classier, and then go through and make predictions for images in the testing set. To know the average accuracy of the classier, we will divide images in the training set into the other training set and a validation set to go through. Images in each scene folder are out-of-order, hence we can randomly select images for both sets. And then train and validate the classier. Repeat the steps for ve time to compute the average accuracy. To know how does values for k inuence the accuracy, we set dierent values for k and train and test the classier by images from the training set. And then we can see the inuences to some extend.

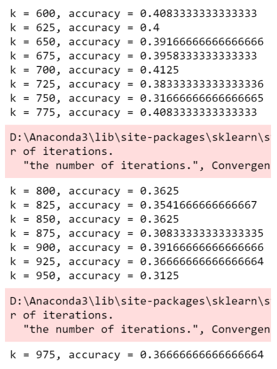
3.1 Experiments

As mentioned in the topic, the `OneVsRestClassifier` from `sklearn`, which is known as one-vs-all classier, was selected to be used. `Image` from `PIL` was used to read images into multi-dimensional arrays. `LinearSVC` was been chosen from three linear support machines in `sklearn` as its multiclass support is handled according to one-vs-the-rest schema. Compared to `KMeans`, `MiniBatchKMeans` is a better choices for processing `K-Means` with faster speed and almost the same accuracy.

3.2 Result and Tune

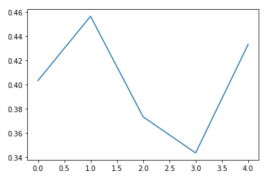
The result of testing successfully came out under the condition that k = 600. But it is a little bit disappointed that the accuracy of the classier validated by the training image was not high, only 40.833%. So We tried to tune the classier by loading dierent values for k, as shown in Figure 2(a).

As we can see that the accuracy for the classier was not high. In the tested range from 600 to 975 with interval of 25, when k = 700, the classier has the highest accuracy, 41.25%. Hence, we set 700 as the value for k for the program of run2. For the average accuracy, we just run the program under the condition that k = 700. The accuracy for each run (5 runs in totall) are listed below, as shown in Figure 2(b).



```
k = 600, accuracy = 0.4083333333333333
k = 625, accuracy = 0.4
k = 650, accuracy = 0.39166666666666666
k = 675, accuracy = 0.3958333333333333
k = 700, accuracy = 0.4125
k = 725, accuracy = 0.38333333333333336
k = 750, accuracy = 0.31666666666666665
k = 775, accuracy = 0.4083333333333333

D:\Anaconda3\lib\site-packages\sklearn\s
r of iterations.
  "the number of iterations.", Convergen

k = 800, accuracy = 0.3625
k = 825, accuracy = 0.3541666666666667
k = 850, accuracy = 0.3625
k = 875, accuracy = 0.30833333333333335
k = 900, accuracy = 0.39166666666666666
k = 925, accuracy = 0.36666666666666664
k = 950, accuracy = 0.3125

D:\Anaconda3\lib\site-packages\sklearn\s
r of iterations.
  "the number of iterations.", Convergen

k = 975, accuracy = 0.36666666666666664
```

(a) Test 1



(b) Official Test

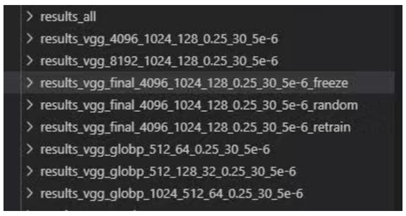
Figure 2: (a) Tune the classier by loading different values k and (b) accuracy of 5-fold cross-validation at k = 700.

The accuracy is not high, around 40%. Hence, we just tried another classier, OneVsOneClassier. This classier has a much higher accuracy for each run, see Figure onevsone. As OneVsOneClassier is the one-vs-one multiclass strategy instead of a one-vs-all multiclass strategy, we decided to continue to use OneVsRestClassier.

4 Developed classifier

4.1 Model

To extract features from 2D image, we use one of the state-of-the-art method VGG19[2] as the decoder. As a common starting point, we use the pre-trained VGG19 built on 2012 ILSVRC ImageNet dataset[3] by Keras. It a simple transfer learning technique. This can greatly improve the convergence speed and performance of the model. Since the task of original VGG model is classification on 1000 classes, we only use the decoder part which end up will the final convolutional layer (Conv). To fit our classification task on 15 different classes, we flat the feature generated from final Conv. Then appended 4 fully connected layers (FC) with 4096, 1024, 128, 15 nodes (decided by some experiments as shown in Figure 3).



```
> results_all
> results_vgg_4096_1024_128_0.25_30_5e-6
> results_vgg_8192_1024_128_0.25_30_5e-6
> results_vgg_final_4096_1024_128_0.25_30_5e-6_freeze
> results_vgg_final_4096_1024_128_0.25_30_5e-6_random
> results_vgg_final_4096_1024_128_0.25_30_5e-6_retrain
> results_vgg_globp_512_64_0.25_30_5e-6
> results_vgg_globp_512_128_32_0.25_30_5e-6
> results_vgg_globp_1024_512_64_0.25_30_5e-6
```

Figure 3: Use different fully connected layers and test the effect.

ReLU was used as activation function for first three FC layers. Besides, dropout was also added to these three layers to avoid over fitting due to the limitation of images for each class. At the end of the whole network, we applied a softmax function for classification. The detailed architecture is showing in Figure 4.

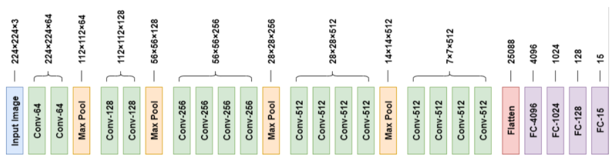


Figure 4: Modified VGG19. Conv-n is consists of a convolution layer with stride, and a ReLU activation function. Max Pool used kernel with stride. First three FC-n layers is consists of a fully connected layer with nodes, a ReLU function and a dropout. The final layer FC-15 includes a fully connected layer with nodes and a softmax function.

The loss function we used is Cross Entropy for multi-classes classification as following:

$$loss = -\sum_{i}^{c} y_i \log(f(x_i)) \tag{1}$$

where f is the whole network, and $f(x_i)$ is the predicated probability value for image x on lass i and $i \in C$ that C indicating the domain of 15 classes. y_i is the ground truth for the corresponding input image x on class i.

4.2 Experiments

There are 1500 2D grayscale images with different size in the training set, which consists of 15 classes, each class includes 100 images. We resize them all to to fit the model. Then zero-mean was applied to the image as a common image pre-processing method to speed up the model training. Since the pre-trained VGG only accept 2D RGB images, we simply converted the image to 3 channels by stacking the grayscale images. So that, 3 channels have same value for every pixel.Then conduct hyper-parameters setting. And we used pre-trained VGG trained on ImageNet for all the convolution layers, the model converged around 30 epochs. Since we used dropout for the FC layers to avoid overfitting which also increased the instability of the training progress, we finally trained the model on 50 epochs. Since the limitation of images for each class (100), we applied a large dropout rate 0.5 to avoid over fitting. The learning rate we selected was 5e-6 based on several experiments for avoiding vanishing gradient problem. The batch size is 20 due to the GPU Memory limitation.

4.3 Results

5-fold cross validation was used for testing different models on the 1500 training images, so that each fold contains 300 images. We used Run 2 results as baseline. As comparisons, we trained three different models. 1) The VGG19-scratch is a trained from random initialized weights. 2) The VGG19-freeze trained from the pre-trained VGG19 model, but all the encoder weights were freezed. 3) The VGG19-retrain was retrained the encoder to ask the model pay more attention on the training images we used. The evaluation method we used is over all accuracy. Since the VGG19-scratch without pre-trained model converged much slower, we increase the number of epochs to 200 to ensure the model fully converged. As shown in Table 1.

Table 1: Run3 Results

Method	Accuracy
Baseline	0.4010
VGG19-scratch	0.5160
VGG19-freeze	0.8760
VGG19-pre	0.9029

Obviously, VGG19-pre out performed other models. The VGG19 which trained on random initialized weights was overfitting due to the limited training samples even we applied dropout. So that, we then train the final model on whole training dataset which contains 1500 images. The final results for test data in the test folders is generated by this model. Since the checkpoint is large, we uploaded it to Google Drive: Checkpoint

5 Conclusion

In summary, three different classifiers were used to classify the training set for 15 scenes. It can be seen that the effect of classification using the K-nearest neighbor algorithm is not ideal. In the case of k = 7, the accuracy is only 0.27. Then we constructed a set of linear classifiers using visual word bags with K-means. The classification training accuracy rate of this sample has significantly improved. In the case of k=700, using 5-fold cross-validation, the average accuracy rate can reach 0.401. Finally, using the VGG19 framework based on deep learning, the classification accuracy of this sample is significantly improved. In the case of using the fully connected layer of 4096-1024-128-15 and retrained the pre-trained VGG19 model, by using 5-fold cross-validation, average accuracy can reach 90.3%.

During the experiment, we want to add the SENet framework to the VGG19 model, but since it will take a long time to unpack the VGG19 model and retrain the new model, so the work cannot be completed in a limited time.

6 Individual contribution

dl1u19: Independently complete the architecture and testing of the micro-image and nearest neighbor classifier models, and independently complete the report of run1.

zs3g19: Independently complete the architecture and testing of the linear classifier group model, and independently complete the run2 report.

pw2a19 & xz9a19: Jointly complete the architecture and testing of the VGG19 model based on deep learning and the fully connected layer, and cooperate to complete the report of the run3 part, and integrate all the reports into the final report.

References

[1] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, Nov 2008.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[3] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.