COMP6223 Coursework3: Scene Recognition

Xuan Li[1], Xiangying Wei[2], Xiao Feng[3] and Banghui Liu[4]

[1]*ID:32774508,* xl1u21@soton.ac.uk
[2]*ID:32788088* , xw11u21@soton.ac.uk
[3]*ID:32691106* , xf3u21@soton.ac.uk
[4]*ID:31882919* , bl1m20@soton.ac.uk

Abstract

This report illustrates our implementation details regard to three different Run. Run1: using ***TinyImage*** *features and* ***K-Nearest Neighbors classifier (KNNs)*** *classifier, achieve the accuracy of 0.25 in validation set. Run2: using* ***Bag of Visual Words(BoVWs)*** *to process 'upgraded'* ***TinyImage*** *and a set of* ***One-vs-All(OvA)*** *classifiers, achieve the accuracy of 0.42 in validation set. Run3: using* ***Scale-Invariant Feature Transform*** *features along with* ***BoVWs*** *and* ***Support Vector Machine(SVM)****, as well as* ***Image Augmentation****, achieve the accuracy of 0.81 in validation set.*

1. Run1

Using **TinyImage** features and **KNNs** to solve the problem. The best training and validation accuracy are 0.30 and 0.25 respectively, but some categories have 0 accuracies in validation set, indicating that the method to extract features is not powerful for some particular type of image.

1.1. Methods

1.1.1 Feature extractor

Using OpenCV package to read the grey image from folders, labelling each image with their corresponding folder name, stored as dict. Cropping each image to a square about the centre, before resizing them into a 16x16 image. Then flatten the image obtained from the last stage, and repeat the above procedures until every image in the folders is processed. Appending them to a list, which is subsequently transformed to a NumPy matrix, before feeding to the KNNs classifier.

1.1.2 Classifier

KNeighborsClassifier in scikit-learn package is used in this part of project as the classifier. The distances between input sample and each previous sample are calculated, the first **K** previous samples with the smallest distance are taken, and finally the classification with the most occurrences among these k samples is selected as the predict label of the input sample.

1.2. Experiments

In this section, we illustrate the training and tuning procedures of Run1, only one hyperparameter, the number of neighbors, is tuned in Run1. Range of changes of hyperparameter n_neighbour is from 1 to 100.

1.2.1 Training

In Run1, after feature extraction, we split the dataset into training and testing parts and feed them to a KNNs classifier with **K** nearest neighbours, which is considered to be a hyperparameter, with other parameters of KNNs classifier remaining default value. After the training phase, use an unseen test set to test its performance.

1.2.2 Tuning

According to figure 1, it is hardly to see any pattern from the validation accuracy, the accuracy line char fluctuating between 0.1 to around 0.27, which is witnessed in n_neighbour = 19.

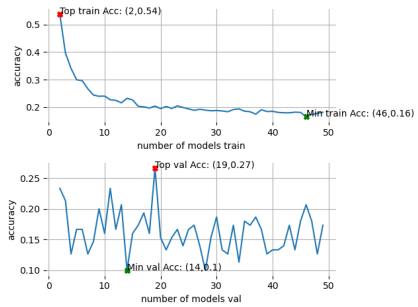


Figure 1. Accuracy-n_neighbors tuning result

2. Run2

Using **view_as_blocks** from skimage to acquire image blocks without overlap, since sliding window would decrease the accuracy of the OvA classifier that is utilized in Run2. **Visual Words** features are used as training features in this run. The training and validation accuracy are 0.58 and 0.42 respectively, with an ensemble of 15 classifiers, the overall accuracy is much better than Run1, and only 'livingroom' has 0 accuracy.

2.1. Methods

2.1.1 Feature extractor

The same image and label reading and storing approaches as illustrated in Run1 are also used in Run2. Except cropping the image, Run2 using **MinMaxScaler** to normalising images. After that, using **view_as_blocks** to split individual images into several non-overlapping 8x8 blocks. Then, sampling each block every 4 pixels in both x and y axis, to obtain a 4x4 NumPy matrix as training feature for KMeans cluster and the generation of codebook for visual words.

MiniBatchKMeans from scikit-learn is used as the KMeans cluster. After training KMeans and getting the trained cluster and cluster centres, which is a changeable hyperparameter, applying **applyingpairwise_distances_argmin_min** from scikit-learn to replace the cluster centres to a set of original training features. This is because some of the clustering centres may not be in the space defined by training features.

The phase of generating a codebook for visual words begins with creating an empty histogram with the same amount of indexes (X-axis) as the number of cluster centres for every individual image. Using **vq** from scipy, to calculate the set of closest indexes of cluster centres to training feature, i.e. which index in histogram should this particular feature belongs to. This phase will process all the features that belong to one single image to obtain its histogram. After repeating the extracting step for all training images, we can acquire 1500 histograms, the number of training images, which subsequently act as training features in the next phase.

2.1.2 Classifier

15 **LogisticRegression** are combined together as an individual classifier using **StackingClassifier**, cross-validation strategy is used in the training step to avoid over-fitting. The results of 15 one-vs-all classifiers will be judged by a **LogisticRegression** to produce the final prediction.

2.2. Experiments

2.2.1 Training

The standard training procedure is utilised in Run2, for both KMeans and One-vs-All model, leaving the works to the **fit** function in scikit-learn , which is available for both **StackingClassifier** and **MiniBatchKMeans**. Using $image\_size = 256$, $number\_of\_cluster\_centres = 500$, $15one - vs - allclassifiers$

2.2.2 Tuning

Experiments are running with 3 hyperparameters, but only one is changeable at each experiment, to reduce computation burden and time costs.

- img_size: [64, 128, 256] Candidate parameters are selected according to common image sizes.

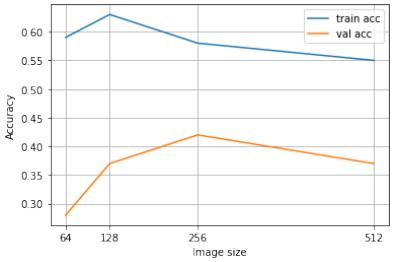


Figure 2. Accuracy-image_size tuning result

- n_clusters: [15, 400, 500, 600] Candidate parameters are selected according to the number around the given number(500), and the number of categories.

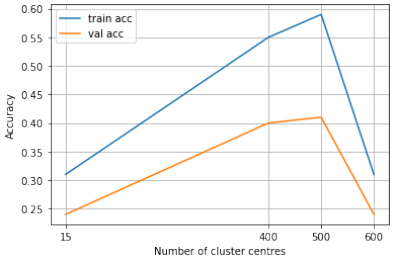


Figure 3. Accuracy-cluster centres tuning result

- n_models: [1, 5, 15, 30] Candidate parameters are selected according to the number of categories each classifier should be responsible for, from 15 to 3 and 1, eventually, 2 classifiers predict the same category, if things happen as expected.

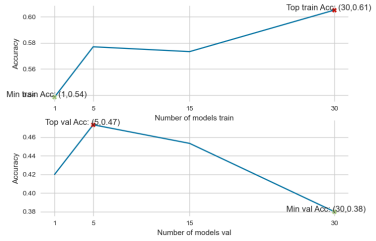


Figure 4. Accuracy-models tuning result(above:training set; below:validation set)

As image size gets larger, validation accuracy is increasing, reaching its peak at 256. However, the further increase would cause a decline in both training and validation accuracy, shown as figure 2.

According to figure 4, an ensemble of 30 classifier models has been best training accuracy, but also obtain the lowest validation result. An ensemble of 30 classifier models has the best validation accuracy. As the number of model increasing, training accuracy shows an upward trend, whereas validation accuracy drops continuously.

3. Run3

Scale-Invariant Feature Transform (SIFT) and **Support Vector Machine (SVM)** are used in this Run. Imporving the accuracy to 0.87 ± 0.02, with the assistance of **Image Augmentation**.

3.1. Methods

3.1.1 Image Augmentation

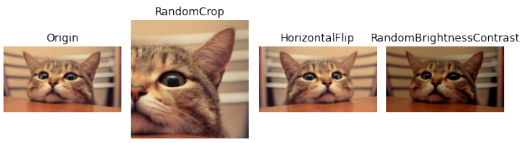


Figure 5. Image Augmentation Methods

Methods that are used in this project are shown in figure 5, including random crop, random brightness contrast and horizontal flip. This method can improve the generalisation performance of the model and prevent over-fitting, also expanded training datasets.

3.1.2 Feature extractor

We use the same method of reading pictures and labels as Run2 in Run3. Besides, to better detect the features of the image, we use **SIFT** algorithm in the OpenCV package to help extract the importance. Traditional feature matching algorithms often extract corners or edges directly so that they can not perform well in changing environments. However, SIFT can solve these problems. We had tried denseSIFT, but it is extremely slower than the regular SIFT.

Firstly, to detect the features of an image, SIFT scans it and identifies potential scale-invariant and rotation-invariant interest points. Secondly, the algorithm locates the feature points in scale space. Thirdly, in order to give the feature point value, it assigns one or more directions to each feature point location based on the local gradient directions of the image. At last, the algorithm expresses feature points with another description that can be detected in some environment changes such as camera angle and illumination.

3.1.3 Classifier

Support vector machine in scikit-learn package is used in this part. This algorithm compute the maximum interval for each class of data in the feature space and generate the best boundary of each class. For nonlinear classification problems, the kernel function can be selected to perform nonlinear transformation on the original data to achieve the purpose of classification.

3.2. Experiments

3.2.1 Training

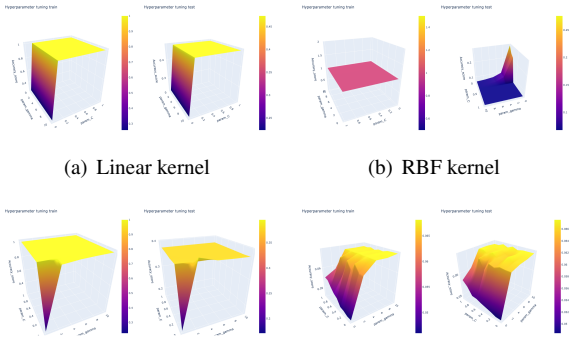
Using **Image Augmentation** to expand the dataset to three times its original size, i.e. each image is transformed into 4 different images using the combination of image augmentation methods. Each augmentation method has a different probability of application. Similar training procedures and strategies are used as shown in Run2. Extracting SIFT features from the image, feed them to a KMeans cluster to generate the visual words codebook, then use the codebook to obtain training features from the image, the histogram, and utilize the histogram to train an SVM.

3.2.2 Tuning

```
param_grid = {
        'kernel':['linear','rbf','sigmoid','poly'],
        'C':np.linspace(1e-8,1,9),
        'gamma':np.linspace(0.1,10,5)
    }
```

Figure 6. Hyperparameters of SVM

The hyperparameters for tuning the SVM is shown in figure 6.



(a) Linear kernel (b) RBF kernel

(c) Poly kernel (d) Sigmoid kernel

Figure 7. Accuracy for different hyperparameters(Interactive 3-dimensional diagram in Jupyter Notebook), for each subfigure, the left one is training results, the right one is validation results

According to figure 7, linear kernel, figure7(a), has the best and most consistent performance on both training and validation sets with most of the combination of hyperparameters and is more sensitive to the change of param_C.

Whereas RBF kernel, figure 7(b), experienced a serious over-fitting. Almost all classified correctly on the training set, but poor at validation set, with an only peak at under 0.25. Poly kernel, figure 7(c), has a similar pattern as RBF kernel on the training set, but its performance is balanced in different combinations of hyperparameters on the validation set, except for the smallest gamma and C. Sigmoid kernel, figure 7(d), hardly ever classified correctly on both training and validation sets. Best parameters for SVM are $C = 0.25, gamma = 0.1, kernel = linear$.

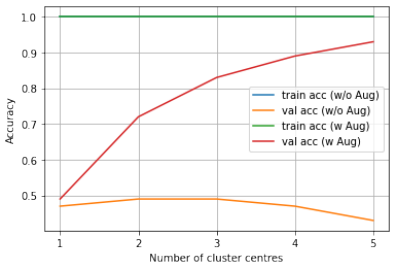


Figure 8. Accuracy of different times of image augmentation

As can be seen from figure8, with image augmentation(w Aug), validation accuracy has improved when increasing the image augmentation times, and the accuracy without augmentation (w/o Aug) continue to drop, we eventually choose $times = 4$, to avoid over-fitting.

4. Additional experiments

In addition to the main work, we also performed some additional experiments to test the performance of the algorithm in each Run. In the first experiment, we use 3 different classifiers in each RUN to test the performance of different feature extraction algorithms. In another experiment we compare the runtime consumption between Kmeans and Minibatch Kmeans.

4.1. Using three classifier in each Run

For each Run, we use three classifiers, **Support Vector Machine(SVM)**, **K-Nearest Neighbors(KNN)** and **Multilayer Perceptron(MLP)** to evaluate the performance of feature extraction in each Run. In order to control variables, the parameter settings of the same classifier in different Runs should be the same. The accuracy of each Run using the three classifers are shown in table 1.

	accuracy		
classifier	SVM	KNN	MLP
RUN1	34%	24%	25%
RUN2	3%	28%	47%
RUN3	83%	41%	83%

Table 1. The accuracy of 3 different classifiers in each Run(all with default hyperparamters)

In Run1, the performance of SVM is the best in three classifiers and gets 34% accuracy. In Run2, MLP reaches the highest accuracy. Note that in Run2 the SVM classifer makes the precision and F-score in some labels ill-defined and get a low accuracy. This issue is solved in Run3, all of the classifiers get better performance than Run1 and Run2. Therefore, the feature extractor in Run3 is the best in the coursework.

4.2. Kmeans vs Minibatch Kmeans

For two of the Runs using the clustering method, we use **Kmeans** and **Minibatch Kmeans** and compare the runtime consumption and prediction accuracy in each Run which are shown in table 2.

	Minibatch Kmeans		Kmeans	
RUN	runtime	accuracy	runtime	accuracy
RUN2	11.06s	42%	5230.29s	34%
RUN3	39.02s	81%	9844.08s	84%

Table 2. The accuracy of 3 different classifiers in each Run

Obviously, using Minibatch Kmeans can greatly reduce the time of clustering. Note that in Run2, the prediction accuracy using is improved by using Minibatch Kmeans instead of normal Kmeans. For Run3, though the accuracy of using Kmeans is higher than using Minibatch Kmeans, using Minibatch Kmeans greatly improves the efficiency of prediction.

5. Conclusions

The feature using raw pixels cannot obtain the semantic information from high-dimensional image data due to its working mechanism, resulting in its relatively low accuracy, though using BoVW can make good use of these features and improve its performance. The SIFT feature is a local feature of the image, which remains invariant to rotation, scaling, and brightness changes, and also maintains a certain degree of stability for viewing angle changes, affine transformations, and noise. This implies that it has relatively strong robustness. Its accuracy is stable at around 0.80, which is significantly overtaking the results of run1 and run2.

6. Division of work

For the average workload, Xuan Li did the SVM part of the work for Run3 and feature extraction part of Run2, and tuning parts for all Runs. Xiang ying wei did the SIFT part of Run3. Xiao Feng did the classifier part of Run2. Banghui Liu Complete Run1 alone. For the report, everyone completes the corresponding part and Xiao Feng wrote the part of Additional experiments.