COMP6223: Computer Vision Assignment 3: Scene Recognition

Name: Mohammed Ibrahim, Brian Formento
Email: mi3g15@soton.ac.uk, bf1g14@soton.ac.uk

Date: 12^{nd} December 2018

1 Run 1

The K Nearest Neighbour (KNN) classifier was implemented using python's opencv KNN functions for training and class prediction.

The training set was split such that 80 % of the images from each class in the original training set was used for training while the remaining 20 % from each class was used as a validation set (**note: same kind of set is also used for run 2**). The training process involves extracting the tiny image features from each image and passing them into a KNN classifier's training function. The tiny image features are obtained by re-sizing the original image into 16x16 using opencv's built-in functions and reshaping them into a 1x256 row vector.

The tuning process involved computing the classification accuracy of the classifier from the validation set for different values of **k** parameter from the KNN classifier as shown in Figure 1. **k** is the number of training points/feature vectors closest to a given feature vector from the test/validation set, where the most frequent class among those training points is assigned as its predicted class.

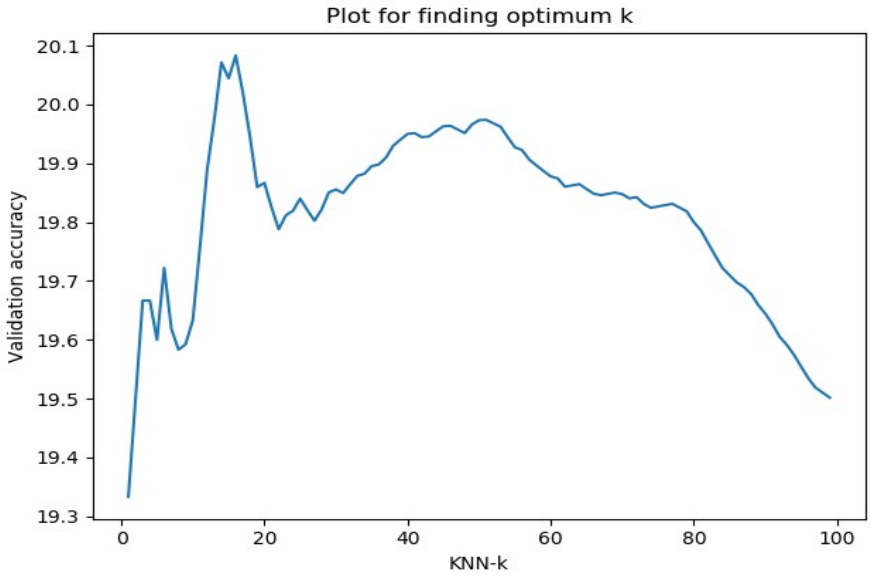

Figure 1: Plot for validation set accuracy for values of **k** from 1 to 100 to find its optimum value that gives the best accuracy. The optimum **k** is 16 for an accuracy of 20 %. This value of **k** was executed on the test set.

2 Run 2

A Support Vector Machine (SVM) was chosen as our classifier, where it can be configured as a set of linear classifiers by specifying its kernel parameter to be 'linear', while also specifying its decision function parameter to perform One vs reset (OvR) instead of One vs One in python's sklearn package implementation of SVM.

The training process involves first collecting a set of 10 uniform random samples from a full set of flattened image patches for each image (this is to speed up the k-Means clustering step while delivering roughly the same performance). The full set of flattened image patches are obtained for a patch window size of 8x8 pixels and a window step size of 4 pixels in the x and y directions. These samples for each image together are then passed into K-Means clustering to learn the visual words vocabulary which are represented as the K-Means centroids. This is followed with generating the Bag of Visual Words (BoVW) feature vectors for each image that is used as our training set for the SVM classifier.

During the tuning process, the validation accuracy was shown to vary for different runs of the code with the same settings for the feature extractor parameters as a result of passing uniform random samples for each image into K-Means, which lead to different visual words vocabulary for each run. This has also lead to other variations such as the validation accuracy for the case of patch window step size of 4 pixels being sometimes greater or lesser than that of 8 pixels (keeping number of centroids same) across various runs, when its meant to be greater consistently. Another kind of variation was observed in the validation accuracy across different number of centroids in one run, where the accuracy was shown to exhibit random variation with increase in number of centroids (see Figure 2) as opposed to its expected increasing behavior.

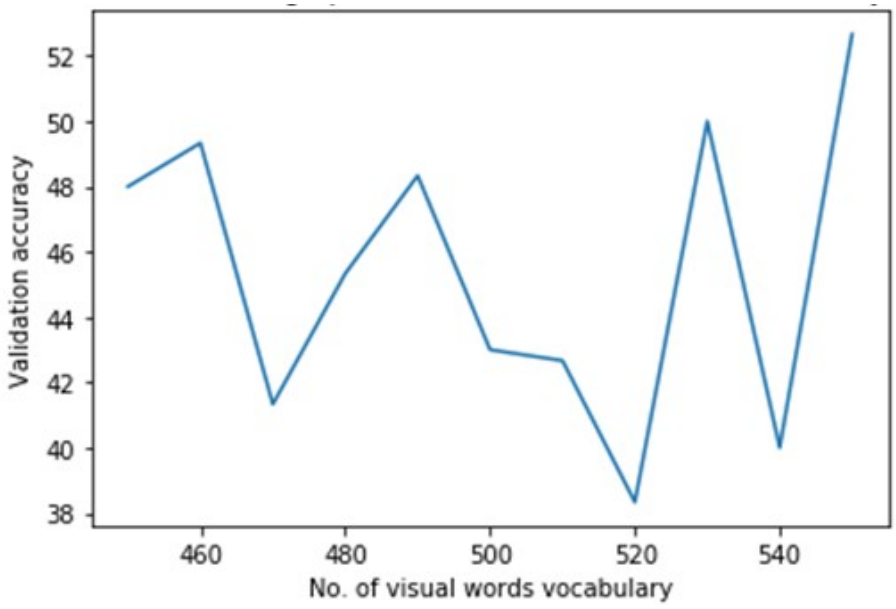

Figure 2: Variation in validation accuracy across increasing number of visual words vocabulary from 450 to 550 in steps of 10. The patch winodow step size of 8 pixels was used.

These kind of behaviors shown in the validation accuracy across runs, number of centroids and patch step size can be resolved by running on a sufficiently larger sized test or validation set as this removes the instability effects in the accuracy caused by random sampling. Therefore, the expected optimum values for the patch window step size and cluster number were chosen for training and testing which are 4 pixels and 500 clusters respectively.

3 Run 3

A modern deep learning method has been used for this task. More specifically, transfer learning with a support vector machine (SVM) classifier.

3.1 Training

The biggest challenge with this task is the lack of data. Ideally, for deep learning, it is widely accepted that the algorithm should have at least 1000 images per class. For serious applications, this should be in the millions. For example, Imagenet has around 14 million images between 1000 labels [1]. Transfer learning has been chosen as a feature extractor method because of this lack of data. Fundamentally this requires a pre-trained model. ResNet50 was chosen with the Imagenet weights as for this task it outperformed alternative networks, such as VGG16 and InceptionV3.

By importing the model and removing the last (classifier) layer it is possible to 'predict the images' and extract the feature vectors. In this case, each input image is automatically rescaled to a 224x224x3 size input, where, as ResnNet50 only accepts RGB images, each of the 3 channels is equal to one another. Inputting such image outputs a 7x7x2048 matrix. This is a lot of features and although it could be flattened to create a 100352 vector, it has been decided to use a 7x7 average pooling layer to reduce this matrix to a 1x1x2048 size and later flatten it to 1 dimension. This reduction allows for a faster computation.

The generator for every training datapoint outputs a 1x15 output label. This label is hot encoded, meaning it's a bitstring vector containing a 1 on the label index. A process of turning this vector to an integer (using np.argmax) and turning the integer to a string label is performed to attach a human-readable label to each data point. Together with the integer labels the feature vectors can be passed through an SVM, a powerful unsupervised learning algorithm which works by creating a vector that best segregates two classes, in this case, a one vs one approach has been used, where each class creates an SV against every other class essentially generating N(N-1)/2 classifiers.

It is important to note that a feature extractor doesn't understand what there is in a picture, but it's good at classifying between drastically different pictures. For example, if we do transfer learning on the forest dataset on ResNet50 it outputs the fountain label. Intuitively it makes sense when a grayscale image is parsed. However, a human could easily tell the image consisted of trees.

3.2 Tuning

To further improve the performance of the model data augmentation can be used. This is the process of tweaking the existing images to extend the current dataset. This includes shearing, rotating, horizontal flipping and zooming. It has been found that training with just a horizontal flip achieves the best performance.

The SVM classifier can be trained in two ways, one vs all or one vs one. The latter although being more computationally expensive as it requires N(N-1)/2 classifiers it al-

lows for stronger data segmentation. One VS all, on the other hand is quicker to train, as only N classifiers are required, yet it leads to less generalization. As computation time is not a concern the SVM has been trained using a one vs one approach. When training on 160 augmented samples per class and testing on 20 it achieves 80/85% accuracy.

Another important parameter to choose for the SVM is whether to have a linear or non-linear kernel. By definition, a non-linear kernel, when set up correctly as a minimum should perform as well as its linear counterpart. However when the number of features is higher than the number of observation points it is often the case that using a non-linear classifier does not benefit the computational cost. In this case both a linear and radial basis function have been tested, the latter showed no significant improvement.

When creating the labels for the test dataset pictures 1314, 2938 and 2962 are missing, so they have been excluded from the testing.

3.3 Conclusion

It is clear that transfer learning is a powerful feature vector extractor. However, it is also obvious that it struggles with certain types of data points. For example, upon closer inspection, the network struggles to correctly classify between house interiors. Out of the 20% error, approximately 10% came from this type of misclassification. This is not all that surprising, as these types of pictures are very similar in structure. If colour and more images were included, then we believe this model would correctly classify images of interiors.

References

[1] Image net statistics, Prof. Li Fei-Fei, PI, Stanford University Prof. Kai Li, co-PI, Princeton University http://www.image-net.org/about-stats

4 Contributions

The implementation, training and testing for run 1 and 2 were done by Mohammed Ibrahim, while the implementation, training and testing for run 3 was done by Brian Formento.