School of Electronics and Computer Science

Faculty of Engineering and Physical Sciences

University of Southampton

Scene Recognition

Danang Prawira Nugraha – dpn1n18@soton.co.uk

Aneeshaa S Chowdhry– asc1n18@soton.co.uk

COMP 6223 - Computer Vision

2018

Table of Contents

1 Abstract

In this Report, we will demonstrate the variation in performance of classifiers, when they use images as whole, patches or parts of images, and then pixel-level details of images.
Firstly, we attempt to classify images by using the entire image as a feature in a K-nearest neighbors Classifier. Then we study the performance of a classifier that uses more details in an image by constructing a Bag of Visual Words. Lastly, we observe rise in classification ability by a model that utilizes pixel level details on an image.

2 Introduction

Scene recognition is a problem when we need to identify the objects in the scene (object detection) and then being able to infer what's the scene about (scene recognition). Further sections in the Report document the different approaches taken to perform scene recognition, and the results observed.

3 Background

3.1 The Data

We are given two sets of images: Training Set, and the Test set. Images in both Datasets are grayscale images belonging to 15 classes that describe the scene: Bedroom, Coast, Forest, Highway, Industrial, Inside city, Kitchen, Livingroom, Mountain, Office, Open Country, Store, Street, Suburb, and Tall Building. The training data has 100 images per class, comprising 1500 images in total. The Test data has 2985 unlabeled images. After looking through most of the images in, we observe that:

1. Most of the images are taken in upright position.
2. There are images that can cause ambiguities between classes because it can contain similar features. For example, we have picture of cars in the street that can be classified as 'Highway' or 'Street'.
3. Main features in some images do not necessarily take up a large portion of the image; they may also have a different view-point. For example, there are images for mountain class that the mountain is in large scale (being up close) or in small scale (being far away). Another example could be a bedroom class image where the bed position is not in the center of the image, the bed is viewed from the side, or the bed is clipped in the side of the image.
4. Some images have darker illumination compared to the other images.

Based on our analysis of the data, we learn that the dataset can have different transformations applied the main features, e.g., change of illumination, different scale, view-point variation etc. Hence, we apply augmentations to images in our training set, to ensure our classifiers are robust, and generalize better. In addition, the ratio of samples between the given Training and Test set is almost 1:2, which further emphasizes our need for image augmentation. Image augmentation is the process of generating more images by applying transformations to an image.

4 Design

4.1 Classification using Tiny Image features

Each image in the training set is first shaped into square by cutting the larger dimension size, e.g. cutting the sides, if the image is in a wide rectangular shape. After that, the images are resized into a small resolution like 16x16, to form a 'Tiny Image'. They are flattened into a one-dimensional vector for each image to form a feature. Features are then standardized by mean-centering, and dividing by unit variance. We feed these features from the Training set to K-nearest-neighbors classifier. A new image is classified to a class of the neighbor, it resembles the most, or the neighbor it is closest to, in the feature space (Sonka, Hlavac, & Boyle, 2013). This technique will be used to classify images from the Test set.

From this setup, we can tune the performance of our classifier by changing the value of K in K-nearest-neighbor and the dimension of tiny image. For each combination of K and tiny image dimension, we will perform classification and evaluate the performance through n-fold cross validation process. The n-fold cross-validation works by partitioning the data into n groups. For each group, it will use the all the other groups to train the model and use the currently excluded from training group to evaluate the performance. This procedure is repeated for all n possible options for the hold out group (Bishop, 2006). We can choose the best parameters that produces the least cross validation error. In addition to this, we can also look at how using image augmentations will affect to our model accuracy.

4.2 Classification using Bag-of-Visual-Words

We extract 8 x 8 small image patches from an image. The image is densely sampled by moving the starting position of image patch extraction by 4 pixels in the direction of x and y from the top-left until the bottom right of the image like a sliding window. These image patches are then standardized and flattened into one-dimensional vector so that an image can have many 1-dimensional feature vectors. With these feature vectors, we build an aggregated vocabulary of image features (visual words) using K-means clustering. Given cluster size K, the algorithm will iteratively assign new images to the closest clusters, and recompute cluster centroids (Sonka, Hlavac, & Boyle, 2013). Using the features from training data to find K-means, we can use the centroids from the result as the representative features of our data.

Next, we create a histogram which count the occurrence of each cluster/visual word in each training image (bagging). These histograms from training data will be used to train our classifier which is using logistic regression in one-vs-all scheme. The sklearn's logistic regression classifier will create multiple binary classifiers for C classes to classify which class each histogram belongs to. Thus, to make predictions, we need to get the feature histogram for each test image, and feed it into our logistic regression classifier. For this setup, we can further tune performance by changing the cluster size K, and the dimension of image patch.

4.3 Classification using CNN

4.3.1 Why Conv Nets?

One of the many ways to perform image classification is to use a Convolutional Neural Network. We prefer a convolutional neural network is preferred over a regular Neural Network in this case, for the following reasons:

1. **Parameter sharing:** A sample image in our training set is 256x256 in size. If we have a neural network with 1000 units in the first layer, the number of parameters in the first layer alone would be = 256x256x1000 =6,55,36000(~65M).

However, if we used 6 filters, of size 5, the number parameters totally would be: 5x5x6 = 156. A Conv net is able to have fewer parameters is because they are shared. i.e., a feature detector used(ex: to detect vertical edges) in one part of the image is useful in another part of the image as well.

2. **Sparsity of connections:** Conv nets leverage the proximity of pixels, in a sense that two pixels close to each other are more related than two pixels that are far apart. This is unlike a regular neural network, where every pixel would be linked to every single neuron. But Conv nets, kill off these insignificant connections. Conv net structure also capture **Translation Invariance**, because a picture of a cat shifted a few pixels, is ensured to result in similar feature as the original image.

Despite the advantages of using a conv net, to build a new network architecture, we must empirically decide on the number of layers, filter sizes, and other network parameters. However, as we have a small data set, a more economical option would be to utilize **Transfer Learning**, instead of training a whole network from scratch. The 'Places' dataset is a collection of 10 million images comprising of 400 scene categories. A number of popular network architectures have been trained on large datasets, and the Places dataset in particular, to perform scene recognition. Although these images may be vastly different from our images, we can use what is already learned by the existing networks, to speed up learning process of our algorithm, because the initial layers will have already been trained to detect at least low level features such as edges/curves etc.

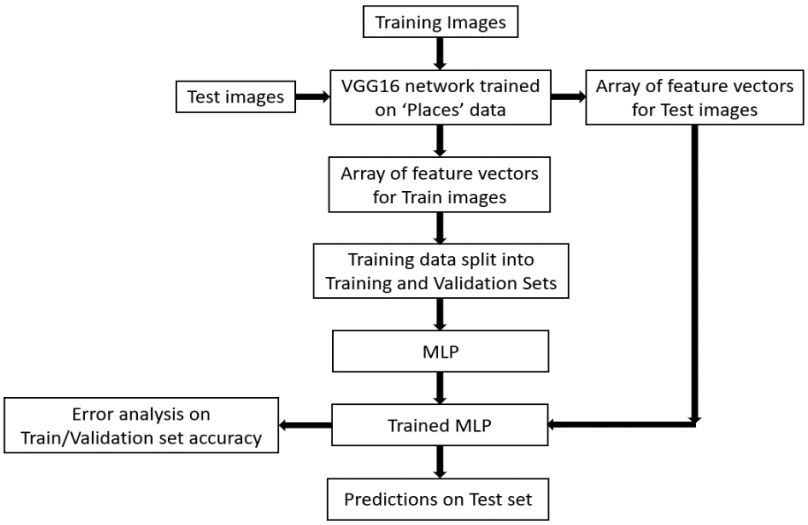
4.3.2 Choosing a Deep Learning Framework

Amongst the several deep learning frameworks available (TensorFlow, Caffe, Theano, Keras, etc.), we choose Keras. Keras, because of its simple interface for users which helps in rapid prototyping by creating neural networks that can work with TensorFlow backend.

4.3.3 Choosing a Network Architecture

We choose VGG16 for our experiment, over other existing networks such as Xception, ResNet50, InceptionV3, MobileNet, DenseNet etc. Although VGG16 is older, and more complex, we select this architecture merely due to familiarity, availability of the pre-trained model in Keras, and that the paper on VGG is easier to understand for new users.

4.3.4 Approach

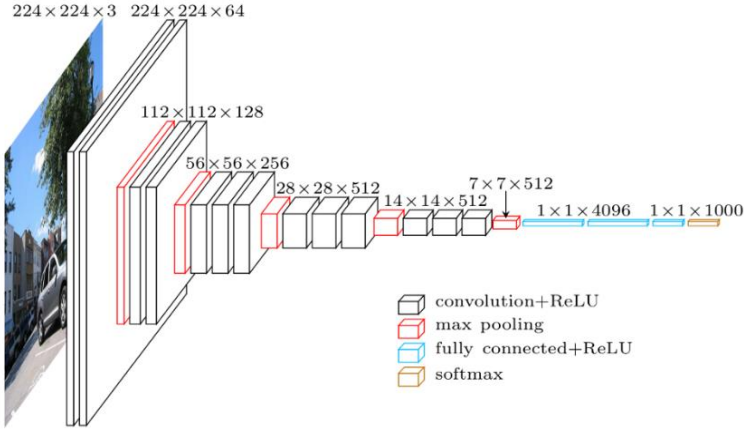


1. Training images, and test images are identically pre-processed(normalized). We have enabled data augmentation using the ImageDataGenerator method in Keras, for the training set.

However, the augmentation is performed real-time, while the images are being fed into a network for training. No transformation is applied to the Test set. Some of the augmentation techniques enabled in our experiment are: Rotation of image, Shearing transformation, Translation of image on different axis, and zooming inside an image.

2. Keras has a number of popular network architectures, and weights for different datasets, that can be imported for Transfer Learning/Fine tuning purposes. We choose the VGG16 network, pre-trained on the 'Places365' dataset(a subset of the Places dataset). The network was created by VGG (Visual Geometry Group, University of Oxford) for the ILSVRC-2014. The input to this conv net, is 224x224 RGB image. Hence, we reshape our grayscale images to be 224x224, and copy the same image content into all three channels, to be made suitable as input to this network.

3. The macro architecture of the network is shown below.



As we use this network for merely feature-extraction, and not prediction/classification, we collect the output of the max pooling layer highlighted in red, just before the fully connected layers. Size of the output array is 1500x7x7x512, where 1500 is the number of training images. Similarly, size of output test array is 2985x7x7x512, where 2985 is the number of test images.

4. To be able to evaluate our further steps for performance, we split the Training data into Training, and Validation sets. We now designate 1050 samples of our Training data as Training set, and the remaining 450 samples as Validation Set(30% of Total training images).

5. We are now faced with the task of classifying the feature vectors into 15 classes. To achieve this, we create a smaller Neural Network/MLP, whose structure is summarized with the help of Keras, is shown below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_60 (Dense)             (None, 256)               6422784
_________________________________________________________________
dropout_42 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_61 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_43 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_62 (Dense)             (None, 15)                1935
_________________________________________________________________
activation_19 (Activation)   (None, 15)                0
=================================================================
Total params: 6,457,615
Trainable params: 6,457,615
Non-trainable params: 0
_________________________________________________________________
```

Dropout layers have been included to regularize the network, eliminate unnecessary hidden units. Note that the last layer: SoftMax, has 15 output units, one for each class. Once again, we've used Keras, for quickly prototyping our MLP, in the interest of time and convenience.

6. We train this network on our Training data, measure performance on the Validation set, and then predict classes of each of our test images. The results are stored in file: run3.txt

4.3.6 Possible Issues

1. The large number of parameters created in MLP, could drain memory, computation power, and lead to overfitting on our training set.
2. We have a larger number of test samples, in comparison with the training set, and could be from a different source. Hence, our model could possibly perform poorly on the test set.
3. The VGG16 network was pre-trained on the Places365 data, to classify images into a 1000 classes. But we utilize the feature vectors generated in an MLP to classify input into one of the 15 classes. This may also lead to poorer performance/distinguishability within classes. For example, on manual analysis of a few test results, we observe that a number of 'bedroom' images were classified as 'Livingroom'.

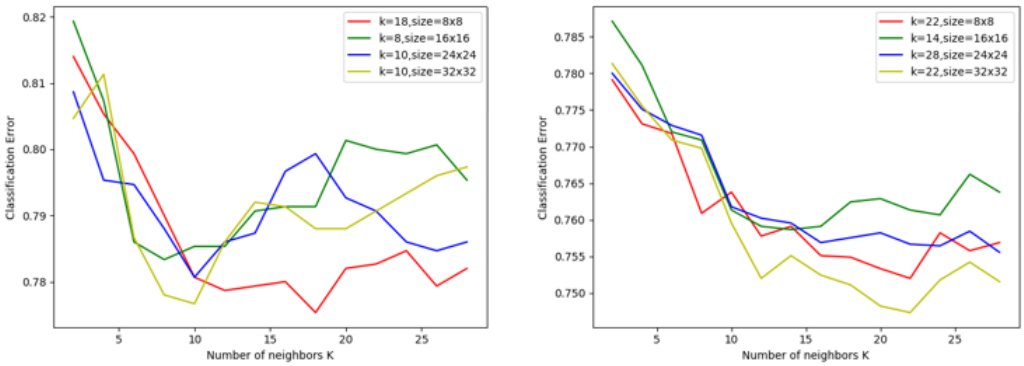
4.3.7 Further steps

- The MLP, can be modified in several ways to include different number of layers, hidden units, extent of drop out, no. of epochs, Train-Validation split, etc.
- Instead of an MLP, we could feed the feature vectors to a BoVW model like the one we used to generate run2.txt, or use any other non-linear classifier
- We can also do away with the Transfer learning idea, and extract Sift/Dense Sift/Phow Features from our training set. These can then be fed into a more familiar BoVW model used/described earlier or use Boosting algorithms such as XgBoost, AdaBoost, etc.

5 Results

5.1 Classification using Tiny Image feature

We are finding the best parameters for this setup by trying even numbered k from 2 – 30 and with tiny image size 8x8, 16x16, 24x24, and 32x32. The image augmentation performed is only tilting (rotation by a small degree) and adding some random gaussian noises. We evaluate the classification score through 10-fold cross validation.



(a) Without image augmentation (b) With image augmentation

FIGURE 5.1 CROSS VALIDATION RESULTS FOR FINDING PARAMETERS

Figure above shows the result of the finding parameters process. Each plot describes the tiny image size being using different k. The model got overall performance increase when image augmentation is used in training. We can get the best training accuracy by using k = 22 and 32x32 tiny image.

However, we want our classifier to be able to accurately predict unseen data instead of the training data. We decided to train our classifier using k = 14 and size = 16x16 to get the test data prediction result with validation set accuracy around 24%. This proves that our model is doing better than a random guess.

5.2 Classification using Bag-of-Visual-Words

We got around 45% validation set accuracy when we perform the classification using k = 500, and it increased into 49% when we set the number of cluster k into 700. It also increased by a slight margin into 50% when k is 1000. This comes at a cost, more clusters mean more processing time to recompute the centroids. Also, the more visual words in the vocabulary, the more our data becomes tailored to the training data as it is used to discriminate features between images based on the learned clusters.

5.3 Classification using CNN as feature extractor

Post completion of 30 epochs(iterations through data), accuracy on the Training set is 79.71%, and 76.44% on the Validation. Notice the rise in performance over previous classifiers. This improvement in performance can be attributed to the fact that we utilize pixel level details instead of parts of an image together, or the image as a whole.

6 Conclusion

Some scene recognition approaches have experimented with, and described in this report. We can infer that the tiny image features are simple features that don't adapt well to variations of image data. The densely sampled image patches can be sufficient for scene recognition but still inefficient since. The bag of visual words approach is very handy as it provides as an abstraction between features and the classifier in the form of clusters of image features, so that the classifier will not be tied to what image features being used. The transfer learning approach using a pre-trained CNN performs better than the previous approaches as it better utilizes details in an image to perform classification.

7 References

[1] Bishop, C. M. (2006). Patterns Recognition and Machine Learning. pringer-Verlag New York.

[2] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.

[3] Sonka, M., Hlavac, V., & Boyle, R. (2013). Image Processing, Analysis, and Machine Vision Fourth Edition . CEngage Learning.

[4] Very Deep Convolutional Networks for Large-Scale Image Recognition: Karen Simonyan, Andrew Zisserman

[5] Places: An Image Database for Deep Scene Understanding : Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, Aude Oliva

8 Contributions

Run1: Danang, Aneeshaa(for image reading/augmentation/tuning ideas)

Run2: Danang
Run3: Aneeshaa, Danang(for debugging)