Scene Recognition

Gavin Ayres
University of Southampton
Southampton, UK
gda1n21@soton.ac.uk

Amisha Singh
University of Southampton
Southampton, UK
as6u21@soton.ac.uk

Abiram Panchalingam
University of Southampton
Southampton, UK
ap18g21@soton.ac.uk

Reshma Abraham
University of Southampton
Southampton, UK
ra2n21@soton.ac.uk

Abstract

Image classification is a subject with a rich history. Most approaches can be separated into two distinct sub-classes of problems, feature extraction from images and multi-class classification in a high dimensional feature space. In this project we provide an empirical analysis of three approaches to this problem. We first evaluate the performance of a k-Nearest Neighbour classifier using features derived from a low-resolution image representation ('tiny image'). We show that the performance of this classifier can be superseded by an ensemble of 15 one-vs-all classifiers trained on a visual bag of words lexicon derived from raw image pixels. Finally, we show that a multi-class logistic regression classifier coupled with features extracted by spatial pyramid matching outperforms each of the aforementioned methods. [1]

1. Introduction

The task of image classification can be viewed as the confluence of two apparently unrelated challenges; extracting robust features from images and developing robust machine learning classifiers for high dimensional input. In this project we provide an empirical study of methods which address each of these tasks separately and how they can be combined effectively. The data set we used to train and evaluate each of our chosen approaches consists of 15,000 images with 15 distinct categories. The challenge we face is developing a robust combination of image features and machine learning classifier. In accordance with the project guidelines we have partitioned our empirical study into three distinct 'runs'. The details of each run are outlined in sections 2.1, 2.2 and 2.3 respectively, and are of increasing complexity.

[1] Each author contributed equally.

2. Methods

In this section we provide a detailed description of the algorithms used for feature extraction and classification.

2.1. Run 1

The first run consists of extracting features known as a 'tiny image' and then feeding these to a k-Nearest Neighbours classifier [7].

2.1.1 Tiny Image Features

The human visual system has a remarkable ability to recognise objects in images and to classify images into discrete categories, even at very low resolution. The aim of this run is to measure the performance of machine classification at low resolution. In particular we use an image resolution of size 16 by 16. The input to our classifier is then formed by flattening the image rows into a single vector per image. See [2] for further discussion on the tiny image feature.

2.1.2 k-Nearest Neighbours

The k-Nearest Neighbours algorithm [7] is a *memory based* algorithm. Given an input point, x_0 which we wish to classify, we find the k points in the training set X which are closest according to our chosen distance metric and then perform classification using majority vote among the k neighbours. As we are using a distance to perform classification, how the data is pre-processed has a significant effect on the performance of k nearest neighbours. In particular, our data should be standardised so that input features are not being measured in different units.

2.2. Run 2

For Run 2, we generated bag-of-visual-words (BoVW) features and classified those using an ensemble of 15 one-vs-rest classifiers.

2.2.1 Bag of Visual Words

Bag of visual words (BoVW) [6] model generates a histogram representation of image based on independent features. BoVW is analogous to the bag of words model used in natural language processing/information retrieval, images can be thought of as documents and its features as the analogue of words. The model has three main steps: feature detection, feature description and codebook generation.

Feature detection computes abstractions of a given image and makes local decisions at every image point about the presence of an image feature. The resulting features are a subset of the image domain, and often in the form of isolated points, called key points.

Feature description deals with representation of detected features in the form of numerical vectors. These vectors are called feature descriptors. Finally, codebook generation converts descriptors to codewords (analogous to words in text documents) that constitute the codebook (analogy to a word dictionary). A codeword is a representative of several similar feature descriptors. One of the methods for codebook generation is through k-means clustering over all the feature vectors. The centres of the learned clusters then make up the codewords. The codebook size is then equal to the number of the clusters.

2.2.2 Logistic Regression

Logistic regression is a classification algorithm that takes an input variable and models the probability of a discrete outcome. Similar to linear regression, logistic regression optimises weight parameters of the model over the training period to increase model accuracy. Logistic regression particularly differs from linear regression in terms of the output of the function; it outputs a real number between 0 and 1. A threshold is used to determine which class a data point belongs to. Logistic regression can be extended to multi-class classification problem by building a one-vs-all (OvA) logistic classifier. OvA uses a ensemble of binary classifiers each of which discriminates between a single class essentially building a multi-class classification model.

2.3. Run 3

For Run 3, we extracted Dense-SIFT features coupled with Spatial Pyramid Matching and classified them using the Logistic classifier implemented in Run 2 (Section 2.2.2).

Method	Average precision
Tiny image + k-Nearest Neighbours	0.21
BoVW + One-v-All Logistic Regression Ensemble	0.58
Spatial Pyramid Matching + Logistic Regression	0.72

Table 1. Average precision for each of the tested classifiers.

2.3.1 Dense-SIFT

Scale Invariant Feature Transform (SIFT) [5] is an algorithm for extracting features derived from scale-invariant keypoints of an image. The algorithm consists of four steps. Firstly, the difference of Gaussians (filter?) is computed for the image. Then image patches are searched for local maxima. These points which are the local maxima of the corresponding image patch are then considered candidate keypoints. In the proceeding step low-contrast keypoints and edge keypoints are removed. Next, the algorithm assigns an orientation to each keypoint. In the final step keypoint descriptors are generated. Dense SIFT [1] is a variation of this algorithm where a feature descriptor is extracted for densely sampled locations in the image, not just those initially identified by the original SIFT algorithm.

2.3.2 Spatial Pyramid Matching

From Run 2, it is evident that bag-of-visual words have an impressive level of performance for scene recognition. It is simple, efficient, and robust to occlusions, viewpoint changes and deformations. However, they completely ignore the spatial layout of the feature descriptors. Consequently disregarding the spatial arrangement of the features and any regularities in the image, both of which can be very powerful cues for scene classifications.

Thus, to leverage the position information of features we implemented spatial pyramids [4] by partitioning the image into increasingly smaller sub-regions and concatenating the histograms of local Dense-SIFT features extracted from each sub-region. This was then combined with a kernel-based pyramid matching scheme [3] that efficiently computes the corresponding sets of features in two images. The pyramid match kernel is simply the weighted sum of histograms of various levels. Although the spatial pyramid representation sacrifices geometric invariance, it compensates for this with improved discriminative power derived from the global spatial information.

3. Experiments

In this section we detail our experimental evaluation of the aforementioned methods. The data was partitioned such that 80% was used for training and 20% for testing. Furthermore, we set the model hyper-parameters using k-fold cross validation [7] on the training set.

3.1. Run 1

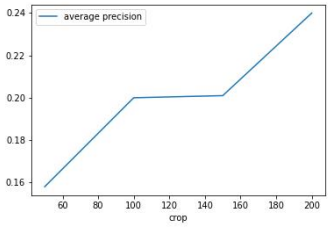

Figure 1. Average precision vs. cropped image size.

The first 'run' had several free parameters to tune in deciding on our final classifier. Firstly, in the construction of the 'tiny image' feature we first cropped our image. The image crop size we decided on was 200. Ablation studies were held where we kept the classifier constant and changed the crop size for the image. The results are illustrated in Fig. 1.

After image cropping, we resized the image to a low resolution (16×16) to produce our tiny image. This image is then flattened into a vector for input to our classifier.

We used a k-Nearest Neighbours classifier to then predict the image class for a given input feature. The precise choice for k was determined via grid search with 5 fold cross validation. The optimal number of neighbours was determined to be 8.

3.2. Run 2

For implementation of BoVW, the first step is feature detection. We extract patches of 8×8 with a step size of 4px from images resized to 256×256. Our descriptor algorithm then builds a list of feature descriptors based on the patches extracted. To build our vocabulary, we sample 500,000 feature descriptors from our list of $\sim$6mn. descriptors and generate a vocabulary of size 600 using KMeans clustering. The feature vectors are normalised before clustering so that each feature exists on the same scale. Finally, to generate the features for our classifier, the feature descriptors are quantised based on the codebook.

As the codewords are representative of groups of patches, the performance of the model is contingent on the codebook generated. To build the codebook, we experimented with different values of k within the range of 50 and 700. Fig. 3 shows a plot of change in average weighted sum of squared distance with cluster number. We chose 600 as the cluster number as within our range of candidates, the rate of change in average weighted sum of squared distance with cluster number (slope) was getting smaller around this value indicating a promising value for number of clusters.

We had 15 discrete outcomes (classes) as a target for our classifier, so we used an ensemble of 15 binary logistic regression classifiers. Each of these classifiers models one-vs-rest probability for a label.

The classifier takes quantised feature vector generated from BoVW model as input and outputs the probability of the image belonging to each of the discrete classes. We take the argmax of this output vector and assign it as the class label for the input image.

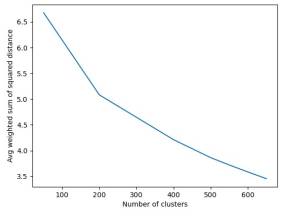

Figure 3. Cluster number vs. avg. weighted sum of squared distance.

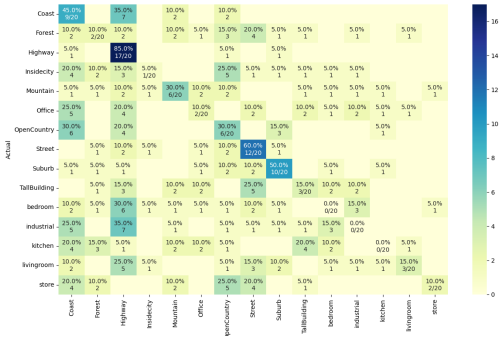

Figure 2. Confusion matrix of the results of run 1 (Section:2.1).

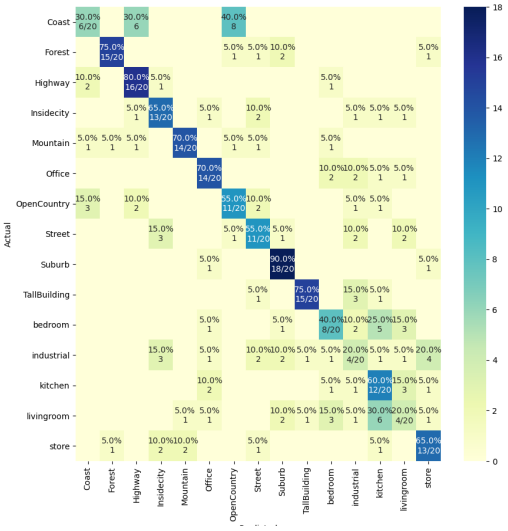

Figure 4. Confusion matrix of the results of run 2 (Section:2.2).

3.3. Run 3

For feature extraction, the key points were sampled with a step size of 4px and SIFT descriptors were then computed from these key points. The extracted descriptors were then clustered using KMeans clustering. We experimented with K values of 100 and 200 and chose the one that gave us higher accuracy, which was 200. For spatial pyramid matching we break the input image into L+1 levels. The performance of various combinations of L and step size are outlined in Fig. 2. L=3 reduced our accuracy which could possibly be due to the highest level of the L=3 pyramid being too finely partitioned, with individual bins yielding too few matches. A step size of 2px with L=2, gave us our final accuracy of 0.72%. For classification, we used a one-vs-all logistic regressor with a maximum iteration of 100 and l2 regulariser.

K	L	Step Size	Accuracy
100	2	4	0.62
		2	0.63
	3	4	0.61
		2	0.66
200	2	4	0.7
		2	0.72
	3	4	0.65
		2	0.66

Table 2. Run 3 experiment results.

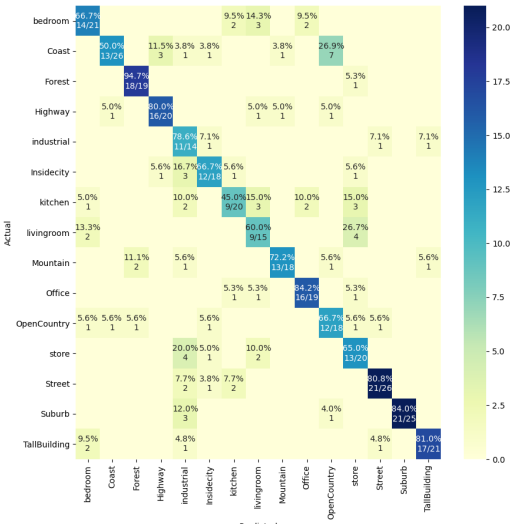

Figure 5. Confusion matrix of the results of run3 (Section: 2.3)

4. Conclusion

The task of image classification is an arduous one. It is contingent on appropriate selections for each stage of the classification pipeline; feature extraction from images and classification algorithm. In this project we studied three different combinations of approaches. We showed that relatively naive approaches to image feature extraction (the 'tiny image' feature) and to classification (nearest neighbours) do not perform well on the task. The issues with each of these approaches can be due to a lack of invariance to scaling and rotation in the image features and that nearest neighbour methods do not work well in high dimensions. We then showed that classification performance can be improved by a combination of bag of visual words features (computed from image pixels) with an ensemble of logistic regression classifiers. The bag of visual words features combats one drawback of the 'tiny image' feature, we can now extract high-level features using a clustering algorithm. However, more sophisticated feature extraction approaches can be used in combination with bag of visual words to produce more robust image features. Specifically, we showed that using dense SIFT features with spatial pyramid matching as the feature extraction technique from which to construct our bag of visual words superseded the performance of previous approaches.

On reflection, this project highlighted the importance of image representation for computer vision. The classification score can be thought of as a measure of how easy it is for a machine learning algorithm to distinguish between images. It forces one to consider what parts of an image truly represent the label one assigns to it and which of these parts would be likely to appear in every such image. Upon completion of this project we have not only gained an appreciation of the difficulty in construction appropriate image representations but also an appreciation for the sophistication of the human visual system.

References

[1] Xavier Munoz Anna Bosch; Andrew Zisserman. In *Proc. 9th European Conference on Computer Vision (ECCV'06)*.

[2] William T. Freeman Antonio Torralba, Rob Fergus. Tiny images, 2007.

[3] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1458–1465. IEEE, 2005.

[4] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[5] David Lowe. Distinctive image features from scale-invariant key points, 2004.

[6] Sivic and Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, 2003.

[7] Robert Tibshirani Trevor Hastie, Jerome Friedman. *The Elements of Statistical Learning*. 2001.

5. Appendix

Filter	Classifier	Hyperparameter	Validation Accuracy (20%)	Running Time (s)
AutoColorCorrelogram Filter	SVM	C=1	0.24	0.87
		C=5	0.27	1.44
		C=10	0.2633	1.98
	Naïve-Bayes	Default	N/A	N/A
	MLP	Hidden Layers = 3, LR = 0.3, Momentum = 0.2	0.2633	35.36
		Hidden Layers = 5, LR = 0.3, Momentum = 0.2	0.2667	54.83
	Random Forest	Default	0.27	4.77
Binary Patterns Pyramid Filter	SVM	C=1	0.5866	1.32
		C=5	0.59	1.45
		C=10	0.59	1.26
	Naïve-Bayes	Default	0.5333	0.43
	MLP	Hidden Layers = 3, LR = 0.3, Momentum = 0.2	0.0966	24.26
		Hidden Layers = 5, LR = 0.3, Momentum = 0.2	0.13	39.55
	Random Forest	Default	0.5867	1.58
Edge Histogram Filter	SVM	C=1	0.59	0.9
		C=5	0.5333	0.63
		C=10	0.5166	0.77
	Naïve-Bayes	Default	0.5333	0.08
	MLP	Hidden Layers = 3, LR = 0.3, Momentum = 0.2	0.29	3.18
		Hidden Layers = 5, LR = 0.3, Momentum = 0.2	0.3966	4.72
	Random Forest	Default	0.5766	0.67
Fuzzy Colour and Texture Histogram (FCTH)	SVM	C=1	0.4133	0.71
		C=5	0.4333	0.44
		C=10	0.4166	0.77
	Naïve-Bayes	Default	N/A	N/A
	MLP	Hidden Layers = 3, LR = 0.3, Momentum = 0.2	0.3066	6.41
		Hidden Layers = 5, LR = 0.3, Momentum = 0.2	0.2833	9.93
	Random Forest	Default	0.3466	1.87
Gabor Filter	SVM	C=1	0.18	0.55
		C=5	0.18	0.32
		C=10	0.1766	0.44
	Naïve-Bayes	Default	0.22	0.05
	MLP	Hidden Layers = 3, LR = 0.3, Momentum = 0.2	0.1933	2.58
		Hidden Layers = 5, LR = 0.3, Momentum = 0.2	0.1866	3.91
	Random Forest	Default	0.1533	0.98
Jpeg Coefficient Filter	SVM	C=1	0.6733	0.45
		C=5	0.6666	0.61
		C=10	0.6666	0.84
	Naïve-Bayes	Default	0.5166	0.1
	MLP	Hidden Layers = 3, LR = 0.3, Momentum = 0.2	0.3266	6.37
		Hidden Layers = 5, LR = 0.3, Momentum = 0.2	0.4766	9.93
	Random Forest	Default	0.6266	0.82
Pyramid Histogram of Oriented Gradients (PHOG)	SVM	C=1	0.6333	1.11
		C=5	0.6366	1.13
		C=10	0.6366	1.19
	Naïve-Bayes	Default	0.53	0.37
	MLP	Hidden Layers = 3, LR = 0.3, Momentum = 0.2	0.2666	21.13
		Hidden Layers = 5, LR = 0.3, Momentum = 0.2	0.3533	33.56
	Random Forest	Default	0.58	1.34

Figure 6. Various features and classifiers tested