University of Southampton

Scott Williams

Hannah Billingsley-Dadd

COMP3204

Scene Recognition

15 December 2017

## Run 1

This run was an attempt to implement a K-Nearest Neighbours algorithm to classify images after a training image set was processed. Training images were cropped to a square about the centre of the image and resized to a small icon. Then we converted all the training images to a vector of floats, each float representing the brightness of a pixel, and plotted these vectors onto a graph (abstracted as a List).

To classify an image, the image is centred and resized and again converted into a vector of floats. The distance was then calculated from this testing image vector to all training image vectors. The nearest vectors to the testing image vector were then found and the output classification of the testing image is the most common label of the nearest vectors.

We experimented with the values of K, the number of nearest neighbours the algorithm looks at, and R, the number of pixels we resize the image to. Our experiments revealed that a K of 16, one more than the number of classes, and an R of 8 seemed to yield the best results. However, there is a lot of inconsistency and randomness to the results. We achieved an accuracy of approximately 15 to 25% which hopefully is also observed in the prediction file.

## Run 2

This run was an attempt to develop a set of linear classifiers using a bag-of-visual-words feature based on fixed size densely-sampled pixel patches. Each image would be interpreted as the histograms of the words found in the patches and the most common word would be used to classify the image.

The hard assigner, used by the lib linear analyser to assign features to identifiers, is trained using the training data set. Patches of a specified width, height and width apart are created for the images and flattened into a vector. These vectors are then quantised using the K-Means algorithm to learn the vocabulary.

Parameters that were configured for this algorithm are the patch width and height and how often the patches are taken. Further to this the number of clusters used by the K-Means algorithm can be varied for this algorithm to alter the accuracy-performance trade off.

## Run 3

Several changes were made to the classifier to make it more accurate than Run 1 and 2. The first of which is that SIFT (Scale invariant Feature transform) extractor was used to extract features from the images. Although we looked at multiple extractors, we chose to use SIFT because it had an increased accuracy as compared to Dense SIFT, even if it took longer to run. SIFT is largely recognised as a quick feature extractor (Andrea Vedaldi, 2010).

SIFT uses the difference of gaussians to remove edges and corners from the image, then local descriptors are extracted by using patches, similar to the algorithm used in run 2. A binary search tree is traversed to find the best matches which are returned to the SIFT as a priority queue. Finally, using a RANSAC algorithm the matches are verified and those which are deemed correct are returned. The algorithm is not affected by rotation or scale of the image making it ideal for classifying the testing dataset (Lowe, 1999).

The SIFT extractor has been implemented in the code using DoGSIFTEngine in the OpenImaj library. It has been used in the hard assigner to

A further way to improve accuracy is to use naive bayes classifier. Naïve bayes

While tuning the parameters the hard assigner was cached to increase the performance of the algorithm. To further make the algorithm faster naïve bayes used the mode ‘MAXIMUM\_LIKELIHOOD’, where only the single most likely annotation is returned. This is because we are only concerned with annotating the images with one string and it makes the algorithm faster to run.

Our run 3 has an accuracy of roughly 35% and takes around 20 minutes to run – a good trade-off between execution time and accuracy.

## Contributions

Both team members contributed an even amount of work to this project. Writing the runs started as an individual endeavour, with one of us starting off the program but then joining with the other to pair program the remainder. As a pair, problems were solved quicker.

## References

Andrea Vedaldi, B. F. (2010). Vlfeat: an open and portable library of computer vision algorithms. *MM '10 Proceedings of the 18th ACM international conference on Multimedia* (pp. 1469-1472 ). Firenze: ACM.

Lowe, D. (1999). Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on.* Kerkyra: IEEE.