Visual Search of an Image Collection

Name: Nithesh Koneswaran

Username: nk00374

Student ID :6474079

Contents

[Setting up the coursework 2](#_Toc23681503)

[Global colour Histogram 2](#_Toc23681504)

[Object classification using SVM 3](#_Toc23681505)

[References 3](#_Toc23681506)

# Requirement 1: Global colour Histogram

First, we must reduce the number of colours in our image, this is called “Quantization”. The aim is to translate boxes or bands of pixel intensities to one-pixel intensity effectively reducing the number of colours in the image.[[1]](#footnote-1) I will be experimenting and testing the different levels of RGB quantization.

I experimented with ‘q’ values of 2,4,8 and 16 where ‘q’ is used to normalise RGB values within the range [0, q-1]. When ‘q’ is a small integer, the RGB colour values are mapped to a small range. However, when ‘q’ is a large integer the RGB values will be mapped to a larger range. I hypothesis that at smaller values of ‘q’, the visual search system will perform poorly since most of the images will produce similar histogram values. At large values of ‘q’ I expect the precision to decrease since the granularity of the images will be more fine-grained making it difficult for the visual system to identify correctly similar results since the histogram values of the images will be quite varied.

The results from Figure 1 shows the average precision for each class at different RGB levels of quantization. At class 8, where the level of ‘q’ is 16, the average precision was the highest among the other levels of ‘q’ for that class, but at class 20 the average precision was the lowest. Class 8 held images of bicycles containing a higher range of colours than class 20. Class 20 held images of the ocean and had far less range of colours than class 8. At level 16 (where ‘q’ = 16) we get a higher granularity of the image suggesting that quantization of high values tend to make the system perform better at searching for fine-grain images compared to low-grain images and vice versa.

The chart below shows the Mean Average precision, which considers the average precision of all the classes, against the different levels of quantization. The system performed the best when the level of quantization was 8. However, the precision for the different levels of ‘q’ were all quite similar. I found that increasing ‘q’, increased the execution time of the program due to an increase in size of the feature descriptor. Therefore, it would be better to use a small value of q rather than a large value if hardware was taken into consideration. Although level 8 performed the best, we do have classes which performed much worse than the other classes (look at figure 1).

Overall, the performance of our visual system is poor, and this is because of the drawbacks of using a global colour histogram as a feature descriptor. One of the drawbacks of the descriptor is that it does not consider any objects and shapes in the image, if we want to return similar images, we need a descriptor that can encode shapes as well as sparse features which we can then use to calculate the similarities.[[2]](#footnote-2) Lastly images that are clearly different but have the same exact/or similar histogram values may be flagged as similar and returned by the visual system. This is because the global histogram descriptor does not consider the location of the pixel meaning that images with the same colour composition but with different pixel locations will be considered similar.

I have researched further and looked at MATLAB’s available functions I found that there are two methods of ‘Quantization’ these are; ‘Uniform Quantization’ and ‘Minimum Variance Quantization’.[[3]](#footnote-3) As the name suggests, ‘Uniform quantization’ evenly divides the colour space into boxes or bands (this is what we have been doing above). If a pixel in an image falls within the boxes, then that pixel takes on the centre/middle value of the band. In ‘Minimum Variance Quantization’ the colour space is divided depending on the distribution of colours in the image. Minimum Variance Quantization will perform better since using Uniform Quantization will end up making images much similar due to its uniform nature. As a result, by using Minimum Variance Quantization we should expect the system to achieve better results since it will allocate more of the frequently appearing colours and less of the less occurring colours.

# Requirement 2: Evaluation Methodology

For evaluating the visual search system, I created an algorithm that will randomly pick 1 image from each class and calculate the precision and recall. At the end the results from each query will be combined to calculate the overall statistic. In order to evaluate our system, I had to create a ground truth. For this I used the file names since the first part of the name contains the index of the class they belong too. When we query for an image, any images retrieved that are not of the same class of the query image will be considered irrelevant. This information will be used to calculate the precision and recall.

# Object classification using SVM

In this section I will be experimenting with SVM and CNN. For the feature descriptor I will be using Alex Net to extract features from the images through its series of convolutional layers. For this section I had to install MATLAB’s Deep learning and Alex Net add-on installations to complete this task. At the start of the network we can find low-level representations of the images, but as we move deeper into the network, we find more higher representations of the images. We can then extract these features by sampling the activations from a fully connected layer into a vector and then simply perform an SVM to classify an object.

The accuracy that we got from the SVM averaged to be 50%. I believe that the results could be improved if we increased the volume of training images, that way the network would be able to generalise the classes much more. The results were probably affected due to the similarities of the classes, again if we had more data the network would have been able to generalise the classes more. We could also Fine-tune the network using Transfer learning which essentially allows the network to learn a new task.[[4]](#footnote-4) The confusion matrix below suggests which classes the neural network worked well with.

Measures could be taken to improve our accuracy further by preparing and augmenting images. Image could be sampled and cropped, the aim is to create variations of the images through a range of transformations allowing the neural network to generalise the data better and prevents it from over-fitting.[[5]](#footnote-5) Looking at examples of deep learning networks, I found that most use the technique of subtracting the mean for each image channel or dividing the feature by its standard deviation before exposing the data to the network. This causes the values of the inputs to be similar helps the neural network to learn more efficiently.

<Insert Confusion matrix>

Instead of a multiclass SVM, I will be creating a binary classifier classifying whether an object is a bike or a

Overall completing this task took around 10hrs. Most of my time was spent trying to set the code up so that it can load the images along with the associated labels. I found that I had to manually pick the test images and label. I struggled to find a systematic approach for this since the number of images in each class varies, therefore a simple for loop to pick 1 or 2 images after every N number of images would not get us an evenly distributed class image. The next thing I struggled with was trying to convert my dataset which contained the images in a cell format of 591x1 each containing 227x227x3 (resized images) to a numerical matrix so that I could feed the dataset into the SVM, without realising I needed to do the feature extraction first. I later found out the function augmentedImageDatastore could be used to resize the images and also be used to get the activations from the fully connected layer and then be fed into the SVM.[[6]](#footnote-6) I will use this technique again to extract features to be used in our visual search system.

# References

[1] tarkmamdough.wordpress “Image Retrieval: Global and Local Color Histogram” [Online]. Available at <https://tarekmamdouh.wordpress.com/2013/08/12/global-and-local-color-histogram/> [Accessed: 26/10/2019]

[2] mathsworks.com “reduce the number of colors in an image” [Online]. Available at <https://uk.mathworks.com/help/images/reduce-the-number-of-colors-in-an-image.html> [Accessed: 26/10/2019]

[] mathsworks.com “Transfer Learning Using AlexNet” [Online]. Available at <https://uk.mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html>  
[Accessed: 14/11/2019]

[] mathsworks.com “alexnet” [Online]. Available at <https://uk.mathworks.com/help/deeplearning/ref/alexnet.html> [Accessed: 14/11/2019]

[] machinelearningmastery.com “Best Practices for Preparing and Augmenting Image Data for CNNs” [Online]. Available at <https://machinelearningmastery.com/best-practices-for-preparing-and-augmenting-image-data-for-convolutional-neural-networks/> [Accessed: 14/11/2019]

1. tarkmamdough.wordpress “Image Retrieval: Global and Local Color Histogram” [Online]. Available at <https://tarekmamdouh.wordpress.com/2013/08/12/global-and-local-color-histogram/> [Accessed: 26/10/2019] [↑](#footnote-ref-1)
2. Pyimagesearch.com “Clever Girl: A Guide to Utilizing Color Histograms for Computer Vision and Image Search Engines” [Online]. Available at <https://www.pyimagesearch.com/2014/01/22/clever-girl-a-guide-to-utilizing-color-histograms-for-computer-vision-and-image-search-engines/> [Accessed: 03/11/2019] [↑](#footnote-ref-2)
3. mathsworks.com “reduce the number of colors in an image” [Online]. Available at <https://uk.mathworks.com/help/images/reduce-the-number-of-colors-in-an-image.html> [Accessed: 26/10/2019] [↑](#footnote-ref-3)
4. mathsworks.com “Transfer Learning Using AlexNet” [Online]. Available at <https://uk.mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html>  
   [Accessed: 14/11/2019] [↑](#footnote-ref-4)
5. machinelearningmastery.com “Best Practices for Preparing and Augmenting Image Data for CNNs” [Online]. Available at <https://machinelearningmastery.com/best-practices-for-preparing-and-augmenting-image-data-for-convolutional-neural-networks/> [Accessed: 14/11/2019] [↑](#footnote-ref-5)
6. mathsworks.com “alexnet” [Online]. Available at <https://uk.mathworks.com/help/deeplearning/ref/alexnet.html> [Accessed: 14/11/2019] [↑](#footnote-ref-6)