Visual Search of an Image Collection

Name: Nithesh Koneswaran
Username: nk00374
Student ID: 6474079

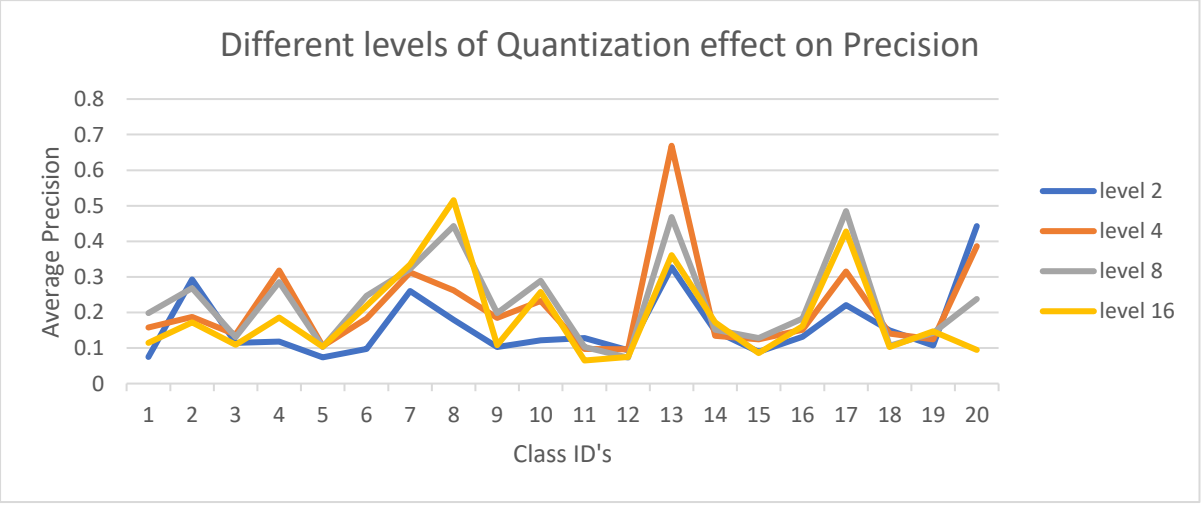
Contents

Global Colour Histogram

In this experiment I will be implementing a global colour histogram as a feature descriptor. In order to do this, I had to first quantize the RGB pixels in our images, which is to translate intervals of pixel intensities to a single value effectively reducing the number of colours in the image.[1] I will be experimenting and testing the different levels of RGB quantization and how it affects the performance of the visual search system.

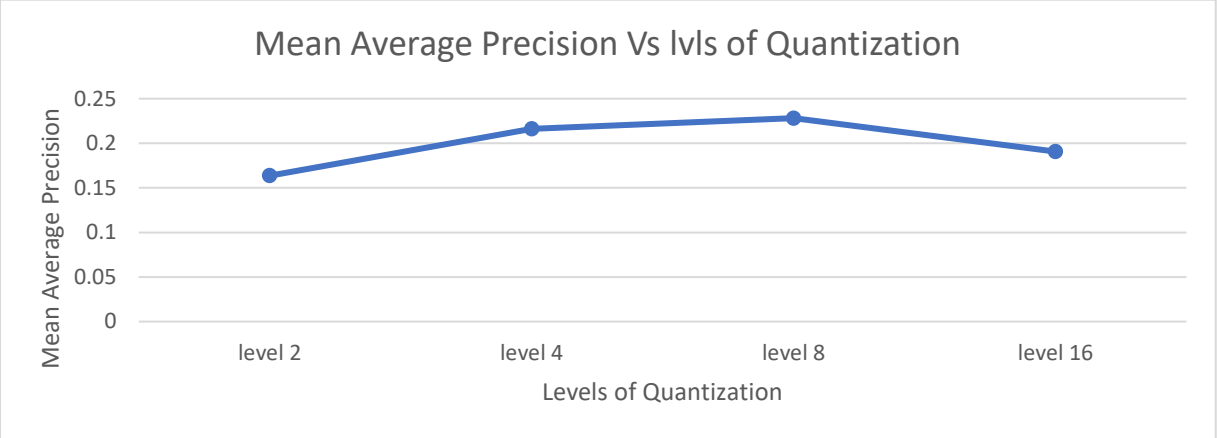
I experimented with 'q' values of 2,4,8 and 16 where 'q' is used to normalise RGB values within the range [0, (q^3)-1]. When 'q' is a small integer, the RGB colour values are mapped to a small range, meaning that they'll be less bins in our histogram. However, when 'q' is a large integer the RGB values will be mapped to a larger range, meaning that they'll be more bins in our histogram. 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 since there aren't many bins. At large values of 'q' I expect the performance of our system to decrease as the increased number of bins will make it difficult for the visual system to identify similar results since the histogram values of the images will be quite varied. This is caused by the large value of 'q' which configures the system to take shorter intervals of RGB values to be mapped to a single value, increasing the number of bins.

The results from the figure below shows the average precision for each class at different RGB levels of quantization. At class 8, where the level of 'q' is 16 (high), the average precision was the highest among the other levels of 'q' for that class, but at class 20, where the level of 'q' is also 16 (high) the average precision scored the lowest amongst the other levels of quantisation. The significance of the results can be explained through the content of the images. Class 8 held images of bicycles containing a higher range of colours than class 20 whilst class 20 held images of the ocean and had far less range of colours than class 8. This suggests that at high levels of quantization, where our feature histogram has many bins, our system will perform better at searching for images that have a large range of RGB values since the histogram has enough bins to represent the colours in the image. But will perform badly with low levels of quantization since there isn't enough bins to clearly represent the image affecting results. We get the same result but with the opposite effect when we look at images that have few colours.



[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]

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 cases where classes performed much worse than the other classes (look at figure above).



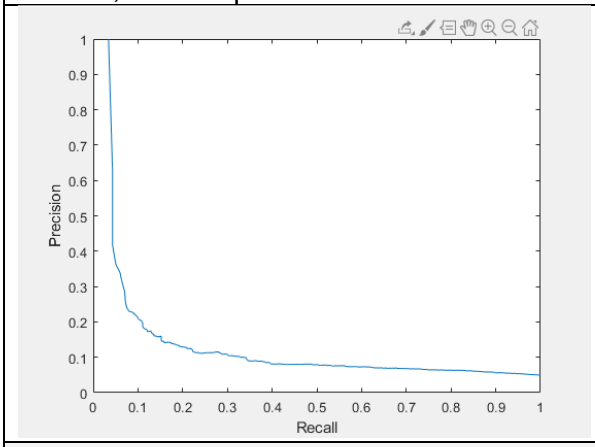
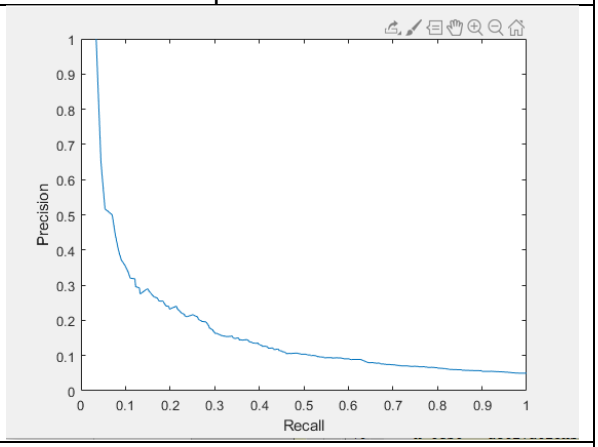
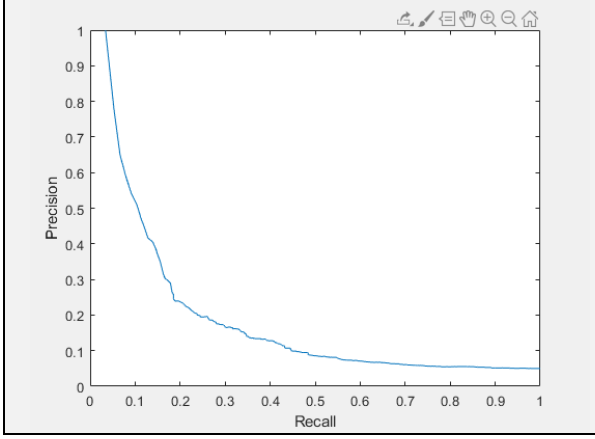
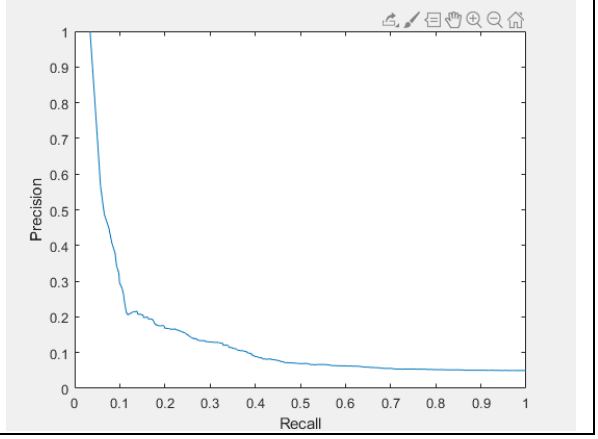
Overall, the performance of our visual system is poor. One of the drawbacks of the descriptor is that it does not consider any objects and shapes in the image, if we want to return similar images, we need a descriptor that can encode shapes as well as sparse features which we can then use to calculate the similarities.[2] 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] As the name suggests, 'Uniform quantization' evenly divides the colour space into boxes or bands. 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.

[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]

[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]

This figure shows the precision-recall curve for the different levels of quantization

Level 2, where ‘q’ = 2	Level 4 where ‘q’ = 4
	
Level 8, where ‘q’ = 8	**Level 16 where ‘q’ = 16**
	

We can see from the precision-recall curves that our visual search systems performed poorly. The level of quantisation that performed the best was when ‘q’ was equal to 8, we know this is the best by measuring the area underneath the curve (AUC) which is equal to the mean average precision, 0.2282.

Spatial Grid Histogram of Oriented Edges & Average RGB

In this experiment I created a function that will extract the texture and colour of an image and concatenate them into a vector. In order to evenly divide the image, I made sure that each image is resized to 256 x 256 x 3 so that each sub window of the image is of equal size. After resizing the image, I turned the image to greyscale and applied Sobel filter in order to get the gradient and magnitude of the image. I then used the following function `'mod(atan2(dy,dx), 2*pi)'` to convert the values to the range between 0 and 2π. I then used a double for loop to go through each sub-section of the image and for each sub section I used a double for loop to go though each gradient value in the current subsection and increment the corresponding bin in the histogram. I also checked if the magnitude was greater than threshold before incrementing the bin. After assigning bin values for a sub section my program will then normalise the histogram and append the results onto a vector. Also, for that current sub section the average RGB values are then appended to the list. After processing the first image we get a feature vector of length 11 (8 from the number of bins and 3 from the mean rgb values). This processed is repeated till all subsection of the image is processed.

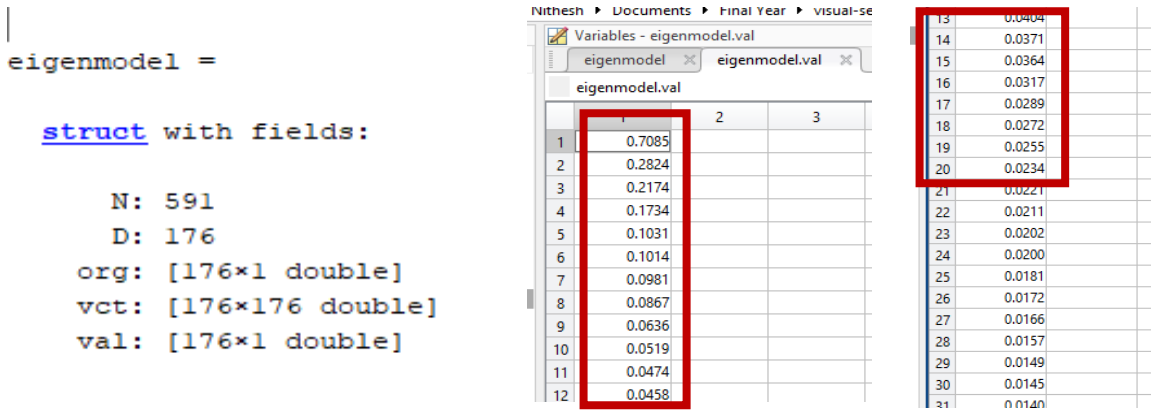
Block Size	Cell Size	Feature Vector	Angular Orientation	Mean Average Precision
4x4	64x64	112x591	4	0.2372
		176x591	8	0.2651
		304x591	16	0.2640
		560x591	32	0.2535
8x8	32x32	448x591	4	0.2222
		704x591	8	0.2476
		1216x591	16	0.2479
		2240x591	32	0.2391
16x16	16x16	1792x591	4	0.2114
		2816x591	8	0.2288
		4864x591	16	0.2235
		8960x591	32	0.2182
32x32	8x 8	7168x591	4	0.2147
		11264x591	8	0.2007
		19456x591	16	N/A
		35840x591	32	N/A

The figure above is the results of my experiments where I have experimented with different grid sizes and angular quantization. We can see that as the block size increases our feature vector for an image becomes large resulting in poor performance for our visual search system. For Angular quantization I found that when the histogram, used to represent the angular orientation of the image, had 8 or 16 bins the visual search system performed the best. Too many bins increased the size of our feature vector (calculated using block size * (angular orientation + 3) and too few bins decreased accuracy suggesting the features are closer together making it harder for the model to generalise the classes. In the table above I shaded the parameters that worked the best giving me the highest accuracy (0.2651).

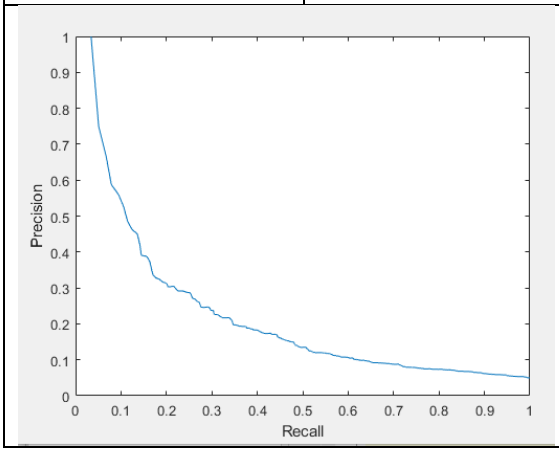
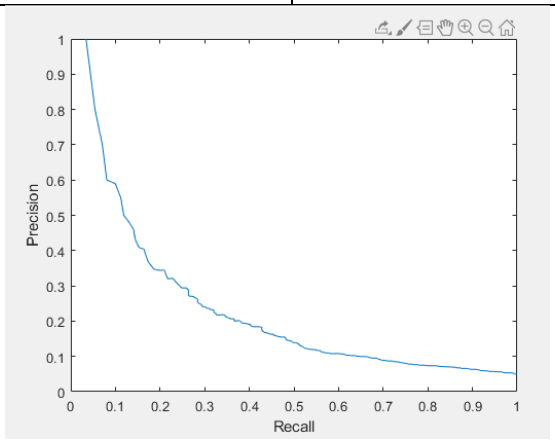
Overall, I assumed that the visual search system would have scored much higher accuracy than the global colour histogram experiment which scored 0.2282. Though this was not the case, I believe the accuracy of this experiment was affected by the resizing of the images to 256x256, this could have affected the gradients of the images and consequently how similar the features are. I also believe that there could possibly be an error in the code, although I have debugged the program several times and the logic of the code should be correct.

For improvements, instead of averaging the images and appending them to the feature vector we could have calculated a colour histogram for each cell in the image. From the previous experiment we found that the quantisation level of 8 was the best, after quantising the colours we could then create a histogram and append it to our feature vector instead of average RGB. Although the downside of this would be that our feature vector may be too large having the consequence of affecting how similar the images are.

In order to improve the best result, I decided to apply PCA to my dataset. The aim is to reduce the dimensionality of my data removing any null space. Through testing I found that it is best to project the data to have the dimensionality of 591x20 (average).



From the image above we can evaluate that variability is mostly expressed in the first 20 dimensions. Through trial and error, I found that reducing the dimensionality to 591x17-591x23 improved the results by around 26% increasing the accuracy from 0.2651 to 0.2753 (when dimensionality is reduced to 20).

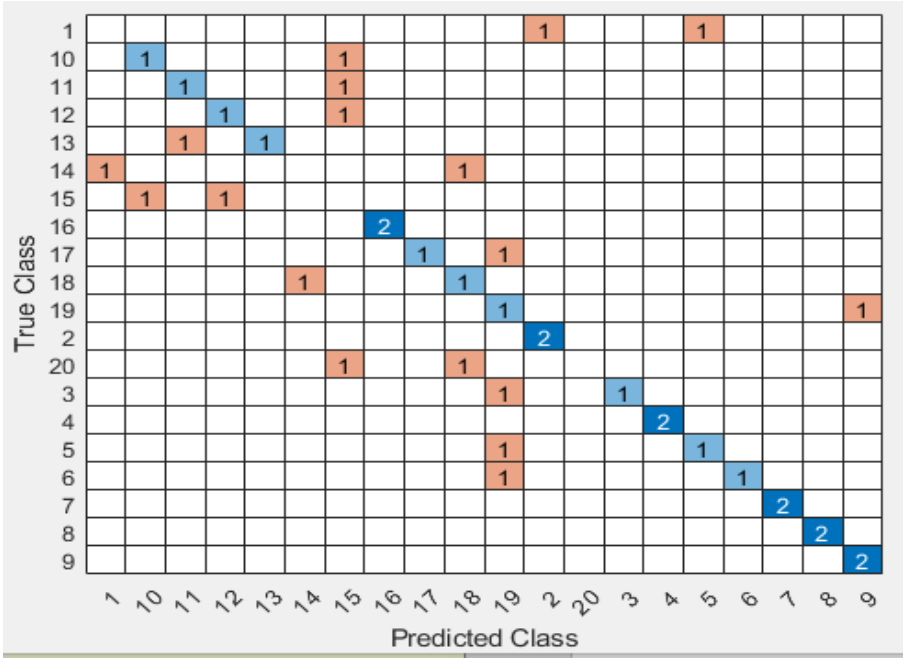
591x176	MAP = 0.2651	591x20	0.2753
			

Object Classification using AlexNet & Multiclass SVM

In this section I will be experimenting with SVM and CNN. For the feature descriptor I used AlexNet 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.

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 prevent it from over-fitting.[4] 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 helping the neural network to learn more efficiently.

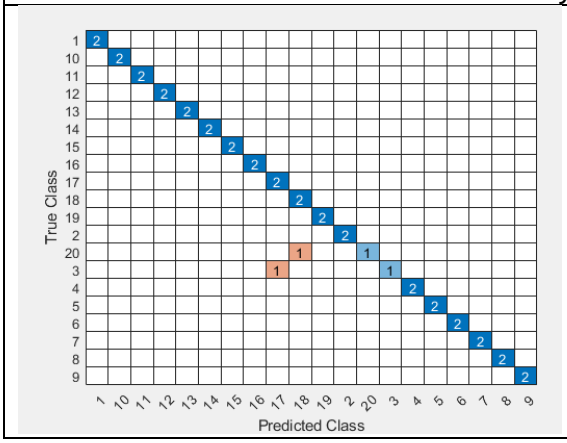
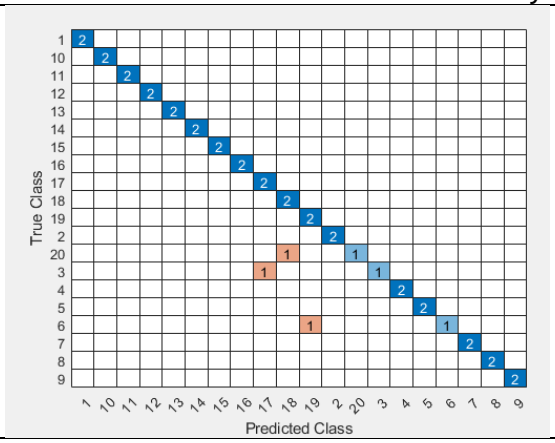
When I first completed the implementation the accuracy that we got from the SVM averaged to be 55%. I believed that the results could be improved if we increased the volume of training images, that way the network would be able to generalise the classes much more. I assumed that the results were probably affected due to the similarities of the classes and again if we had more data the network would have been able to generalise the classes more. I also discovered that Transfer Learning could be used for a network to learn a new task.[5] The confusion matrix below suggests which classes the neural network worked well with.



[4] 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]

[5] 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]

After reviewing the implementation, I discovered that the images were being divided by 255. Upon removing this, the accuracy of my results dramatically increased from 55% to 95%. It is important that the data is pre-processed like how AlexNet had been trained on. Therefore, I removed the division of 255 and subtracted the mean RGB channels of the training set from each image. However, upon further investigation I discovered that subtracting the mean RGB channels of the whole training set from each image had caused my accuracy to decrease by 2.5% percent. I believe the accuracy had decreased because I subtracted the mean from the msrc object training set instead of the training data that was used to originally used to train the AlexNet architecture. I have therefore decided to ignore mean subtraction method since it had reduced my accuracy and is not fit to be used, it is also most likely that the pre-process is done automatically through the AlexNet library.

Without Mean Subtraction – 95% accuracy	With Mean Subtraction – 92.5% accuracy
	

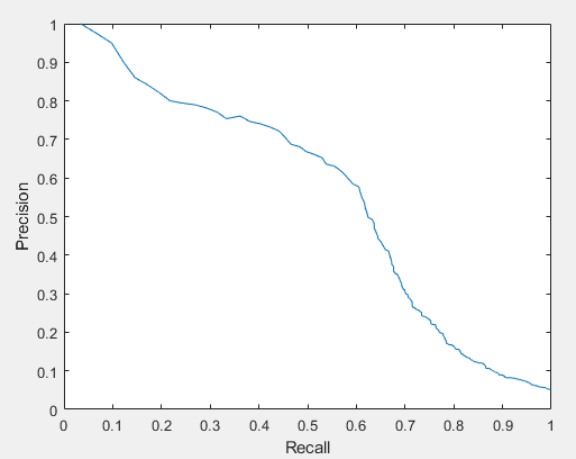
Overall completing this task took around 10hrs. Most of my time was spent trying to set the code up so that it can load the images along with the associated labels. I found that I had to manually pick the test images and label. I struggled to find a systematic approach for this since the number of images 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 from a cell format of 591x1 each containing 227x227x3 (resized images) to a numerical matrix so that I could feed the dataset into the SVM. However, the reason why this did not work was because I skipped the feature extraction. I later found out the function augmentedImageDatastore could be used to resize the images and be used to get the activations from the fully connected layer and then be fed into the SVM.[6]

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

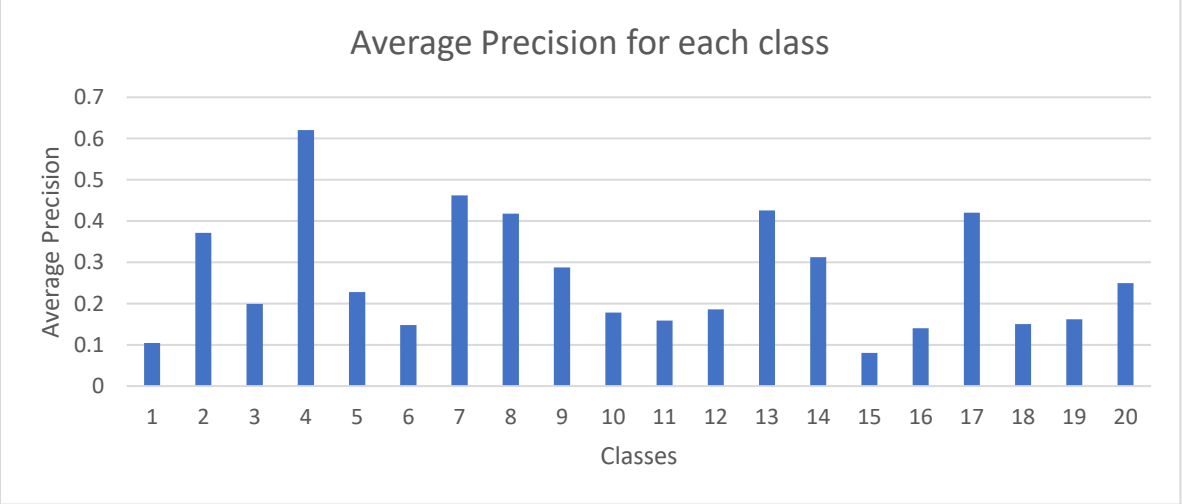
AlexNet Feature Descriptor for Visual Search

After implementing the CNN and SVM Object Classification I decided to use AlexNet's architecture as a feature extraction tool. In order to use AlexNet's architecture I resized my images to 227x227x3 and fed them into the architecture. At the fully connected layer, called 'fc7', I took the values and concatenated them into a vector which was then used as our feature descriptor. I then experimented with several distance measures to boost the mean precision accuracy further.

Using Euclidean distance as the distance measure I managed to get a Mean Average Precision score of 0.6175. The figure on the right shows this implementation's precision and recall curve and the figure below shows the average precision for each class.



In the figure below, we can see the average precision for each class. We can see that the system performed relatively well for most of the classes. However, we have 8 classes that have an average precision that is less than 50% suggesting the image retrieval of these classes are quite random. I had expected much more accurate results since the object classifier implementation performed very well.



It is possible that the system could have gotten mixed up between class 1, 5, 12 since they share a lot of similarities. They all share a green background and an animal in the image. These features are also evident in class 16 and 19 where they share a green background. In order to improve the system, we could try Transfer Learning to fine-tune the network.

Another method of improvement would be to experiment and try different pre-trained neural network such as YOLOv2, Googlenet or VGG-19. In addition to this we could experiment with a range of optimisation techniques such as an evolutionary algorithm. An evolution algorithm could be used to determine and optimise the dimensions, sizes and parameters of the CNN architecture.

In this step I applied PCA to my dataset to project them into a lower dimensional space in order to improve accuracy. Looking at the Eigen values for my features I found that the data variability was largely expressed over hundreds of dimensions. Through trial and error, I found that reducing the dimensionality to around 591 x 30 provided us with the best result.

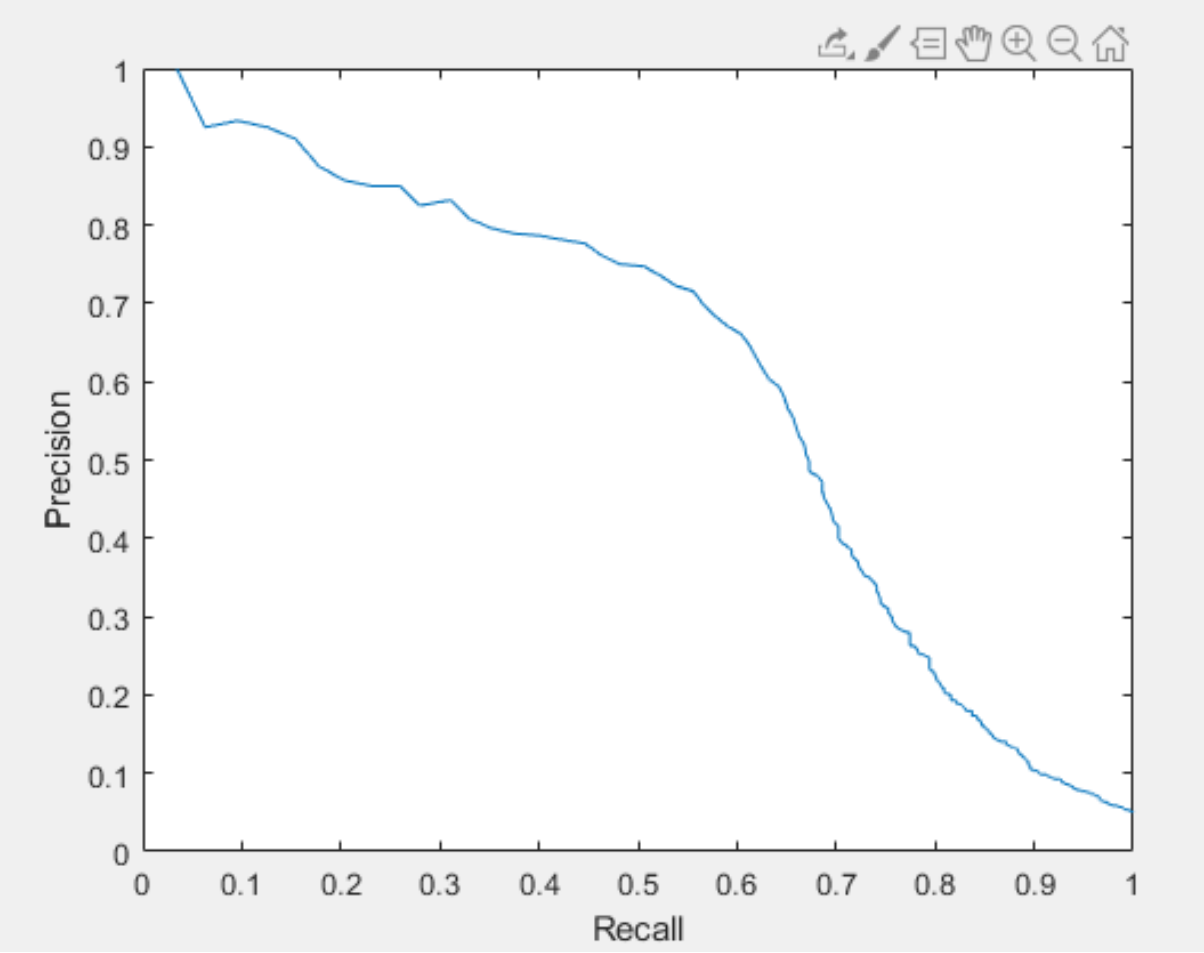
```
eigenmodel =

  struct with fields:

      N: 591
      D: 4096
    org: [4096×1 single]
    vct: [4096×4096 single]
    val: [4096×1 single]
```

Feature Vector	MAP
591x20	0.6586
591x30	0.6605 <- Best Result
591x40	0.6557
591x50	0.6509
591x60	0.6420
591x70	0.6382
591x80	0.6339
591x90	0.6315
591x100	0.6295

Before lowering the dimensional space, our visual search system had an accuracy of 0.6175. After lowering the dimensional space, our system had a 14% increase. Below shows the updated precision and recall curve.



Evaluation Methodology

For evaluating the visual search system, I had a predefined set of query images that was kept the same so that I can compare the experiments. At the end the results from each query would be combined to calculate the overall statistic. In order to evaluate our system, I created a ground truth. For this I used the file names since I assumed that 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 was then used to calculate the precision and recall. For each of my experiments I have plotted a PR curve and a confusion matrix, for any classification experiments.

Use of PCA

For some of my experiments I projected the feature descriptors into a lower dimensional space using PCA to improve performance. I found that PCA only improved the accuracy by around 10-20%. The drawbacks of using PCA is that we might be throwing away important data that could affect the similarity between the data.

Different Distance Measures

Norm	Global Colour Histogram	Spatial Grid without PCA	Spatial Grid with PCA	AlexNet without PCA	AlexNet with PCA
Vector	591x512	591x176	591x20	591x4096	591x30
L0.5	0.2418	0.3002	0.2294	0.6083	0.5483
L1	0.2679	0.2923	0.2624	0.6121	0.6256
L2	0.2282	0.2651	0.2753	0.6175	0.6605

I have implemented 2 other distance measures to try increase the performance of my visual search system. From research, L2 norm tends to work better on datasets that have fewer dimensions. When dealing with a high number of dimensions its better to work with Lk norm where the value of k is a fraction.[7]

Looking at Global Colour Histogram we can see that using L1 Norm gave us the best performance out of the three distance measures. However, I had assumed since the feature vector had a high number of dimensions, L0.5 norm should have outperformed L1.

With ‘Spatial Grid without PCA’ we can see that as we move from L2 to L0.5 the performance of the system increases as expect. With ‘Spatial Grid with PCA’ as we move from L0.5 to L2 the performance increases as expected since our feature descriptor has less dimensions than before, meaning that it would be preferable to use L2 norm.

With ‘AlexNet without PCA’ the performance decreased from L2 to L0.5 which goes against the knowledge that its preferable to use a fractional k value for Lk norm when the dataset exhibits high dimensionality. Lastly, with ‘AlexNet with PCA’, the performance increases from L0.5 to L2 as expected since we decreased the number of features.

[7] Charu C. Aggarwal, Alexander Hinneburg, Daniel A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space", [Online]. Available at https://bib.dbvis.de/uploadedFiles/155.pdf [Accessed: 30/11/2019]

Visual Search Output

Global Colour Histogram using L1 Norm, Mean Average Precision Score: 0.2679

Spatial Gridding using L0.5 Norm, Mean Average Precision Score: 0.3002

AlexNet Visual Search with PCA

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] 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]

[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]

[4] 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]

[5] 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]

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

[7] Charu C. Aggarwal, Alexander Hinneburg, Daniel A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space", [Online]. Available at https://bib.dbvis.de/uploadedFiles/155.pdf [Accessed: 30/11/2019]