Visual Search – EEE3032 Coursework Report

Mohit Motwani
Mm02893@surrey.ac.uk
6665370

Tabltime of Contents

Abstract

This report introduces and describes different methods for Visual Search, an important application to find information about unknown objects that we find in day to day lives. The Introduction describes important concepts like Visual Search, Feature Extraction and the Evaluation Metrics necessary for the reader. Moving ahead, different experiments ranging from basic features from Global Colour Histograms to advanced methods like Transfer Learning are explained, explored and their results are analysed. Additionally, Image Classification with Support Vector Machines is performed to demonstrate the effectiveness of CNNs as feature extractors. Lastly, a conclusion about the analysis of various experiments and further experiments are proposed to improve the results with the methods discussed in the report.

Introduction

Visual Search

Visual Search is the concept of using an image as a query to search among a dataset of images, i.e., a search engine for images. This requires extracting features of images and comparing them. As Visual Search is a computationally expensive and difficult task, the dimensions of the features and the robustness of the feature extractors are of enormous importance.

Feature Extraction

Every entity has certain characteristics that identifies its uniqueness or describes the entity from certain aspects. The characteristics of an image are also called features. The process of extracting these features is called Feature Extraction. Essentially these features are vectors, a set of numbers, which can be processed by a computer.

In Visual Search, these features from images are compared to determine if two images are similar. Therefore, the feature extractors, the algorithms to extract features, are enormously important. In this experiment, four feature extractors are used : *Global Colour Histogram*, *Spatial Grid Texture Histogram*, *Scale Invariant Feature Transform with Bag of Visual Words* and *Pre-trained Convolutional Neural Networks*. These feature extractors are described and their results are shown in the Experiments section.

Comparing Features

Once we have the features, we need distance metrics to compute similarity between two features. The reliability of different distance metrics may vary depending on the dimensionality of the features[1], the experiment and the data. The following methods are used to compare features in this experiment.

1. Manhattan Distance

Manhattan Distance is the distance between two points(or equivalently vectors) measured along axes at right angles[8].

$$Manhattan(x, y) = \sum_{i=1}^{n} x_i - y_i$$

2. Euclidean Distance

Euclidean Distance finds the shortest distance between two points (or equivalently vectors) in a feature space.

$$Euclidean(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3. Cosine Similarity

Cosine Similarity measures if two vectors are pointing in the same direction. Mathematically, it measures the cosine of the angles between two vectors[5]. This is a reliable metric because it doesn't change with the magnitude of vectors but their direction. Therefore it is prudent to consider this metric for high dimensional data.

$$cosinesimilarity(x, y) = \frac{x.y}{|x| * |y|}$$

4. Mahalanobis Distance

Mahalanobis Distance is a measure of distance between two points in units of the standard deviation of a projected space P. In this experiment the projected space is calculated using Eigen Modelling.

Let U and V be the Eigen Values and the Eigen Vectors of a feature set, respectively and x and y be features of two images. Let u be the mean vector of the feature set. The following are the steps to compute Mahalanobis Distance

1. Project features to the Eigen Space

$$xf = V^T * (x - u)$$
$$yf = V^T * (y - u)$$

2. Compute the Mahalanobis Distance

$$Mahalanobis(xf, yf) = \sqrt{\sum_{i=1}^{n} \frac{(xf_i - yf_i)^2}{U_i}}$$

Evaluation Metrics

1. Precision

Precision measures the proportion of returned results that are correct.

$$Precision(n) = \frac{Correct\ Predictions\ in\ the\ n\ results}{n}$$

2. Recall

Recall measures the proportion of the true results that were returned. If N is the true class for the present query, then recall is

$$Recall\ (n) = \frac{Correct\ Predictions\ in\ the\ n\ results}{Frequency\ of\ Class\ N}$$

3. Average Precision (AP)

$$AveragePrecision(n) = \frac{\sum_{i=1}^{n} precision(i) * relevantPrediction(i)}{Frequency\ of\ class\ N}$$

Where $precision(i)$ is the precision at i^{th} result and $relevantPrediction(i)$ is 1 if the i^{th} prediction was correct otherwise 0. In this experiment, the AP is calculated per class of the dataset.

4. Mean Average Precision (MAP)

$$MeanAveragePrecsion(n) = \frac{\sum_{i=1}^{N} AveragePrecision(i)}{N}$$

In this experiment, MAP is used as the prime metric to describe the effectiveness of a feature extractor over the dataset.

Note that the average precision per class results in this report is the average of the average precision of 10 random queries for that class. As the precision may highly depend on the image queried, an average of different provides a better picture of the feature extraction method.

Experiments

The dataset used in this experiment is the Microsoft Research Cambridge Image dataset which has a total of 591 images and 20 categories.

Class	Label	Frequency
1	grass	30
2	trees	30
3	building	30
4	aeroplane	30
5	cow	30
6	face	30
7	car	30
8	bike	30
9	sheep	30
10	flower	32
11	sign	30
12	bird	34
13	books	30
14	chair	30
15	cat	24
16	dog	30
17	road	30
18	water	30
19	body	30
20	water with sky	21

Table 1 - MSRC Dataset Category Information

One important observation about the dataset is there is immense ambiguity among certain categories. For example, category 18 and category 20 are very similar because all images contain water but have different objects in the background or foreground; category 18 has mostly boats and fences and category 20 has sky in most images. Another example, is the similarity between categories 1, 5, 12 and 16 where all have significant amount of grass. Therefore, it is wise to take the numeric metric results with a grain of salt as they are computed based on the category information in the dataset.

1. Global Colour Histogram

In Global Colour Histogram, the ranges of RGB values are binned or quantized and the frequency of pixels in each colour range is calculated. These frequencies across different ranges are the features of the image. For example, consider two ranges: *Bin A* - $(0-9, 0-9\ 10-19)$ and Bin B - $(20-29, 20-29, 20-29)$ in order *(R, G, B)*. A pixel with values $(0, 5, 12)$ will fall into *Bin A* and a pixel with values $(22, 23, 28)$ falls into *Bin B*. Similarly, all pixels are slotted into bins and the frequency of each bin is computed. This frequency vector becomes the feature of our image.

A single parameter Q is used to quantize the bins. It defines the number of bins in each channel. It's possible to have 3 parameters for quantization for each channel. However in this experiment each channel has equal number of bins. *Table 2* shows the Mean Average Precision (MAP) – over different metrics over different quantization.

	Manhattan (N=20)	Euclidean (N=20)	Cosine (N=20)
Q=4	12.13	10.6	10.3
Q=8	12.6	10.19	11.16
Q=12	12.1	10.6	10.5

Table 2 - MAP over different values of Q and different metrics

The RGB histogram encodes the colour information of the image. This is demonstrated in *Figure 1*.

Figure 1 - Visual Search Result of Category 2 (trees) as query image

The first image in the row is the query image and the rest are results of the visual search. It's evident how similar in colour the images are: all images have green as the dominant colour near the centre and have white colour at the top. The 6th, 9th and the 10th image belong to the category of roads in the dataset. But due to the similarity in the colour information (white-blue sky, green trees), they have similar features. Visually, these are good results. Figure 2 is the precision-recall (PR) curve of the above result.

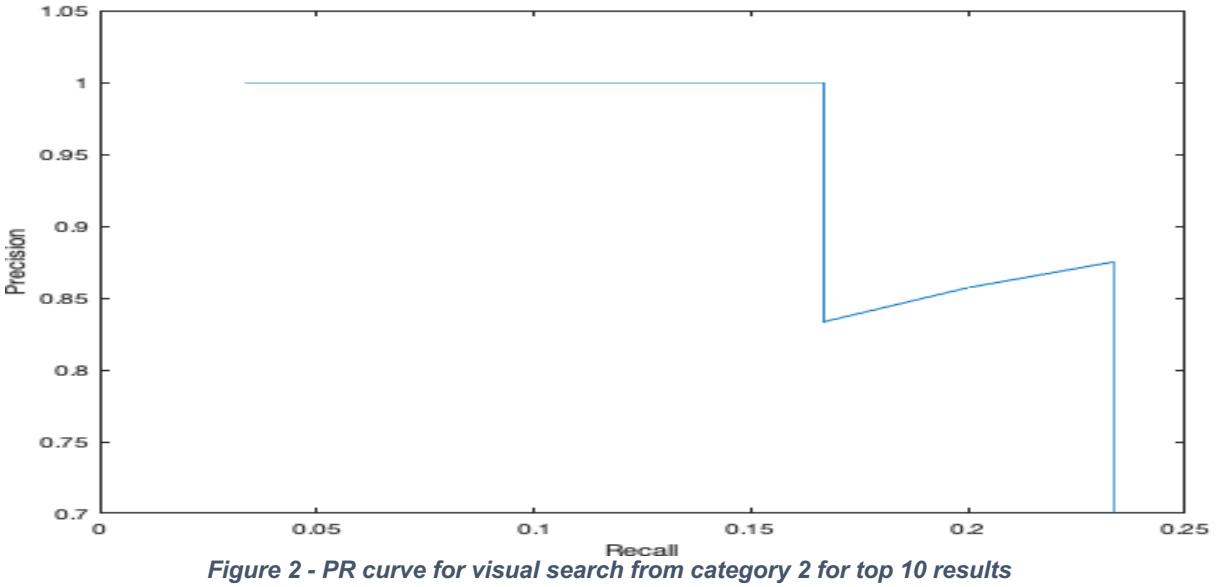

Figure 2 - PR curve for visual search from category 2 for top 10 results

You can see the precision score getting a hit around the middle due to the 6th and last results, even if visually the images are very similar. However in practice, merely encoding colour information is not enough to extract all similar images. First, entirely unrelated images with similar colour distribution will return high similarities. Second, the Colour histogram extractor is very sensitive to illumination changes (intensity of light on objects) in the images. Third, Colour histogram doesn't not extract global properties of an image. This can be demonstrated by Figure 3.

Figure 3 - Visual Search of an Image of Category 6 (Face)

Most of the images have a dominant colour of white in the images. However, the top results are entirely unrelated. *Figure 4* shows the PR curve of Global Colour Histogram over all images of the dataset with Q=8.

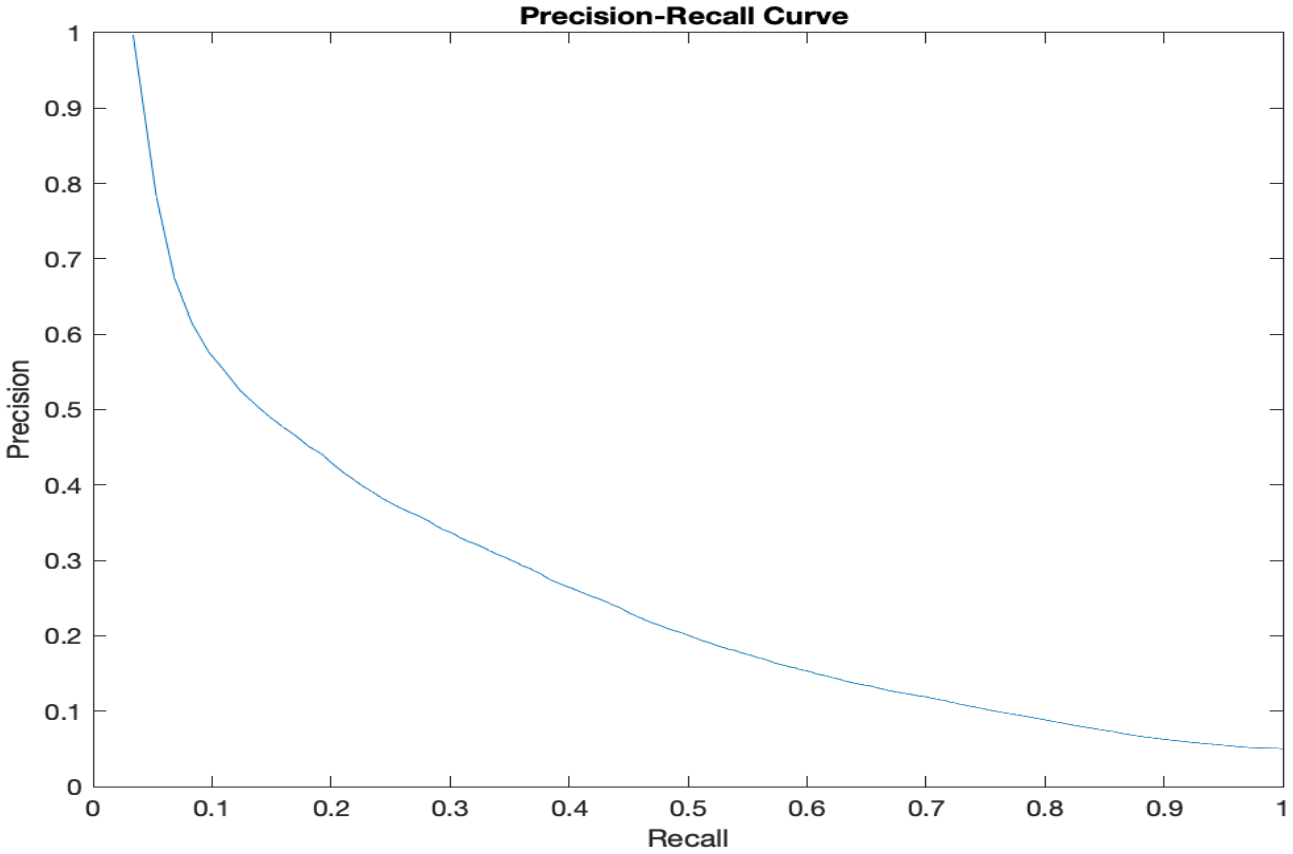

Figure 4 - PR Curve of Global Colour Histogram with Q = 8 over Manhattan Distance

The PR curve in Figure 4 demonstrates that the RGB Histogram is not a robust feature extractor. Furthermore, you can notice in *Table 7* (Appendix), with the average precision per class, that categories 2, 8, and 9 have relatively better Average Precision. One reason could be attributed to the dominance of a single colour within these categories. Additionally, categories 11 and 12 have poor average precision due to many different colour distributions in their images.

2. Spatial Grid Texture Histogram

In Spatial Grid Texture, the image is divided into grids, where each grid spans certain rows and columns of pixels. In each grid, the orientation of the edges is calculated (texture) and a histogram of these orientations is constructed. Additionally, the average of each colour channel from the grid is added to the histogram vector. These essentially are the features contributed by the grid. Lastly, features of all the grids are concatenated to represent the image.

	N = 20	Manhattan	Euclidean	Cosine
Grid Size	**Orientation Bins**			
20x20	**20**	12.9	11.3	9.3
	30	12.3	11.3	8.6
25x25	**20**	13.13	10.7	10.28
	30	13.75	11.27	10.39
50x50	**20**	12.3	11.3	10.4
	30	13.8	10.8	10.7

Table 3 - MAP of Texture Descriptor over different grid Sizes and orientation ranges. All Images were resized to 200x300. Orientation bins is how the angles of edge directions are quantized between (0, 360) degrees.

From *Table 3*, we can notice that the MAP of higher orientation bins is slightly better than lower number of orientation bins irrespective of the grid size. Figure 5 shows image

Figure 5

There is a clear pattern in all the images in *Figure 5:* there is high texture(many edges) in at the centre of the images, noisy or high-variability(many possible orientations) texture at the bottom and low texture at the top of the images.

Figure 6

Figure 6 shows another good result. The texture at the centre of these images is substantially similar.

Figure 7 – Visual Search Results of Texture Descriptor. Texture Descriptors fails to extract global properties of an image.

However, the texture descriptor fails to extract global properties of an image. *Figure 7* demonstrates this. The extractor returns images with grass and tree patches (understanding local properties) but fails to extract images with human bodies around these patches. Furthermore, the texture descriptor is not robust against affine transformation. For the categories 12 and 19 of birds and bodies, where both entities shift (translate along x and y axis) a lot from image to image, the MAP is significantly low (Table 8 Appendix). *Figure 8* shows the PR curve of Spatial Grid Texture Descriptor over all images of the dataset with grid size 25 and 30 orientation ranges with Manhattan similarity.

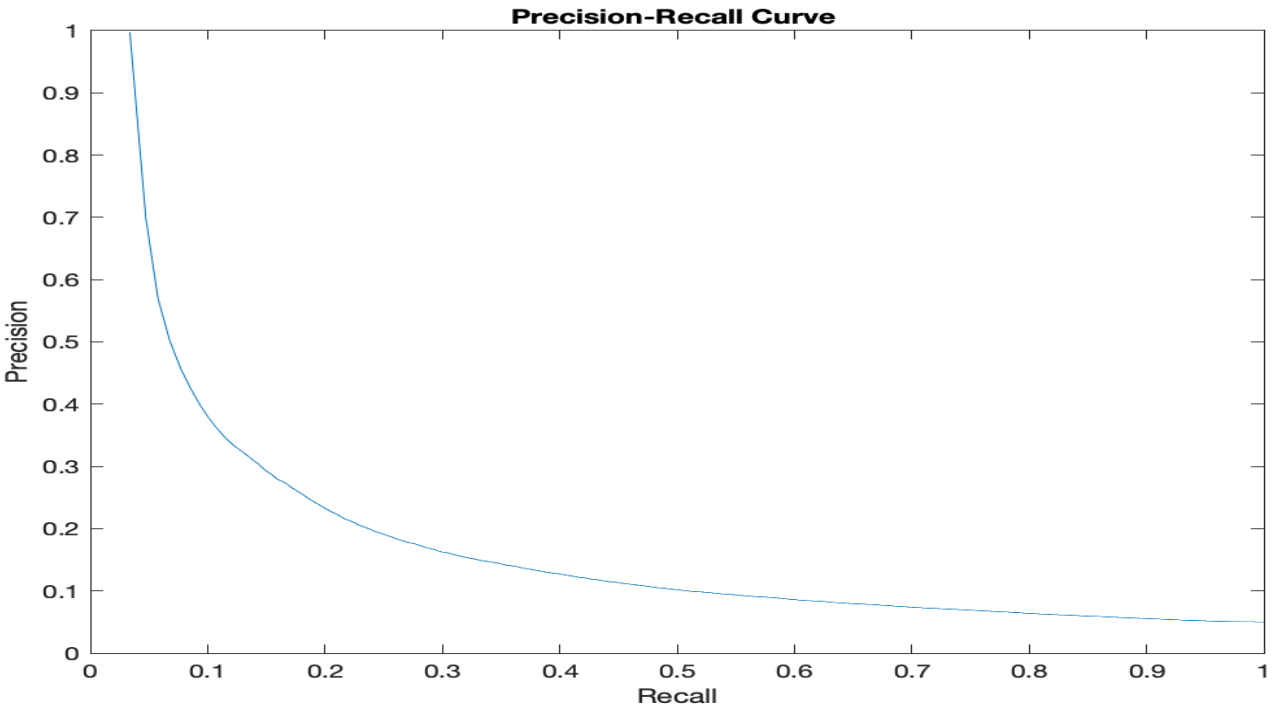

Figure 8 - PR curve of Spatial Grid Texture Descriptor over all images of the dataset with grid size 25 and 30 orientation ranges with Manhattan similarity

2.1 Applying PCA

As the feature dimensions of texture descriptors are high (3168 features per image for 25X25 and 30 orientation ranges), we can reduce the dimensionality of the features using Principal Component Analysis (PCA). Using Eigen Modelling, we can derive the Eigen Values and Eigen Vectors of the features. The Eigen Values represent the variance across corresponding Eigen Vectors. The higher the variance, the higher the information contributed from the features across the corresponding Eigen Vectors. The lower variance vectors can be discarded because they do not contribute significantly to the feature set. The vectors with highest variance are the Principal Components of the features.

Our features are projected to the Eigen Space and Mahalanobis Distance is computed. Table 4 shows the results of two types PCA performed on the texture descriptor. In first type the top principal components which contribute 97% of the variability are selected. Second, the top 20 principal components are selected.

	N = 20	Manhattan		Euclidean		Cosine		Mahalanobis	
Grid Size	**Orientation Bins**	**97% variance contribution**	**20 components**	**97% variance contribution**	**20 components**	**97% variance contribution**	**20 components**	**97% variance contribution**	**20 components**
20X20	**30**	10.01	9.7	11	10.5	10	10.11	11.1	9.04
25x25	**30**	10.54	9.4	11.4	9.95	10.6	10.6	9.6	9.5
50x50	**30**	10.3	10.4	10.67	10.1	11.23	11.27	9.7	9.9

Table 4

You can notice that after significantly decreasing the dimensionality (from 3168 for grid size 25x25 to 119 or 20 components), we still get similar results (*Figure 10*). It should be noted that the change in MAP for Manhattan distance after PCA is due to fact that metrics perform differently in different dimensions. Manhattan can be considered more reliable in higher dimensions than Euclidean Distance[1]. *Figure 9* is the PR curve with Mahalanobis Distance for the entire dataset.

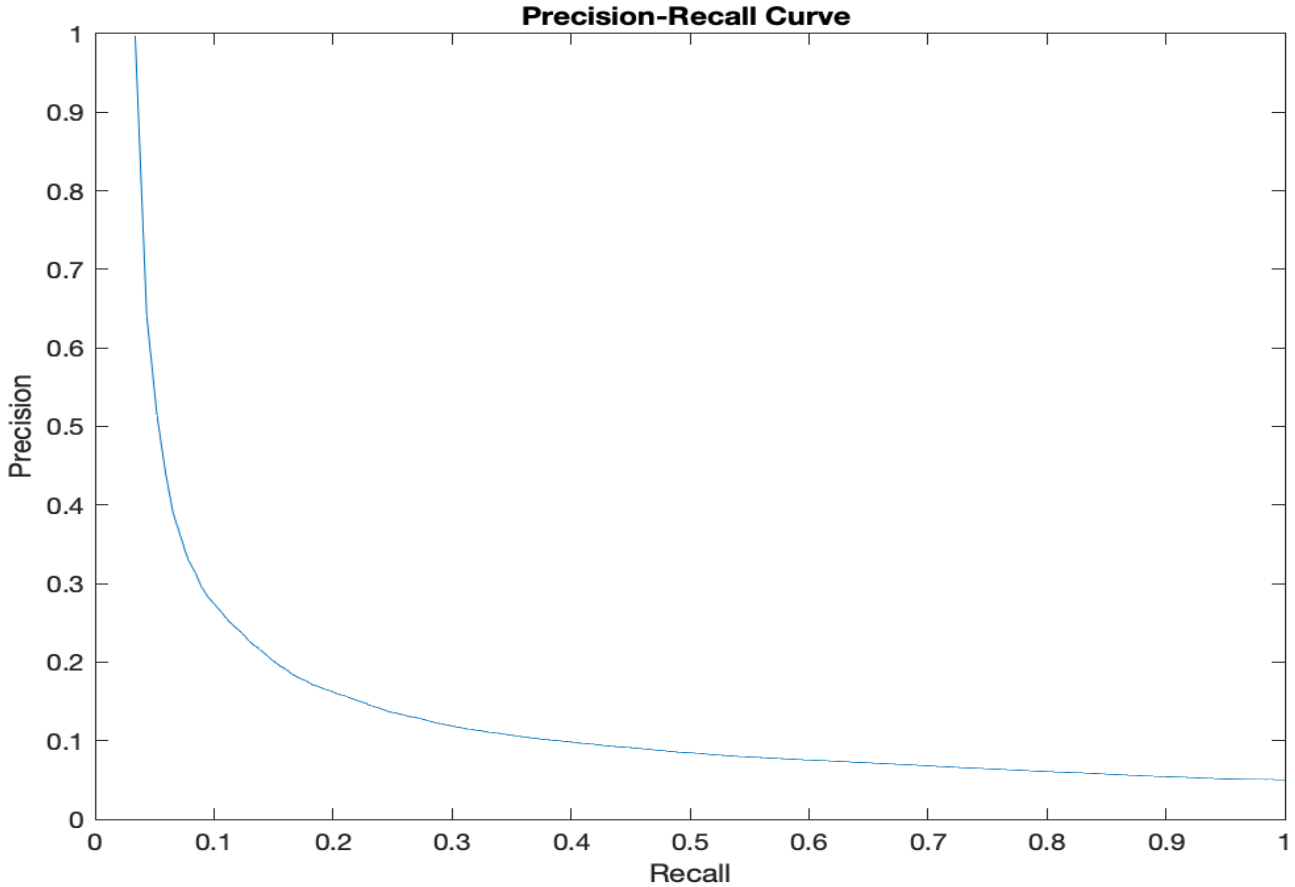

Figure 9 - PR curve with Mahalanobis Distance for the entire dataset. Parameters - 97% variance contribution (119 Principal Components) Grid Size 25 and 30 Edge Orientations.

Figure 10 – Results of visual search for the Books category after applying PCA to the texture descriptor.

3. SIFT And Bag of Visual Words

Scale Invariant Feature Transform[4] is a method to extract sparse and affine invariant features from an image which are robust against affine transformation up to a certain extent. Following is an high level overview of SIFT:

1. The image is scaled to multiple levels and a Gaussian filter of increasing standard deviation is applied to each scaled image. This set of images is known as the scale space.
2. Difference of these Gaussian Images is taken, i.e., Difference of Gaussian (DOG).
3. From the DOG images, local maxima and local minima (extrema) pixels are selected considering neighbours of the pixel from current image and corresponding pixels and their neighbours from adjacent images.
4. These points are further refined using Taylor Expansion and comparing edge extrema with principal curvatures. These are the keypoints of the image.
5. Each keypoint is then assigned a reference orientation depending on the orientations of the neighbourhood. The further away pixels from the keypoint contribute less to the orientation of the keypoint compared to the nearer pixels.
6. Descriptors of each keypoints is calculated by rotating the coordinate system to match the reference orientation.
7. Take the 16 by 16 neighbourhood of the keypoint (in the reference orientation), and divide the it into 4 by 4 grids (16 grids). Each grid computes a 8-orientation histogram. These histograms are concatenated together to become the SIFT descriptor of the keypoint. The length is 16 * 8 = 128 descriptors per keypoint.

Computing extrema from the scale-space, assigning reference orientations to keypoints and computing texture around the keypoints, allows the descriptors computed to be affine invariant and illumination changes.

The Bag Of Visual Words (BOVW) is used to represent the image in terms of its keypoints, as opposed to words to represent a document in Natural Language Processing Applications. In BOVW, all descriptors of the dataset are clustered using K-Means Clustering. To summarise the clusters the descriptors belong in, a histogram for each image is constructed. This histogram is the feature of the image. Note that the histogram is essentially encoding the kind of patches (around the keypoints) that are present in the image.

The experiment was conducted with k=150 and k=300. *Table 5* show the results of the visual search with SIFT descriptors and BOVW.

N = 20 Clusters	**Manhattan**	**Euclidean**	**Cosine**
150	8.5	9.4	10.6
300	9.16	9.8	12.09

Table 5

The following images shows a few results

Figure 11 – Visual Search Results with BOVW computed from SIFT Descriptors

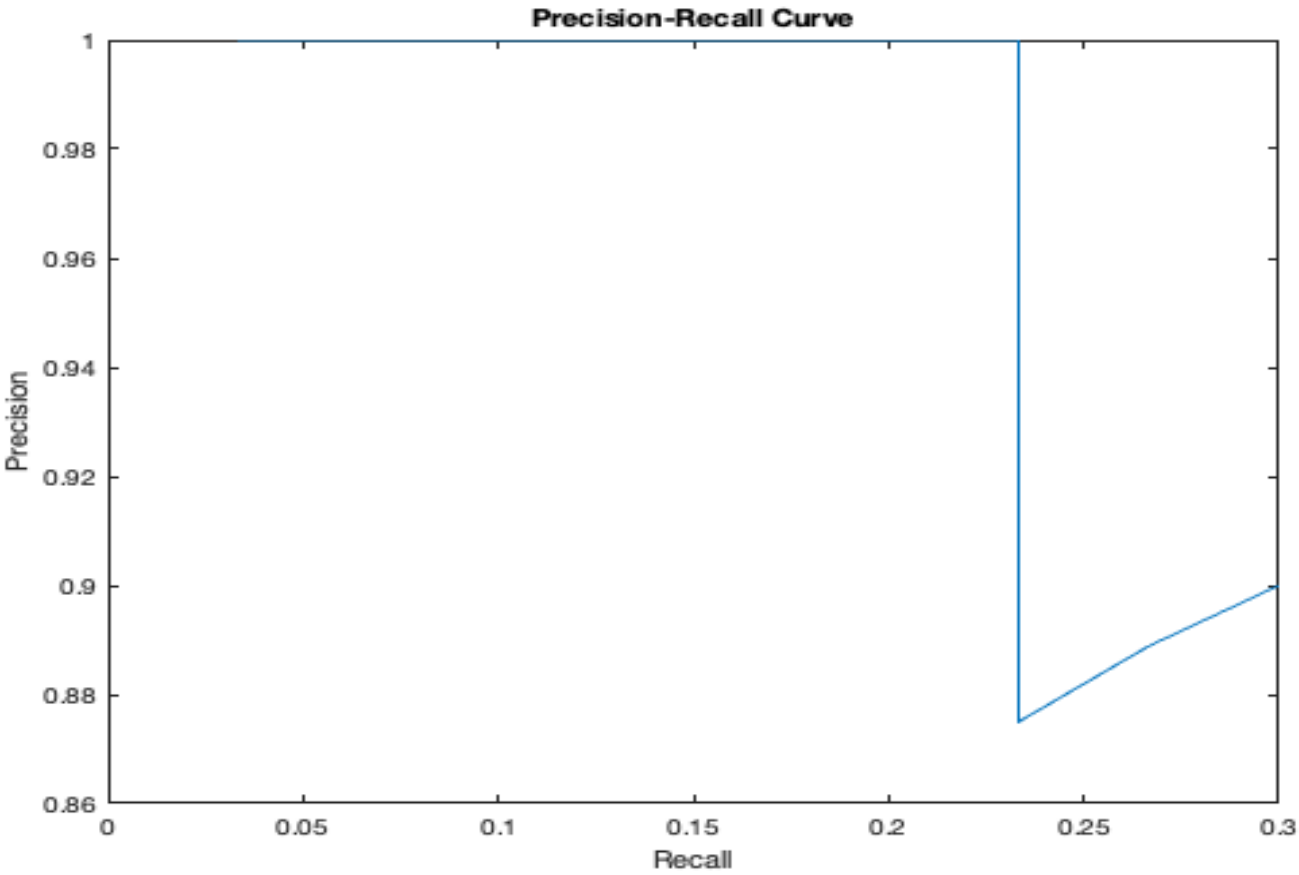

Figure 12 – PR curve for results of Figure 11

Figure 13

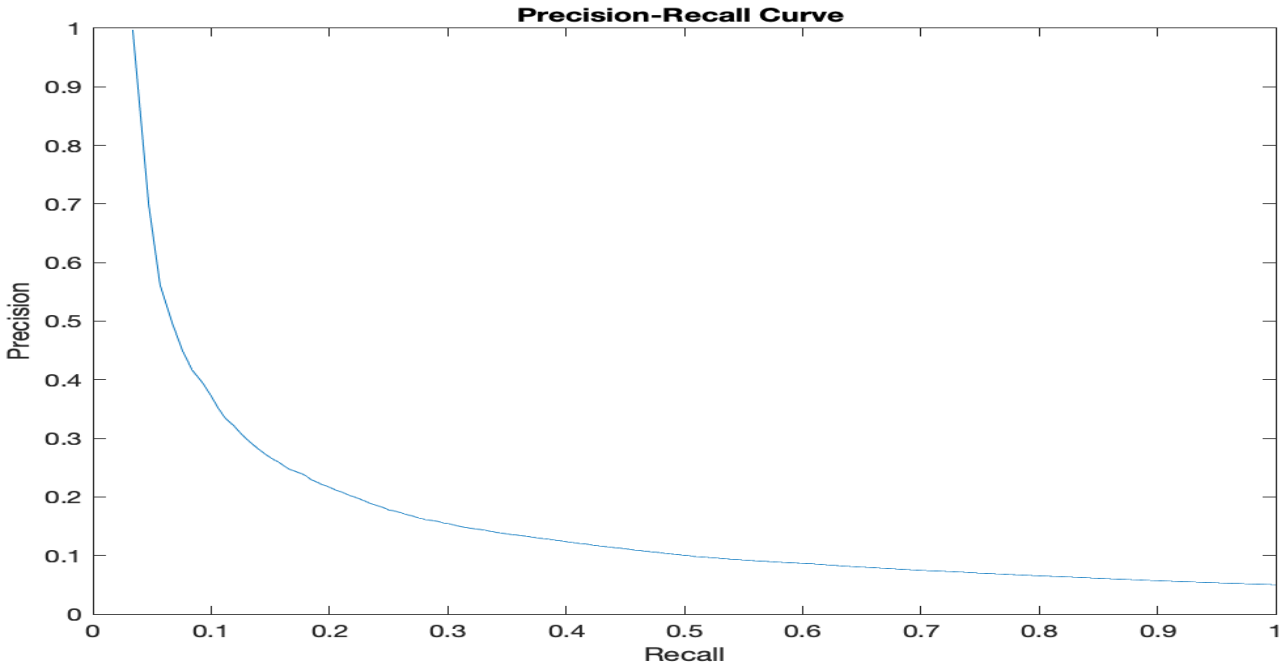

Figure 14 - PR curve with Cosine Similarity for the entire dataset with 300 clusters made from SIFT Descriptors

Although the results are not promising, there are three hypothesis for the poor metrics. First, the SIFT descriptors may only be able extract comparable features for objects or scenes which are nearly same (affine transformed). Second, k-means clustering initialises random clusters[7] and the accuracy of the resulting clusters depends on this random initialisation. Re-training the clustering algorithm may yield better results. Third, k-means uses Euclidean Distance which may not be preferable in higher dimensions [1]. A different distance metric may yield better results.

4. Transfer Learning

Transfer Learning is the concept of using pre-trained Deep Learning models on different but related datasets and applications. It is essentially transferring the knowledge it has learnt from a dataset to another similar application. In this experiment, the pre-trained models used were:

1. VGG-19 [2]
2. ResNet-50 [3]
3. ResNet-152 [3]

The above models were trained on the ImageNet Dataset and all are Convolutional Neural Networks (CNN)[6]. A CNN has two parts: a convolution layer part which does the feature extraction and a dense layer part for prediction. In this experiment, only the convolution layer part is used to extract features from the Image. *Table 6* shows the Mean Average Precision of these models.

N = 20 Pre-Trained Models	Manhattan	Euclidean	Cosine
VGG-19	9.02	12.7	42.7
ResNet-50	19.3	29.5	45.9
ResNet-152	18.99	31.05	47.04

Table 6

The improvements in the results is fascinating given the models weren't trained on this dataset. The convolution kernels, also known as filters, are the prime feature extractors in these pre-trained models.

Querying an image from category 19, whose average precision has been low across all feature extractors but high with pre-trained CNNs(*Table 10 and 11*, Appendix), might demonstrate the effectiveness of transfer learning.

Figure 15 – Visual Search Result for Image in Category 19 (body) with ResNet 152 features

Figure 16 – Visual Search Result for Image in Category 19 (body) with SIFT – BOVW features

Figure 17 – Visual Search Result for Image in Category 19 (body) with Texture Descriptor features

Figure 18 – Visual Search Result for Image in Category 19 (body) with Global Colour Descriptor features

It is obvious to understand the results of the Colour Histogram Feature Extractor. The same is not true in the case of BOVW and Texture Descriptor. One hypothesis in case of BOVW is that SIFT has detected key points of the tree patches which are common across most images in Figure 16.

The significantly better results of ResNet could be attributed to large and highly varying ImageNet dataset. Consequently, the convolution kernels and pooling operations have learnt to extract low-level and global features common in most images and therefore are able to extract relevant feature from this dataset to yield significantly better results compared to the classical computer vision techniques used earlier. *Figure 19* can summarise the effectiveness of CNNs as feature Extractors.

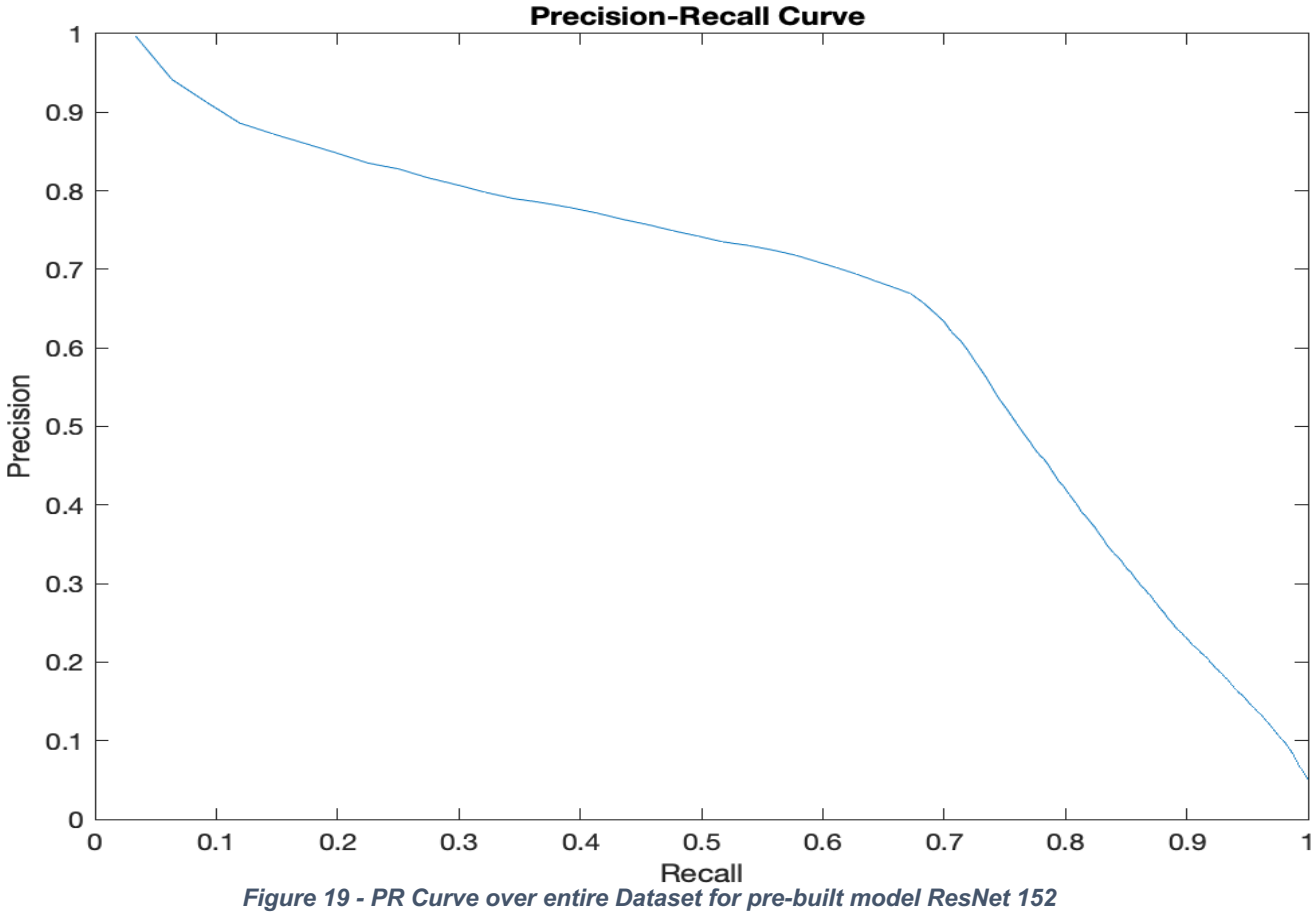

Figure 19 - PR Curve over entire Dataset for pre-built model ResNet 152

5. Image Classification with Support Vector Machines

Image Classification is the concept of categorising a particular image into pre-defined categories. Support Vector Machine(SVM) is a classification algorithm that learns decision boundaries, also called hyperplanes, in the feature space of the data set. These hyperplanes effectively separate data in a feature space and therefore, can classify based on which hyperplane side a particular feature set lies.

As the pre-trained models are evidently better feature extractors for this dataset, we can use these features to train a SVM Model. Figure 20 and 21 show the confusion matrix, with VGG19 and ResNet50. Table 11 and 12 (Appendix) show the Precision per class for VGG 19 and ResNet_50, respectively. Before training, the data set was randomised and divided into 502 training image (85%) and 89 testing images(15%).

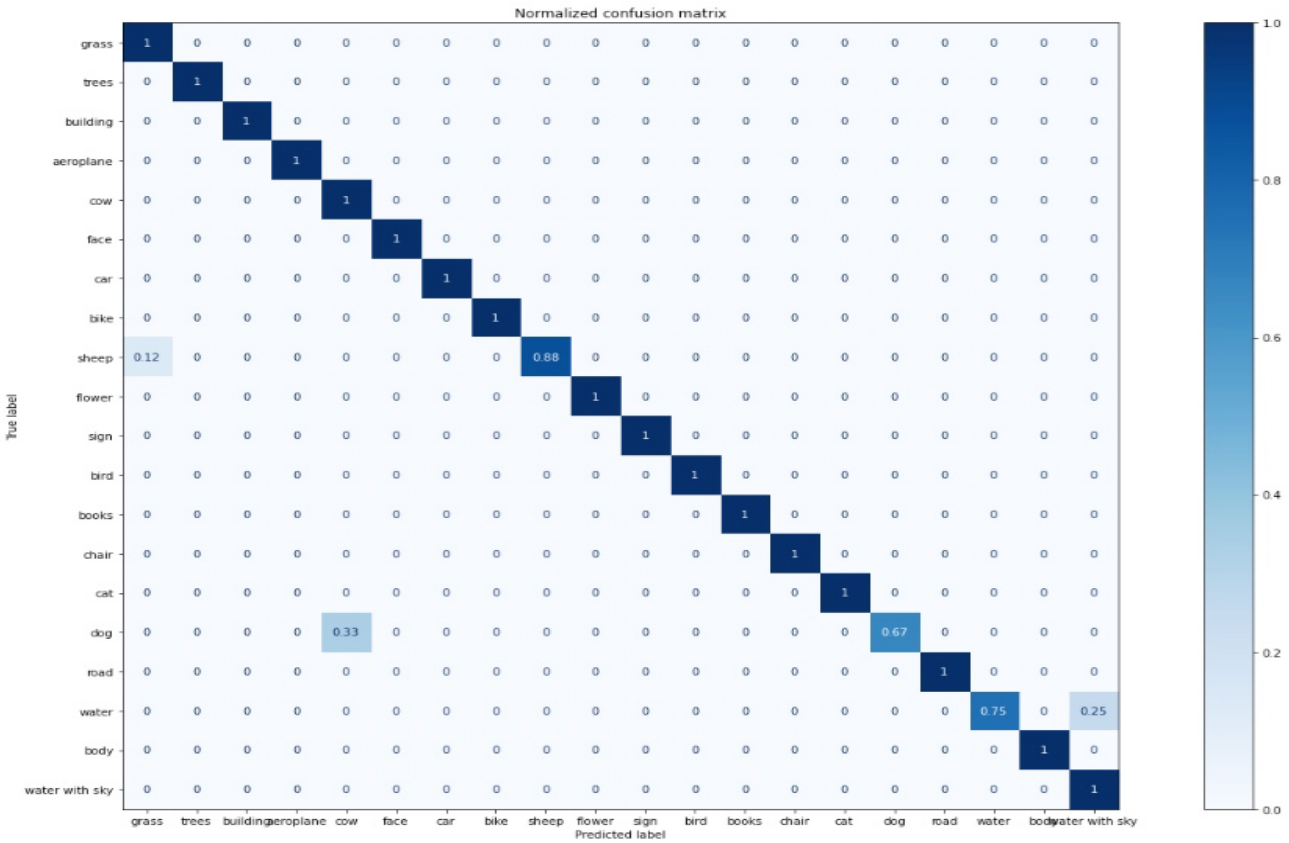

Figure 20 Confusion Matrix of SVM classification with ResNet Features

The Mean Average Precision of VGG 19 and ResNet 50 is 83% and 93% respectively (Appendix, Table x and y). Although, the test dataset was not large, these results do indicate two things. First, reinforcing that pre-trained models are good feature extractors of images. Second, Machine Learning is an effective method for classification.

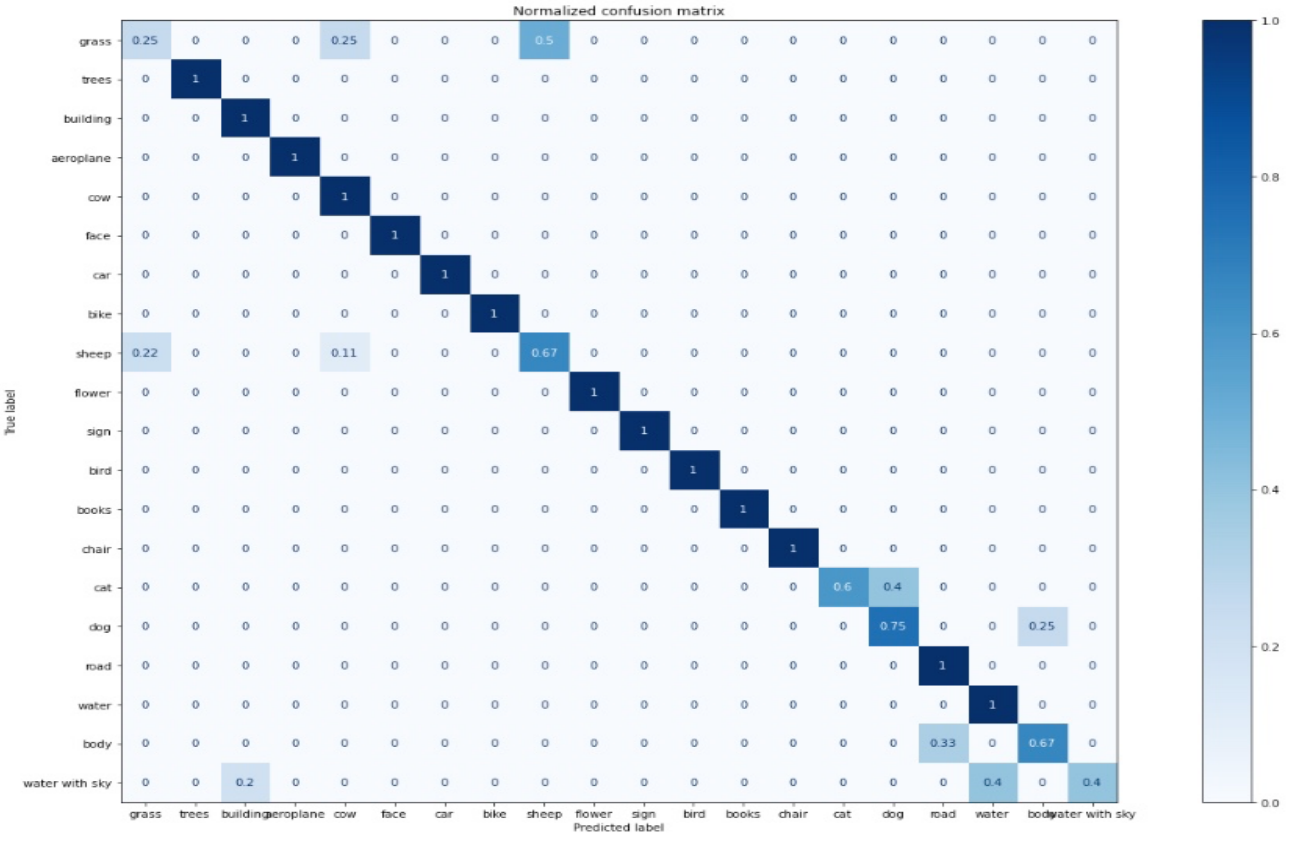

Figure 21 Confusion Matrix of SVM classification with VGG19 Features

Conclusion

While Visual Search is an effortless task for humans, it's still a difficult problem in computer vision. Basic Feature Extractors like Colour Histograms and Texture Descriptors are only good to extract local and specific properties and do not generalize well over varied and massive datasets. In this experiment, SIFT didn't show promising results although it has been claimed to perform impressively. More experimentation with different parameters might be required.

Transfer Learning saw a big leap in the Mean Average Precision compared to other descriptors. This exhibits the effectiveness of Convolutional Neural Networks as Feature Extractors and Deep Learning's capability to generalize from massive datasets. Furthermore, training Support Vector Machines with the features extracted from the pre-trained models also showed promising results for Image Classification, albeit on a small testing dataset.

Combining features from SIFT and CNNs might be one path of experimentation to pursue for Visual Search. Additionally, Principal Component Analysis might be necessary on CNN Features to faster computation. CNNs are the best performing methods[6] for Image Classification and therefore attaining better results can be expected by training with Dense Layers.

References

[1] On the Surprising Behavior of Distance Metrics in High Dimensional Space , Charu C. Aggarwal et al, IBM Watson Research Center
[2] Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan, Andrew Zisserman, University of Oxford
[3] Deep Residual Learning for Image Recognition, Kaiming et al, Microsoft Research
[4] Distinctive Image Features from Scale-Invariant Keypoints , David G. Lowe, University Of British Columbia
[5] A Comparison of Semantic Similarity Methods for Maximum Human Interpretability, Pinky Sitikhu et al, Tribhuwan University
[6] ImageNet Classification with Deep Convolutional Neural Networks, Alex Krizhevsky et al, University Of Toronto
[7] Study of K-Means and Enhanced K-Means Clustering Algorithm, Dr. S.P. Singh et al, International Journal of Advanced Research in Computer Science
[8] Manhattan Distance, Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology

Appendix

Average Precision Tables

Class	Manhattan	Euclidean	Cosine Similarity
1	8.34	6.77	5.84
2	31.56	19.01	34.21
3	6.20	4.40	4.81
4	20.66	9.12	12.08
5	13.36	9.63	7.91
6	12.64	11.26	5.85
7	16.42	19.38	13.17
8	13.66	16.15	14.44
9	22.35	14.52	15.81
10	10.31	9.44	7.63
11	7.76	5.46	7.82
12	5.21	5.99	6.48
13	14.99	17.99	17.53
14	9.36	5.64	6.22
15	10.15	11.76	11.67
16	7.19	6.33	6.74
17	20.99	12.64	23.13
18	7.08	5.36	6.70
19	5.28	4.66	7.33
20	9.94	8.21	7.97
MAP	**12.67**	**10.19**	**11.17**

Table 7 - Average Precision with RGB Histogram of Quantization 8 taken over top 20 results

Class	Manhattan	Euclidean	Cosine Similarity
1	7.60	6.81	5.87
2	49.86	31.32	32.80
3	4.71	3.79	5.27
4	37.60	30.00	31.55
5	6.37	5.92	6.36
6	11.18	8.42	9.57
7	16.12	13.86	10.93
8	6.94	10.34	6.23
9	14.94	10.23	5.91
10	8.35	7.45	6.24
11	10.66	7.65	10.02
12	9.21	6.36	7.22
13	37.51	36.15	27.91
14	4.03	4.95	4.05
15	6.52	6.66	6.49
16	4.78	4.64	3.70
17	13.11	9.37	8.27
18	6.52	5.17	6.25
19	5.24	7.13	6.26
20	13.73	9.09	6.95
MAP	**13.75**	**11.27**	**10.39**

Table 8 - Average Precision per class with Texture Descriptor Grid Size 25X25 and Bin Orientations 30

Class	Manhattan	Euclidean	Cosine Similarity
1	20.94	18.72	17.14
2	42.83	59.49	65.95
3	21.19	36.89	52.27
4	33.63	58.74	66.67
5	16.57	28.90	40.58
6	18.74	45.72	66.67
7	16.04	28.94	55.80
8	36.83	63.54	66.67
9	13.93	32.14	35.56
10	30.44	43.76	61.72
11	11.36	32.54	57.30
12	13.57	15.75	32.56
13	29.77	43.59	63.97
14	29.49	40.38	65.39
15	7.78	22.89	47.21
16	4.80	5.92	14.72
17	7.72	14.32	53.46
18	4.96	8.72	16.19
19	3.33	3.67	36.06
20	15.98	16.42	24.93
MAP	**19.00**	**31.05**	**47.04**

Table 9 – Average Precision per class with ResNet 152 features

Class	Manhattan	Euclidean	Cosine Similarity
1	21.72	30.35	16.58
2	43.73	56.42	63.59
3	25.90	41.50	53.31
4	33.81	60.72	66.67
5	15.11	19.20	34.97
6	14.79	41.14	66.26
7	21.01	24.46	57.71
8	29.82	52.89	66.67
9	15.26	22.98	33.04
10	32.35	46.13	59.80
11	14.95	35.76	63.51
12	10.76	12.48	31.35
13	37.90	46.59	60.93
14	21.81	30.25	51.69
15	5.92	19.17	39.46
16	4.70	6.00	17.60
17	11.06	14.85	52.01
18	5.99	8.05	16.03
19	3.41	4.11	39.54
20	16.09	18.81	28.62
MAP	**19.30**	**29.59**	**45.97**

Table 10 – Average Precision per class with Resnet 50 features

Class	precision	recall	f1-score	support
1	0.5	1	0.67	1
2	1	1	1	2
3	1	1	1	4
4	1	1	1	4
5	0.8	1	0.89	4
6	1	1	1	4
7	1	1	1	2
8	1	1	1	5
9	1	0.88	0.93	8
10	1	1	1	6
11	1	1	1	5
12	1	1	1	3
13	1	1	1	10
14	1	1	1	3
15	1	1	1	3
16	1	0.67	0.8	3
17	1	1	1	5
18	1	0.75	0.86	8
19	1	1	1	8
20	0.33	1	0.5	1
Average	0.93	0.96	0.93	
Accuracy			0.96	Total 89

Table 11 – Average Precision Per Class for Image Classification with SVM from ResNet 50 features

Class	precision	recall	f1-score	support
1	0.33	0.25	0.29	4
2	1	1	1	3
3	0.5	1	0.67	1
4	1	1	1	4
5	0.6	1	0.75	3
6	1	1	1	5
7	1	1	1	6
8	1	1	1	5
9	0.75	0.67	0.71	9
10	1	1	1	7
11	1	1	1	3
12	1	1	1	4
13	1	1	1	4
14	1	1	1	7
15	1	0.6	0.75	5
16	0.6	0.75	0.67	4
17	0.5	1	0.67	1
18	0.75	1	0.86	6
19	0.67	0.67	0.67	3
20	1	0.4	0.57	5
Average	0.83	0.87	0.83	
Accuracy			0.85	Total 89

Table 12 - Average Precision Per Class for Image Classification with SVM from VGG 19 features