VLAD for Large Scale Image Retrieval

Subhankari Mishra

December 24, 2015

Abstract

The objective of this project is to perform 'Large Scale Image Retrieval' efficiently by experimenting with the various implementations namely 'Bag of Visual Words' and 'Vector of Locally Aggregated Descriptors (VLAD)'. It involves implementation of the baseline method which is the 'Bag of Visual Words' and comparing the results of the baseline implementation with VLAD implementations.

1 Introduction

The goal of 'Large Scale Image Retrieval System' is to identify similar images for a given query image in a large set of images. In order to find the similar images, there is a need to represent an image in a meaningful format on which some calculation can be performed to find the nearest image(s).The basic way of representing an image is representing it with its pixel values. It is observed that the approach of comparing the pixel values to find similar images results has a very poor accuracy as it only considers the 'spread' of colors over the images and ignores all other aspects of the images. Some of the other approaches are Color Histogram, 'GIST', etc. which gives the global descriptor for an image. Global descriptors often fail to have high accuracy due to scaling, occlusion,lighting and view-point.
So, a 'Large Scale Image Retrieval' system converts an image into a mathematical representation such that similar images have similar representations and dissimilar images have dissimilar representations. The above mentioned goal has three sub goals. First is the need of invariant representation of images to over the robustness to view point, lighting, occlusion, intra-class variability etc. Second is the representation should be informative. Third, the representations computation should be efficient, as the size should be small so that it can be stored in RAM, and should be easy to process e.g. for fast comparison between images. To achieve the first sub goal i.e. invariant representation we would be using 'Scale Invariant Feature Transform (SIFT)' descriptors to extract local descriptors from the images. In this project, the focus is on the third sub goal of achieving higher accuracy with compact representation of the images which is sufficiently efficient for large-scale processing which includes the cost of computing the representation, cost of learning the classifier on these representations and the cost of classifying a new image.

2 Motivation

Image retrieval has been a key topic of interest among the researchers in the field of computer vision. Initially the focus was to classify images with the best possible accuracy. But with the enormously increasing data in recent years, the computational cost has become a bottleneck. So there is an urgent need to look for ways to make the application computationally efficient without loosing on the accuracy.

3 Performance Criterion

The performance would be measured by measuring the 'Mean Average Precision (mAP)' for the VLAD representation of the images encoded for different sizes of clusters and comparing the same to the 'Bag of Features' representation.
Precision is the fraction of retrieved documents that are relevant to the query.

$$\text{Precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{Recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

mAP provides a single-figure measure of quality across recall levels. If the set of relevant documents for an information need $q_j \in Q$ is $\{d_1, \ldots d_{m_j}\}$ and R_{jk} is the set of ranked retrieval results from the top result until you get to document d_k, then

$$\text{mAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk})$$

When a relevant document is not retrieved at all, the precision value in the above equation is taken to be 0. For a single information need, the average precision approximates the area under the uninterpolated precision-recall curve, and so the MAP is roughly the average area under the precision-recall curve for a set of queries.

4 Problem formulation

'Large Scale Image Retrieval' raises memory and efficiency problems as the mathematical computations were running over high dimensional images which makes it difficult to perform the computation while keeping the entire data-set in memory. This makes the system very slow and inefficient for real-time computations. To resolve this, an image should be represented in such a way with which matching algorithms can be run efficiently. In this project, the state of art - 'Bag of Visual (BOV)' representation needs to be compared with the VLAD representation.
BOV representation converts the image to a very low dimension which is efficient for real-time computation but BOV are informative only when the number of visual words i.e. the size of the vocabulary is large. To resolve this problem, the representation should be informative even with small vocabulary, which is given by the VLAD representation.

5 Approach

The model is trained with a set of input images, which returns a ranked list of similar images for a given query image . A set of input images are converted to a lower dimensional representation using the approaches described later and are stored for further computation. The query images are also brought down to similar representations and are compared with the input set to identify the similar images by computing the nearest neighbors.

The first approach i.e. the 'Bag of Visual Words (BOV)' is similar to 'Bag of Words' concept in the field of information retrieval. In this approach, each image is considered as a document containing multiple visual words. There are multiple ways of extracting information/features from a given image. But most of the approaches fail to provide good results due complexity of images available with respect

to robustness to viewpoint, lighting, occlusion, intra-class variability. To overcome this, an invariant representation of the image is required. To achieve this, 'Scale-Invariant Feature Transform (SIFT)' descriptors are used to represent an image. SIFT identifies the key points in an image and then transforms each key point into a 128 Dimensional vector. So an image is represented by a vector with dimension 'number of key points' X 128. The image patches are converted to visual words by clustering them into groups of similar patches and considering each group of patch as 'one' word. A patch is assigned to a word by computing the closeness of this patch to another patch. Here 'K-means' clustering is used to build the code book. The training and test set are the images represented by the word frequency in each category of the code book for each image i.e. the histograms for the word frequencies in each image.

The second approach i.e. 'VLAD representation' is the focus of this project. In VLAD, vocabulary is learnt and each cluster is assigned to its nearest centroid similar to BOV. Then, the alogorithm records the residuals i.e. the difference of each local descriptor from the assigned cluster center. The residuals are accumulated for each cluster and later the k sum of residuals are concatenated into a single k X 128 dimensional descriptor. Each element of an unnormalized VLAD is sign square rooted (i.e. an element x_i is transformed into $\text{sign}(x_i)\sqrt{|x_i|}$) and the transformed vector is L2 normalized.

6 Algorithms

6.1 Bag of Visual Words

- Retrieve the SIFT descriptors denoted as $(\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{n})$
- k-means:Partition the n set of descriptors into $\mathrm{k}(\leq \mathrm{n})$ sets $\mathrm{S} = \{\mathrm{S}_1, \mathrm{S}_2, ..., \mathrm{S}_\mathrm{k}\}$ and minimize the with-in cluster sum of squares i.e.
  $\text{argmin}_S \sum_{\mathrm{i}=1}^{\mathrm{k}} \sum_{\mathrm{x}\in\mathrm{S}_\mathrm{i}} \parallel \mathrm{x} - \mu_\mathrm{i} \parallel^2$ where μ_i is mean of points in S_i.
- Assign the local descriptor to the nearest neighbor by computing
  $\text{argmin} \parallel \mathrm{x} - \mu_\mathrm{i} \parallel^2$
- For each visual word compute the number of local descriptors assigned to it.

6.2 VLAD

Given a codebook learned with k-means $\{\mu_\mathrm{i}, \mathrm{i} = 1, ..., \mathrm{N}\}$
and a set of local descriptors $\mathrm{X} = \{\mathrm{x}_\mathrm{t}, \mathrm{t} = 1, ..., \mathrm{T}\}$:

- Assign the nearest neighbor:$\mathrm{NN}(\mathrm{x}_\mathrm{t}) = \text{argmin}_{\mu_\mathrm{i}} \parallel \mathrm{x}_\mathrm{t} - \mu_\mathrm{i} \parallel$
- Compute the difference from the cluster center: $\mathrm{v}_\mathrm{i} = \sum_{\mathrm{x}_\mathrm{t}:\mathrm{NN}(\mathrm{x}_\mathrm{t})=\mu_\mathrm{i}} \parallel \mathrm{x}_\mathrm{t} - \mu_\mathrm{i} \parallel$
- Concatenate all the v_i and apply l_2 normalization.

7 Implementation

7.1 Dataset

Inria's Holiday Dataset was used for this project. The data-set is a set of images which mainly contains some of the researchers personal holidays photos and some of them were taken on purpose to test the robustness to various attacks: rotations, viewpoint and illumination changes, blurring, etc. The data-set includes a very large variety of scene types (natural, man-made, water and fire effects, etc) and images are in high resolution. The data-set contains 500 image groups, each of which represents a distinct scene or object. The data-set contains the below materials.

- the images themselves i.e. 1491 images.
- the set of descriptors extracted from these images
- a set of descriptors produced, with the same extractor and descriptor, for a distinct dataset (Flickr60K).
- two sets of clusters used to quantize the descriptors. These have been obtained from Flickr60K.
- some pre-processed feature files for one million images

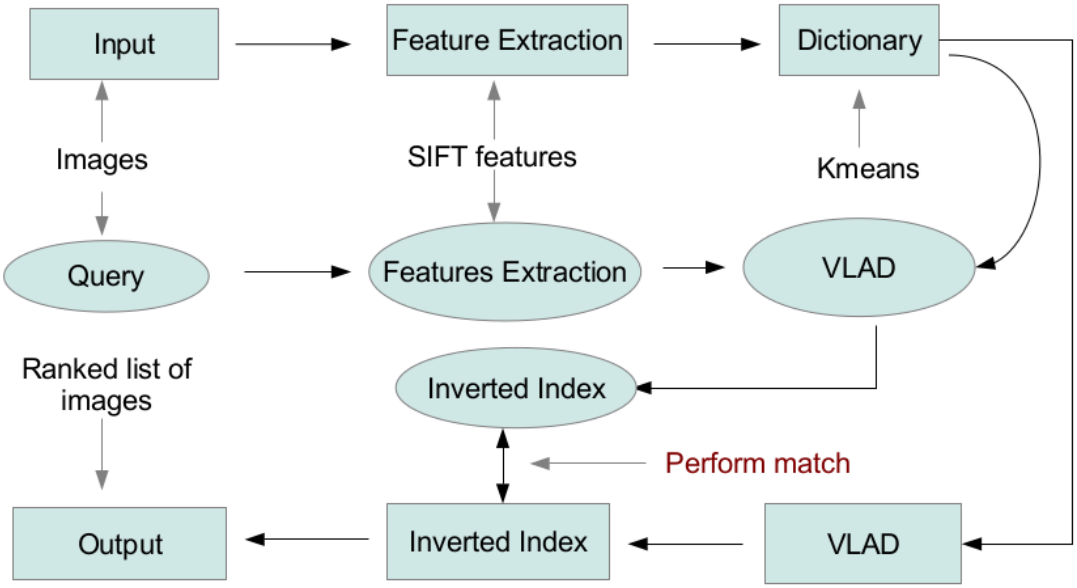
For this project the images themselves would be used.

- **format**: The data has 1491 JPEG images grouped in 500 groups.
- **size of data** The size of the data is 2.7 GB.
- **description** The file-name for the images are numeric and unique. The first image of a group is depicted by name xxxx00.jpg. Images in the same group have the same file-name till the fourth place.

7.2 Data Flow / Pipeline

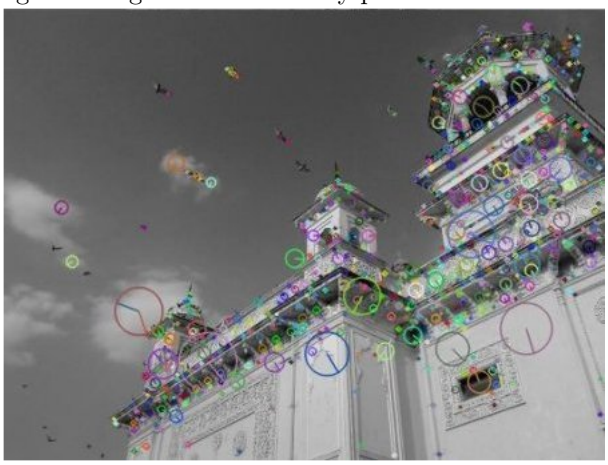
As a first step, the raw images need to be converted to any representation which is computation friendly. To achieve this, Local Descriptors were extracted from the images. The extracted descriptors were used to learn the vocabulary which, in turn, were used to build the BOV representation and VLAD representation for the images. The BOV and VLAD vectors were stored in the inverted index format with key as the words and column number respectively. The query image were also converted to similar representation where the vocabulary used for the building the representation was the one that was learnt for the training set. To identify the similar images, the query images were compared to the stored lower dimensional training set images and a ranked list of similar images were returned.

The outer flow in the below diagram shows the training data flow and inner flow represents the same for the test set.



- **Feature Extraction**:
  Local descriptors for each image were extracted using VLfeat's 'Scale Invariant Feature Transform (SIFT)' algorithm. The feature extraction is implemented in Matlab. The raw images are read using 'imread' function in matlab. As the SIFT algorithm expects the input image to be grayscale single precision. The input RGB image is converted to grayscale using 'rgb2gray' function in matlab and later to single precision using 'single' function. The SIFT algorithm identifies the local key-points in an image and returns the frames and descriptors for the same. Each column of the frames returned is a feature frame and has the format [X;Y;S;TH], where X,Y is the (fractional) center of the frame, S is the scale and TH is the orientation (in radians). Each column of returned descriptors is the descriptor of the corresponding frame in the returned frames. A descriptor is a 128-dimensional vector.
  The SIFT algorithm identifies the keypoints by creating multiple images for image input given for multiple octaves and scales and adding Gaussian blur incrementally to each successive image in each scale under each octave. The difference of Gaussuans in calculated for each pair of scales in each octave. The identified keypoints are the extrema identified in these difference of Gaussian images.
  After feature extraction each image is represented by < number of local decriptors > x 128 dimensions.

  Below is a Sample Image showing the identified key-points:



- **Whitening and Principal Component Analysis (PCA)**:
  The local features extracted from each image were power-normalized by applying $\text{sign}(z)|z|^{\alpha}$ to each element in the local descriptors with $0 \leq \alpha \leq 1$. The power normalized features were inturn L_2 normalized by using numpy.linalg.norm by applying the below equations.

$$\text{v} = \text{numpy.linalg.norm(data)}$$

$$\text{data} = \frac{\text{data}}{\text{v}}$$

  Before passing the data to the clustering algorithm to learn the vocabulary, PCA is performed to reduce the dimension further so that the clustering algorithm is faster. This is due to the reason that it performs clustering on lower dimensions as compared to the original data. It is believed that performing PCA, while preserving 95 percent variance, gives good results. The number of components that contribute to the 95 percent of variance were identified by getting the explained variance for all components, dividing the explained variance by sum of the explained variance and them computing the number of components for which the sum would be 0.95.

$$\text{pca} = \text{PCA()}$$

$$\text{pca.fit(data)}$$

$$\text{var} = \text{pca.explained\_variance\_}$$

$$\text{sum} = \text{sum(var)}$$

$$\text{new\_var} = \frac{\text{var}}{\text{sum}}$$

Compute the cumulative sum of new_var till the sum reaches 0.95. The number of components the number of elements contributed to this sum. The total sum of new_var would be 1.
This was performed using Sci-kit Learn's PCA. The normalized features for each image is projected using the PCA learned above.

- **Learning the Vocabulary**: The local descriptors are passed to the clustering algorithm to learn the vocabulary. The K-Means clustering algorithm aims to partition the data into K partitions, where K is the user input, by computing the distance of each data-point to the cluster centers. There are various implementations of K-Means algorithm which varies in the way each initializes the centroids which has a huge impact on the convergence of the algorithm. The Sci-kit Learn's Online Mini Batch K-Means algorithm is used for this project. This is due to the fact that Mini Batch K-means is faster as compared to K-Means or K-Means++ algorithms. The centroids obtained from the algorithm are returned after the vocabulary is learnt.
  In the online mini-batch k-Means, a random small batch of the input data is selected and the cluster centers are updated in small steps for each of the random samples picked. The mini-batches tend to have low stochastic noise and do not suffer increased computational cost when data sets grow large with redundant examples.

- **Build BOFs**: The BOF feature is built using the centroids obtained from the above clustering algorithm and The local features for each image in the data set using the algorithm described in section 5 and 6.1.

- **Build VLADs**: The VLAD feature is built using the centroids obtained from the above clustering algorithm and The local features for each image in the data set using the algorithm described in section 5 and 6.2.

- **Build Inverted Index** The obtained BOVs and VLADs were stored as inverted index for further processing. The inverted index approach saves more on memory consumption and makes the computation faster by looking at the descriptors vertically i.e. by words vs documents while comparing instead of horizontally i.e. documents vs words.

- **Scoring and Ranking** The score were calculated by performing a dot product between the query image/vector to the training set of images/vectors. The scores obtained for each query image against the training set are sorted in descending order.
  The top 10 ranked images are returned as suggestion.

8 Baseline Method

The baseline for this project was to compute and evaluate the experimental results for the nearest neighbor search on the BOV representations available in the data-set.

8.1 Data Set

As the Holiday Data set did not had the Bag of Features(Histogram) representation for the images which was needed for this baseline implementation. Ungulate Data set was used for the experiments for the baseline as those have the histogram representation for each image which can be immediately taken as input for the nearest neighbor search.

- **format**:The data has four files two ivecs file and two fvecs. The each fvecs file 4096D Fisher descriptors for all classes of the Fungus group i.e. 134 classes with 50 example per class.
  The data in each fvecs file has a dimension of 4096 X 6700.
  The data in each ivecs file contains the label for each corresponding image in the fvecs file.
- **size of data** The size of the data is 200 MB. The vectors have floating point value. Each value in the vector needs 4 bytes or 8 bits. So total number of bits needed to store one of these vectors of size 4096 X 6700 is $878182400bits \approx 109772800bytes \approx 107200KB \approx 107.2MB$.
- **description** The was extracted using from the binary format using conversion scripts provided with the data set. The data was normalized to using power-law normalization and l2-normalization during the computation.

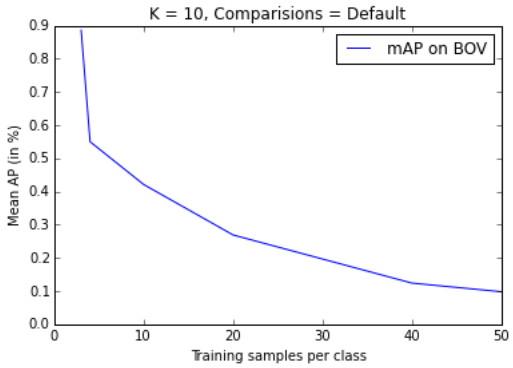
8.2 Evaluation

For the purpose of evaluating the existing implementation and to define our baseline, we performed the below experiments on the extracted data.

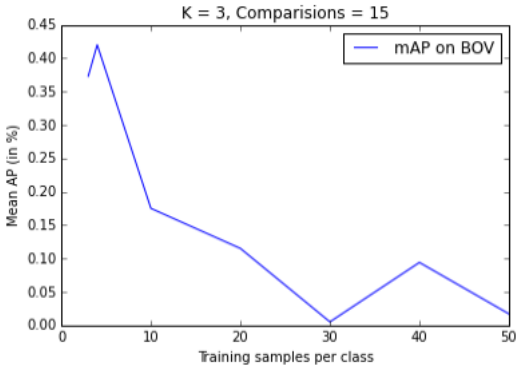
- Randomly selected the training samples and query samples from the available image samples.
- Normalized the data using power-law normalization and then l2-normalization.
- Built the kd-tree using VLfeat kdtreebuild.
- Then queried the kd-tree to identify the nearest neighbors in the selected training samples for the queried images.
- After receiving the indexes for the nearest neighbors prepared the result file for the evaluation process by aligning the ranked list of images against each queried image.
- The Evaluation code available in the Holiday Data set was available to compute the Mean Average Precision over the queried results for the specific training set and query set used for a particular experiment. The Evaluation code had to be changed a little bit as per our requirement so that the relevant results can be classified by comparing if the queried image and the result image belong to the same class.

8.3 Experiments

- **Number of Neighbors = 10, Number of Comparisons = Default**: The Approximate Nearest Neighbor(ANN) search was carried out by randomly selecting equal number of training and query samples for each class and calculating the Mean Average Precision(mAP) for the result of each run. The ANN was configured to retrieve 10 nearest neighbors by making any number of comparisons which the default value. The number of training and query samples considered were 3,4,10,20,40 50. The below table shows the values for the number of samples and mAP in percentage.



- **Number of Neighbors = 3, Number of Comparisons = 15** A similar experiment was run with configuration set to 3 neighbors and 15 max comparisons. The results are as below.

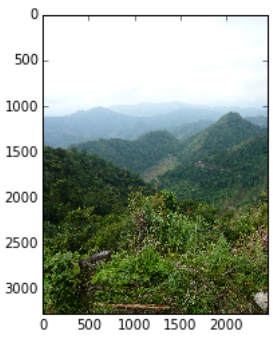


9 Experiments and Results

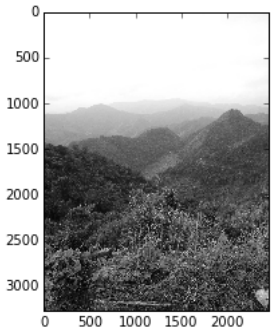
9.1 Feature Extraction

The SIFT algorithm on average extracted 15000 local descriptors per image.
Each local descriptor was 128 dimensions long.
It took approximately 22 hours to extract local descriptors from all 1491 images and write them to file for future reference.
The values for each element in local descriptor varied between 0 to 255.
Below are the input and result for the feature extraction phase.

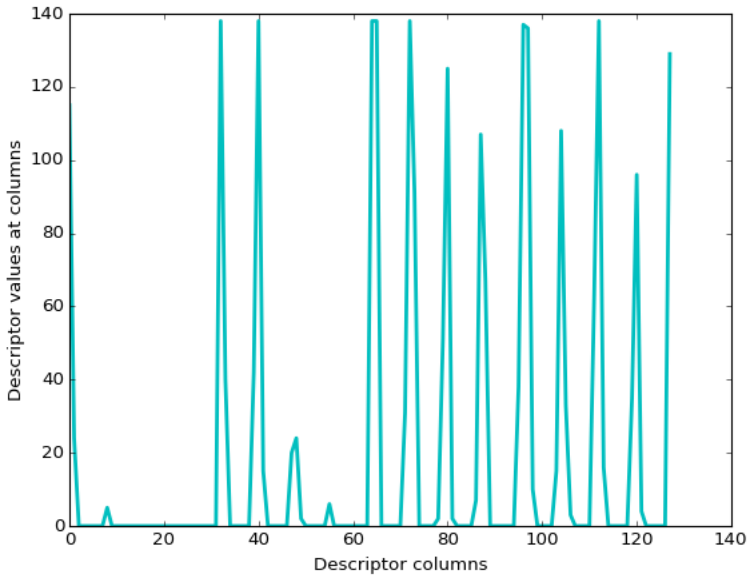
Input:



Gray-scale image:



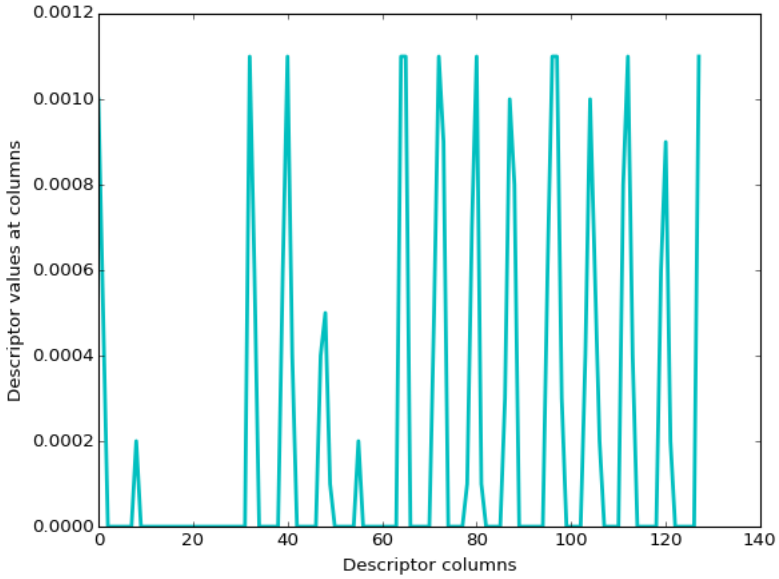
Un-normalized 128D feature representation of 1 local descriptor:



9.2 Whitening and Principal Component Analysis (PCA)

As the extracted feature values were between 0 to 255, which are not considered good for mathematical computations, there was a need to bring those down to lower values i.e. between 0 and 1. In addition to that, the process of normalization would make the dominant features be less dominant, thereby giving a bit higher values to the recessive features so that the model is not biased. The process of normalizing the data took close to 10 hours with two jobs running in parallel. The below image describes a local descriptor after power law normalization and L_2 normalization.

Normalized 128 feature representation of 1 local descriptor:

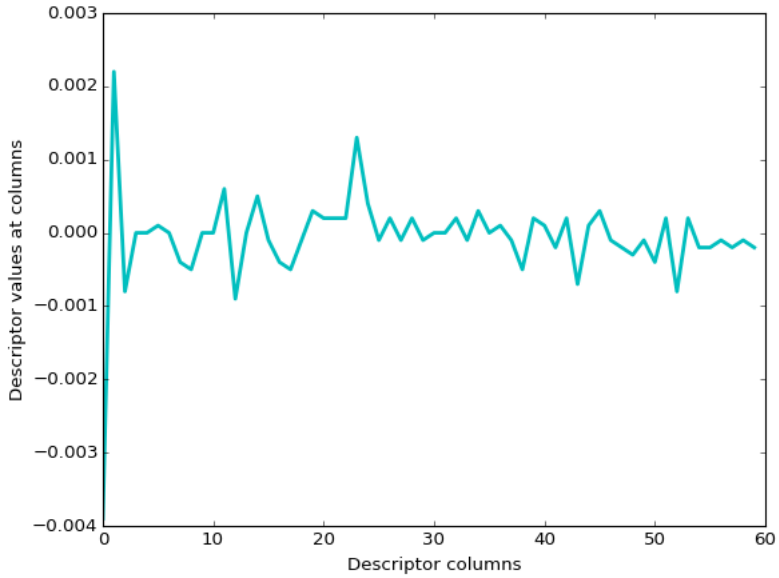


As the entire data-set cannot be stored in memory at once, in order to perform PCA, 100 random local descriptors were selected from each image. The processing of selecting random local descriptors from each image and writing them to a file took approx 8 hours.

Once the random points were extracted, the PCA was performed and all the local descriptor for all images were projected using the PCA learned from the random descriptors. The process of performing the PCA,projecting the data and writing the data-set took approx 16 hours.

The number of components which preserved the 95 percent of variance were found to be 60.

After PCA the feature representation of 1 local descriptor:



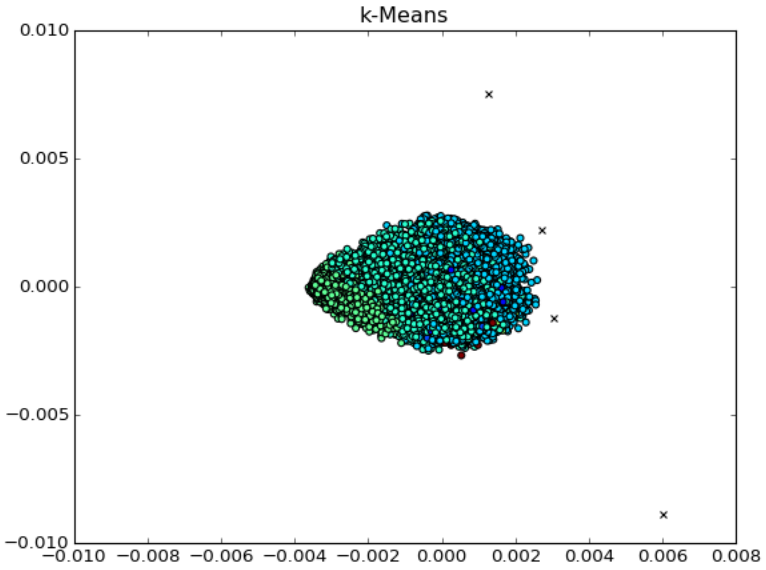
9.3 Learning the Vocabulary

Once the data was written back to files, it was split into train and test set.
The experiments were performed with two types of split.

- Random split: The data was randomly permuted and split at 50 percent. The first half was considered as the training set and second as test set. It was observed that with this split, there were many instances where there was no matching image or image of the same group in the training set leading to a very low accuracy.
- 1^{st} Image of the group as query image: Scanned through each image in the data-set, if the name of the image was xxxx00.jpg moved it to the test set. After this split, there were 500 test images and 991 train images.

The process of selecting random local descriptors from each image was repeated for k-Means. But in this case, the local descriptors were of lower dimension and were selected only from the training set. For k-Means 1000 local descriptors were selected from each image. The process took approx 5 hours. The online mini-batch k-Means was run for various configurations with K values randing from 4 to 1024.
The extraction of centroids with 1000 iterations for the largest k value i.e. 1024 took approx 4 with 20 GB memory and 12 compute nodes. Whereas, the same process for 10000 iterations took approx 24 hours with 1 compute node and 10GB memory. The execution with default configuration took 20 hours with 12 compute nodes and 20 GB memory. The obtained centroids were written to file for future reference. Below plot shows the clustered points for one image for K = 16.

Clustered local descriptor for 1 image:



9.4 Build BOVs

The BOVs were computed for each image in the training set using the cluster centroids obtained in the previous step for each k value following the algorithm described in section 6.1.
Similarly, the BOVs were also computed for the test images but using the cluster centroids obtained in the above step from the train images.

The process took approximately 4.8 hours for the training set and 2.4 hours for the test set with two jobs running in parallel with 12 compute nodes each and 20 GB memory.
The obtained BOV vectors were k dimensional.
All the vectors obtained for train and test set were written to two separate respective files for further processing.
The longest BOV vector i.e. 1024 dimension consumes 7.5 KB for each image.
So we reduced the memory used to represent an image from 4MB to 7.5KB per image which would be really efficient for computation.

9.5 Build VLADs

A similar process was repeated for computing VLAD vectors for the images following the algorithm described in section 6.2.
The process took approximately 5.5 hours for the training set and 2.7 hours for the test set with two jobs running in parallel with 12 compute nodes each and 20 GB memory.
The obtained VLAD vectors were k x D dimension i.e. for k = 16 and D = 60, the obtained vectors were 960 Dimensional. The longest VLAD vector i.e. with k = 1024, 61440 dimension consumes 429 KB.

9.6 Build Inverted Index

The BOV and VLAD vectors for the test set and train set were transposed and stored in sparse vectors to save more on the memory consumption.
scipy.sparse.csr_matrix were used to build the sparse matrices. The process took less than 30 minutes for the longest vector i.e. VLAD representation with k = 1024.

9.7 Scoring and Ranking

For scoring the dot product was performed between the sparse vectors obtained in the previous step. The top 10 highest scored images were returned as suggestion for a given query.
The results were written to a file in the below format.

132101 0 132100 1 132102 2 110501 3 109201 4 142500 5 144600 6 124401 7 124500 8 132701 9 135200

Where the first name is for the query image and the rest are the ranked list of suggestions with the rank preceding the suggested image name.
The scoring and ranking phase took approximately 40 minutes for the longest vector i.e. VLAD representation with k = 1024.

9.8 Memory Consumption

Single image:

raw image	Unnormalized features	normalized features	after PCA	BOV	VLAD
4 MB	11 MB	28 MB	13 MB	7.5 KB	429 KB

Complete Data Set:

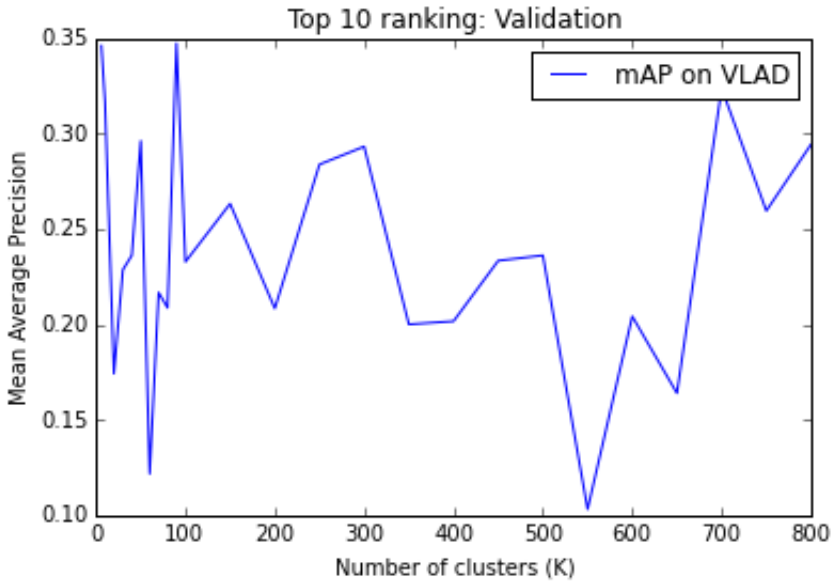
raw image	Unnormalized features	normalized image	after PCA	BOV	VLAD
2.7 GB	18.3 GB	45 GB	22 GB	9.7MB	626 MB

9.9 Evaluation

The Holiday's data set evaluation package was used for this project. The evaluation package takes a file with all the images available in the data-set and computes the ground truth by building a set of relevant list of images for each group of images by comparing the first four digits of the image name. Given a result data written in the format mentioned in the Scoring and Ranking section, it computed the mean average precision for the result by comparing it to the ground truth to identify the relevant images for a given query.

9.10 Results

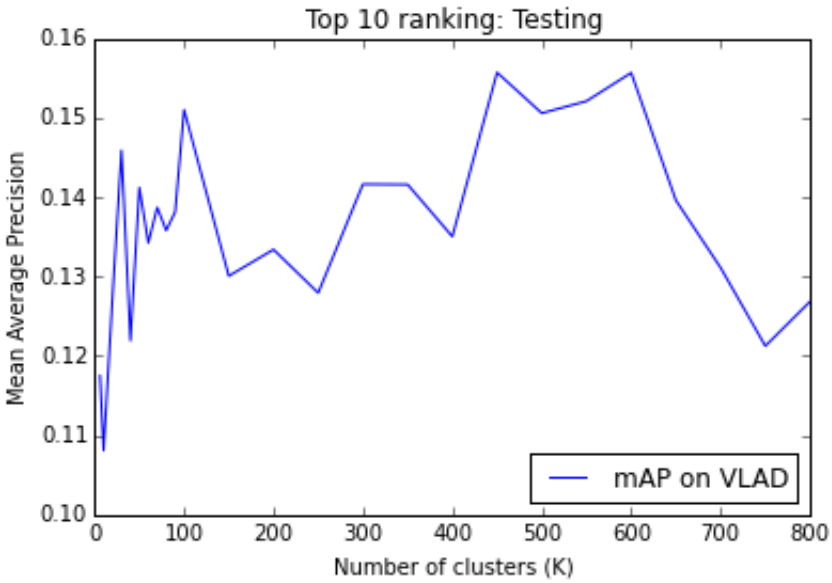
- Configuration: Validation test with Random split for test and train image.
  745 train and test image each.
  Number of clusters varying from 6 to 800.
  Batch size of k-Means = 100
  Max_iter for k-Means = 1000
  PCA performed with 50 random points from each image.
  Number of components for 95 percent variance = 75
  Vocabulary learnt with 50 random points from each image.
  10 VLAD vectors chosen randomly from the training set.



Observation: The maximum mAP obtained was 35 percent at k = 90 and 6.

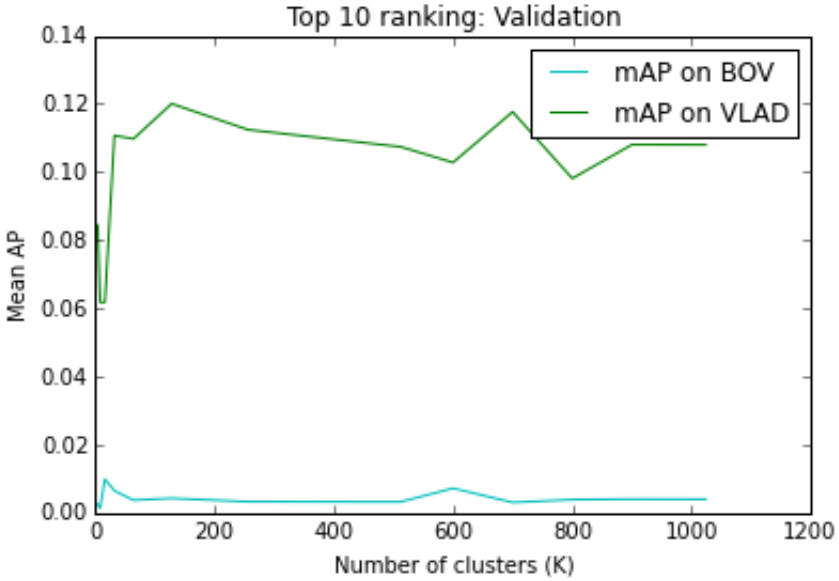
- Configuration: Test with test set with Random split for test and train image.
  745 train and test image each.
  Number of clusters varying from 6 to 800.
  Batch size of k-Means = 100
  Max_iter for k-Means = 1000
  PCA performed with 50 random points from each image.

Number of components for 95 percent variance = 75
Vocabulary learnt with 50 random points from each image.



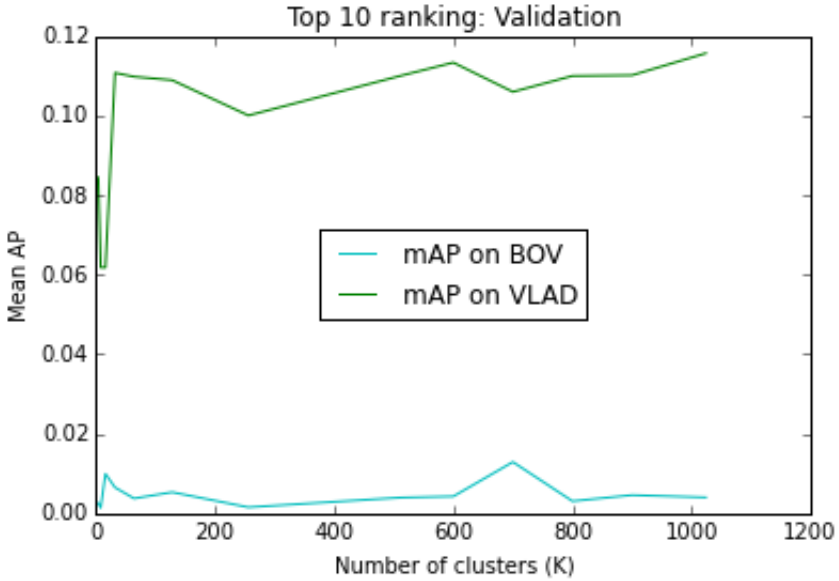
Observation: The maximum mAP obtained was 15 percent k = 450.

- Configuration: Test with 500 query images obtained by splitting the set with 1st image of the group as query image.
  991 train images.
  Number of clusters varying from 4 to 1024.
  Batch size of k-Means = default
  Max_iter for k-Means = 10000
  PCA performed with 100 random points from each image.
  Number of components for 95 percent variance = 60
  Vocabulary learnt with 1000 random points from each image.



Observation: The maximum mAP obtained was 11 percent with VLAD representation at k = 700. The BOV representation performs terribly bad as compared to the VLAD representation.

- Configuration: Test with 500 query images obtained by splitting the set with 1st image of the group as query image.
  991 train images.
  Number of clusters varying from 4 to 1024.
  Batch size of k-Means = default
  Max_iter for k-Means = default
  PCA performed with 100 random points from each image.
  Number of components for 95 percent variance = 60
  Vocabulary learnt with 1000 random points from each image.



Observation: The maximum mAP obtained was close to 12 percent with VLAD representation at k = 1024. The BOV representation performs terribly bad as compared to the VLAD representation.

10 Experimental Setup

- **Operating System** Both Windows 64 bit and Ubuntu 64 bit Operating Systems were used for the project. The first half of the project was implemented on Windows 64bit, 8GB RAM, Intel i5 processor, the second half was implemented on Linux(Ubuntu) with the same configurations due higher availability of soft-wares for Linux system and ease of installation.
- **Matlab**: Matlab version r2015b is used for these experiments on Windows and Linux 64 bit operating system.
- **Anaconda**: Anaconda bundle has Scikit-Learn, Ipython Notebook and all dependencies which makes the set up a really easy task.
- **VLfeat**: The VLFeat open source library implements popular computer vision algorithms specializing in image understanding and local features extraction and matching. Algorithms include Fisher Vector, VLAD, SIFT, MSER, k-means, hierarchical k-means, agglomerative information bottleneck, SLIC superpixels, quick shift superpixels, large scale SVM training, and many others. It is written in C for efficiency and compatibility, with interfaces in MATLAB for ease of use, and detailed documentation throughout.
- **Holiday's Evaluation Code**: For the purpose of evaluating the model, I used the Evaluation code available in the Holiday's data set.
- **HPC at NYU** The HPC cluster Mercer acted as a boon for this project for scheduling the long running jobs which would have been never possible on a personal computer. All the jobs starting from feature extraction to ranking and scoring were run on NYU HPC mercer with different configuration as required by each job.

10.1 Software Configuration

The installation of Matlab and Anaconda were very simple. VLfeat is installed through MATLAB by executing the mex file as discussed in the Read me file in the software package. VLfeat supports other tools also but for this experiment I had to use the MATLAB version.

11 Impediments and improvements

Initially, there was a lot of confusion regarding the data to be used for the project. I started with the precomputed features available in the Holidays data set which were 235 GB, so I wrote a script to download the data from the online directory.
Following a discussion in the classroom after a long time, I realised that I had to work with the real images instead of the features.
Once the features were extracted, I ended up getting more confused regarding the kind of normalization to be used. For this, I ran my entire pipeline more than 4 times before realizing that there was a bug at the first step of normalization. As a result, the experiments took really long time and I got plenty of partial results.
The unfamiliarity with HPC, Matlab and Python proved to be time consuming.
To improve the execution time of the experiments, standard pythons functions were used to perform

the Euclidean distance calculations and matrix operations.
Operations were vectorized as much as possible.
I planned to run some more experiments for different configurations, but the k-means clustering process is taking more than 36 hours and is still running. So, I can't proceed with those experiments.

12 Conclusion

It is observed from the experiments that VLAD representation gives higher accuracy as compared to the BOV representation even for smaller number of clusters and both of these representations are very efficient with respect to computation. Both the results were very low as compared to the reference paper.

13 Future Work

As the BOV performed poorly than VLAD, I plan to investigate further and try improving the same. In addition to above, VLAD representation did not perform as good as the reference paper. I intend to look into that aspect. As an improvement to the project, I plan to implement and compare the performance of Fisher Vectors representation as compared to VLAD and BOV.

References

[1] Jorge Sanchez, Florent Perronnin, Thomas Mensink, Jakob Verbeek, *Image Classification with the Fisher Vector:Theory and Practice*, International Journal of Computer Vision, Springer Verlag (Germany), 2013

[2] Relja Arandjelovic, Andrew Zisserman, *All about VLAD* , Department of Engineering Science, University of Oxford

[3] Herve Jegou, Ondrej Chum2, *Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening* ,INRIA Rennes,CMP, Department of Cybernetics, Faculty of EE, CTU in Prague