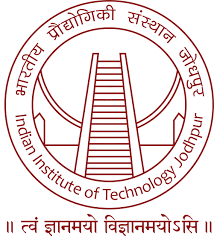
**Learning Document Structure**

**For**

**Retrieval And Classification**

Chinmay Garg (B16CS041) Anurag Shah (B16CS034)

Mentor: Dr Gaurav Harit

*Department of Computer Science and Engineering*

*Indian Institute of Technology, Jodhpur*

**Motivation**

The sole biggest motivation behind choosing this project was our fascination towards machine learning.

Machine Learning is the rave of the moment, as it is gradually becoming the brain behind business intelligence.

We wished to implement machine learning for document classification as it is more challenging due to handling of voluminous and highly non-linear data, generated exponentially in the era of digitization. Proper representation of documents increases efficiency and performance of classification, ultimate goal of retrieving information from large corpus.

And hence, in this project we will be implementing a already proposed algorithm for document classification.

**Objective**

To retrieve “similar” documents from a large heterogeneous collection of document images.

**Sub-Objective**

To classify the Arabic dataset provided on the basis that whether they contain a table or not.

**Challenges/Research Issues:**

1. Integration of OpenCV libraries with a C++ IDE.
2. Poor documentation of functions and classes implemented in OpenCV libraries on the official website of OpenCV.
3. Size of a matrix that can be constructed is limited (around 4700000 bytes). Thus, limited number of descriptors can be extracted from a limited number of images for codebook creation.
4. Very small dataset available for training and hence during testing on externally provided images, accuracy obtained was not up to the expectation.
5. The algorithm proposed in the provided paper – Horizontal Vertical Partitioning has almost non-existent documentation.
6. No proper reasoning given behind choosing a specific value behind any constant in this paper.

**Methodology/Algorithm:**

We have implemented our project in five major steps.

1. Codebook creation
2. Writing Features
3. Cross Validation
4. Training the Random Forest
5. Classification of Images
6. **Codebook creation**

Q) First of all, what is a codebook?

Codebook is a file which contains exemplary codewords which represent the basic structural elements of the images.

We used SURF method to obtain the 64-dimensional descriptors of the image. Compared to SIFT, SURF descriptors are several times faster and more robust to noise. We selected a small set of representative document images for extracting the 64-dimensional SURF descriptors. Finally, after extracting all the descriptors of all the images we had using K-Means method, codebook was generated.

1. **Writing Features**

In this step we compute features for each document image. We find the nearest (L1-norm) codeword in the codebook for each SURF descriptor extracted from an image. We then compute a normalized histogram of codewords in each region obtained by partitioning the image up to 3 levels which give us 13 partitions (which is obtained from the below formula).

N = +

N = number of partitions

H = level of partitioning of height-1(0 based indexing).

W = level of partitioning of width-1(0 based indexing).

1. **Cross Validation**

From the dataset we train the random forest with randomly chosen 300 images and test it with other left images. We repeat this process ten times. Now for each iteration we get the result whether the random forest tree has predicted correctly or not and, on that basis, we calculate the accuracy.

1. **Training the Random Forest**

Here we train our Random Forest using the histogram of all the images and the classes they belong to (that is to which they actually belong to).

1. **Classification of Images**

Here we classify the image as to which class it belongs to.

We extract the descriptors and keypoints of the image we want to test using the SURF method and store the histogram of the image. Using the trained Random Forest Tree formed in previous step and the histogram formed we classify the image.

**Algorithms we used.**

1. **K-Means Algorithm**

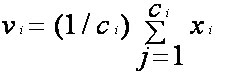
Let X = {x1, x2, x3, ... , xn} be the set of data points and V = {v1, v2, …….,vc} be the set of centers.

1) Randomly select *‘k’* cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.

4) Recalculate the new cluster center using:



where,*‘ci’* represents the number of data points in *ith* cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3).

1. **SURF (SPEEDED UP ROBUST FEATURES)**

SURF is an enhancement of SIFT algorithm which has the following four computational steps:

1. Extract keypoints.
2. Keypont localization.
3. Assigning orientations to each keypoint.
4. Generating descriptors for each keypoint.
5. **Random Forest Classifier**

Random Forests are an ensemble learning method (also thought of as a form of nearest neighbour predictor) for classification and regression that construct a number of decision trees at training time and outputting the class that is the mode of the classes output by individual trees.

**Results:**

1. For Hessian Threshold = 5000, maximum descriptors = 1000000,

On training data of 150 relevant and irrelevant images each,

When cross-validated on 56 relevant and 233 irrelevant data,

Accuracy = 97.2789%.

1. For Hessian Threshold = 8000, maximum descriptors = 10000,

On training data of 150 relevant and irrelevant images each,

When cross-validated on 66 relevant and 252 irrelevant data,

Accuracy = 96.8558%.

1. For Hessian Threshold = 60000, maximum descriptors = 50000,

On training data of 150 relevant and irrelevant images each,

When cross-validated on 56 relevant and 233 irrelevant data,

Accuracy = 93.5374%.

1. On manual testing of external images:

Relevant: 8/11 images classified correctly.

Irrelevant: 19/19 images classified correctly.

**Conclusion:**

The algorithm proposed in the paper was successfully implemented.

The accuracies obtained were nearly same as that expected in the paper (97.2%).

Efficiency of HVP and RFC were successfully tested.

More importantly we got our first brush with machine learning, document page segmentation, document classification and OpenCV libraries.

**References:**

https://docs.opencv.org/3.0-beta/doc/tutorials/tutorials.html

https://docs.opencv.org/3.1.0/d2/d75/namespacecv.html

https://docs.opencv.org/3.1.0/modules.html

https://docs.opencv.org/3.1.0/annotated.html

https://kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw

https://stackoverflow.com/questions

https://lampsrv02.umiacs.umd.edu/projdb/project.php?id=72

https://pdfs.semanticscholar.org/31da/8cace27a08e6339145f95f190a77197f01b2.pdf

https://www.quora.com/

https://github.com/opencv/

https://github.com/

Date: April 16, 2018 Mentor’s signature