Cost-Effective and Scalable Image Matching Across Heterogeneous Online Social Networks

Devakunchari R, Anna University, Chennai, India.
Valliyammai C, Anna University, Chennai, India.

Abstract

In this modern era, social media facilitates us in communicating with the people across the world. Sharing of photos on social networks is due to their addictive interest in receiving the likes from other users in the network to gain popularity. Users on social network upload nearly 1.8 billion photos every day. Malicious users try to gather the publicly available photos on social networks and use it for creating bogus accounts. To identify those anomalous users, the photos shared are collected and processed by the social network manager to classify the original person from the fake one. As there are billions of users in each social network, there are an enormous amount of photo uploads which leads to the problem of scalability, slower processing performance and execution speed. The primary objective of this paper is to identify the similar image for a given a query image on a large set of image datasets crawled from online social networks through the Internet. For handling the scalability incurred from large image data sets, the image matching computation is implemented in distributed computing Map Reduce framework. The face recognition involves supervised machine learning approach employing Computer vision algorithms, namely Fisher face, Eigenface and Local Binary Pattern Histogram (LBPH). The similarity functions are used to calculate the distance between the query image and image sequence in the Hadoop file system. Experiments use the trained data sets to find the least similarity measure. The results obtained show that LBPH provides better and accurate matching results compared to other two face recognition approaches.

Keywords: Apache Hadoop, Apache Pig, Machine Learning, Computer Vision

Introduction

Social Networks are gaining an exponential increase in popularity day by day. People show huge interest to spend all of their leisure time to stay back online on social networks. To be active on a social network is changed as a resemblance of the aliveness of an individual. It shows the current trend of all kinds of users towards the social network. To gain fame, people try to upload status, photos, and share posts instantly in social networks. People show off their presence by instantly uploading the current photos immediately into the social network. It is of their great intention to receive more likes from other users in social networks. People make use of the opportunity available in social networks to make the worst use of the information available on social networks. Anomaly users try to impersonate the publicly available photos and portrait it as their picture either in the same network or different network.There are many live cases of fake profiles, getting generated using the other photos, across heterogeneous social networks. People in LinkedIn, Facebook and Google+ have no standard options for discriminating fake user profiles. These profiles need to be identified to protect other users from the harmful effects caused by anomaly users. Several bogus accounts in social networks do not contain the original photo as their profile photo. The nature of a user is guessed using many of the features. Here, the profile photo plays a significant role in classifying the user profiles on social networks. The rest of the paper consists of relevant concepts in connection with the large scale processing of images using machine learning techniques, computer vision algorithms on Hadoop and its framework with obtained results.

Related Work

Huang et.al (2015) proposed a COX1 Face DB by a new benchmarking and related study based on a recently collected still/video face database. A novel Point-to-Set Correlation Learning (PSCL) method is proposed, and experimentally showed that was is used likely as a baseline method for V2S/S2V face recognition on COX Face DB. Gao (2015) targets learning robust image representation for single training sample per person face recognition using the supervised autoencoder. The autoencoder extracts the features which are robust to variations in illumination, expression, occlusion, and pose, and facilitates the face recognition. It is also applied for face verification. Experiments are executed on the AR, Extended Yale B, CMU-PIE, and Multi-PIE data sets. Test results show that by coupling with the commonly used sparse representation-based classification, the model outperforms other conventional models. A new face identification framework proficient of manipulating the full variety of pose variations within ±90° of yaw is proposed by Ding et.al (2015). The generated robust patch-based face representation system reproduces the synthesized partial frontal faces. For every patch, a transformation dictionary was learnt under multi-task learning system. Eventually, face matching is performed at the patch level. Nguyen et.al (2015) presented a novel feature extraction method named Local Patterns of Gradients (LPOGs) for robust face recognition.

Materials and Methods

A. Apache Hadoop

Apache Hadoop ("Welcome to Apache Hadoop!" 2014) is a framework available as open source to store and process big data in a clustered distributed environment utilizing programming models. Hadoop is capable of processing large data sets in a distributed format using a large set of commodity hardware which is clustered using map reduce paradigms. It supports scalability to improve from a single machine to thousands with local computation and storage facility. It can detect and identify failures by itself. Even in the case of failure, the cluster setup is capable of providing highly available service. The Hadoop framework consists of several ecosystem tools capable of supporting different functionalities.

B. MapReduce Framework

A programming model used for distributed processing, supports scalability and availability computation based on Java. MapReduce consists of three steps, namely map, shuffle and reduce. The Mapper can translate sets of data into key/value pairs for processing. Then, the reducer will shuffle, merge and aggregates the data tuples into required output. The fortunate about using MapReduce is to support large-scale process several times over the cluster. There are two data processing functions, namely mappers and reducers. Scaling an application to any number of clusters results in a small configuration change. This flexibility attracts many programmers towards MapReduce model.

C. Apache Pig

Pig ("Welcome to Apache Pig!", 2013) is a Hadoop ecosystem tool developed by the Apache Software Foundation. It is a scripting language used for processing the large data sets that reside in Hadoop. The scripting language (also known as Pig Latin) compiles and converts the scripts into MapReduce Operations. It provides optimization by executing a minimum of map reduce jobs to run. Loaders and extensible User Defined Functions (UDFs) is used for achieving customization in data processing and format.

D. Hipi

Hadoop Image Processing Interface (HIPI) is a library which supports image processing designed to deploy and process images on Hadoop and MapReduce parallel framework. Hipi provides processing of images with high efficiency and throughput over the cluster nodes making use of MapReduce model. It is helpful to store the vast number of images in Hadoop Distributed File System (HDFS) and uses them for further processing efficiency in distributed networks. Hipi is integrated with some features of OpenCV ("Face Recognition with OpenCV", 2016) which is a library available open source with extensive computer vision algorithms. The essential and primary input to initiate processing in HIPI is Hipi image bundle (Hib). A Hib is an image collection file in HDFS. Hipi framework has several useful tools for creating hibs, to run a MapReduce program for building hibs over images downloaded from the internet.

E. Face Recognition Approaches

(1) Eigenface algorithm

Eigenface algorithm works by Principal Component Analysis (PCA). This method is used to reduce the issue of representing images of high dimensionality. It converts the possibly correlated variable set (N) into a smaller set of uncorrelated variables. Those images extracted are analyzed using PCA to filter out the required features out of all available features. Dimensionality reduction using PCA is accomplished with the features extracted from the image dataset, followed by mean and covariance calculation resulting in identifying eigenvectors. For the image datasets X = {x_1(a,b), x_2(a,b), x_3(a,b), ...} and Y = {y_1(a,b), y_2(a,b), y_3(a,b), ...}.Calculate the mean of query image x_i (a,b),

$$\text{Mean, } \mu_x = \frac{1}{N}\sum_{i=1}^{N} X_i(\mathrm{a,b}) \tag{1}$$

Calculate the mean of reference image y_i (a,b),

$$\text{Mean, } \mu_y = \frac{1}{N}\sum_{i=1}^{N} Y_i(\mathrm{a,b}) \tag{2}$$

Compute the covariance of query and reference images using (1) and (2),

$$\mathrm{Covar}(\mathrm{x_i}, \mathrm{y_i}) = \frac{\sum_{i=1}^{N} (X_i(a, b) - \mu_x)(Y_i(a, \mathrm{b}) - \mu_y)}{(N - 1)} \qquad (3)$$

Calculate the Eigen values (λ_i) and Eigen vectors (A_i) of Covar ($\mathrm{x_i}, \mathrm{y_i}$),

$$\mathrm{Covar}\ (\mathrm{x_i}, \mathrm{y_i}) * A_i = \lambda_i * A_i \qquad (4)$$

Use the Eigen values to order the Eigen vectors in descending order. Acquire the k principal components where k is the Eigen vector of largest Eigen value. The eigenvectors are arranged in descending order by their eigenvalue. The eigenvectors of large 'k' eigenvalues are the 'k' principal components. Then face recognition is performed by projecting all training subject and testing reference image into PCA subspace. The nearest neighbour is found out between the training and testing images.

(2) Fisherface algorithm

PCA method provides maximum variance of data using a linear combination of features, but the identified components will not have any discriminative information such as light. The Linear Discriminant Analysis (LDA) makes a class specific dimensionality reduction. It increases the ratio of between-classes to within-classes scatter, instead of maximizing the total scatter. The same classes will cluster tightly together while different classes are as far away as possible from each other in the lower-dimensional representation. This method learns a class-specific transformation matrix so that they do not capture illumination apparently as the Eigenfaces approach. The Discriminant Analysis rather finds the facial features to distinguish between the persons. The performance of the Fisherfaces heavily depends on the input data as well. Similar to eigenface reconstruction of the original image can be done but not well. Let Y be a random vector with samples taken from p classes.

$$Y = \{Y_1, Y_2, Y_{3\ldots} Y_p\}$$

$$Y_i = \{y_1, y_2, \ldots y_n\}$$

Compute the mean of class ($\mu_{i)}$ where class i ∈ {1…... p} and Maximize the class seperability by projection W where W = {A_1, A_1, $A_{1\ldots\ldots}$ A_k}.

$$W_{optim\,=}\ \text{argmax}_w\,|W^T S\,W| / |W^T S_w W| \quad (5)$$

Where S and S_w are the scatter matrices calculated using the total mean μ and class specific mean μ_i. W_{optim} gives the optimal value of similiarity between the classes. The number of samples N is nearly lesser than the input dimension (the number of pixels), hence the scatter matrix S_w turn out to be singular. Hence PCA was performed on the data and the samples are projected to N- p dimensional space. Now S_w is not singular as LDA was performed later on the reduced data. The transformation matrix W, that projects a sample into the (p-1)-dimensional space is specified as,

$$\mathrm{W} = \mathrm{W}_{lda}^{T}.\mathrm{W}_{pca}^{T} \quad (6)$$

(3) Local Binary Patterns Histogram (LBPH) algorithm

The Eigenfaces method maximizes the whole scatter, which can drive to difficulties if an external source creates the variance since components with a maximum difference over all classes aren't necessarily useful for classification. The Fisherfaces method uses LDA and performs well to preserve some discriminative information. LBPH describes only low-dimensional, local features of an object instead of a high dimensional vector. The Local Binary Patterns Histogram has its origins in 2D texture analysis. The fundamental notion of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighbourhood. Take a pixel as centre and threshold against its neighbors. If the intensity of the centre pixel is greater than equal its neighbour, then denote it with 1 and 0 if not. Finally, a binary number for each pixel, just like 11001111 is obtained. So, with eight surrounding pixels, 2^8 possible combinations, called Local Binary Patterns or sometimes referred to as LBP codes are obtained.

The operator for Local binary pattern is specified as,

$$\text{Lbp}\,(p_{c,}\,q_c) = \sum_{a=0}^{A-1} 2^{\mathrm{a}}.\,\mathrm{s}\,(\mathrm{i_a} - \mathrm{i_c}) \quad (7)$$

Where $(p_{c,}\, q_c)$ is the centre pixel of intensity i_c and intensity of neighbouring pixel i_a .The sign function ‘s’ is given as,

$$s(p) = \begin{cases} 1, & \text{if } p > 0 \\ 0, & \text{else} \end{cases}$$

Let the centre Point (p_c, q_c) the position of the neighbor (p_a, q_a), a ∈ A can be computed as,

$$p_a = p_c + r\cos\left(\frac{2\pi a}{A}\right)$$

(8)

$$q_a = q_c - r\sin\left(\frac{2\pi a}{A}\right)$$

(9)

where ‘r’ is the circle radius and ‘A’ is the number of sample points.

E. Supervised Machine Learning

Machine learning is the formation and study of computer algorithms that learn and improve automatically from data. Supervised Learning is a task of deducing a function from labelled training data, whereas the training data includes many examples of the training set. The model contains the input along with the derived output value. Examples of supervised learning algorithms include Bayesian networks and decision trees.

Dataset

The dataset consists of different images of every subject with varying facial expressions and postures. Some pictures of the subject are taken at altered timings. The images for training the machine learning model are extracted from the Google images and online social networks like Flickr, Twitter, and Facebook. These pictures are unstructured with different formats. The images are sized and scaled equally to 125 x 150. Four subjects each with 20 different postures of varying light conditions, facial expressions, and environment are used. Sample training images obtained from social networks are shown in Fig.1. The dataset is also obtained from Faces 96 (20 images per subject) and Yale face database with persons containing different postures under same light and environment with various movements and orientations. It includes 165 grayscale images in GIF format of 15 individuals. There are 11 images per subject, each with distinct facial expression or configuration.

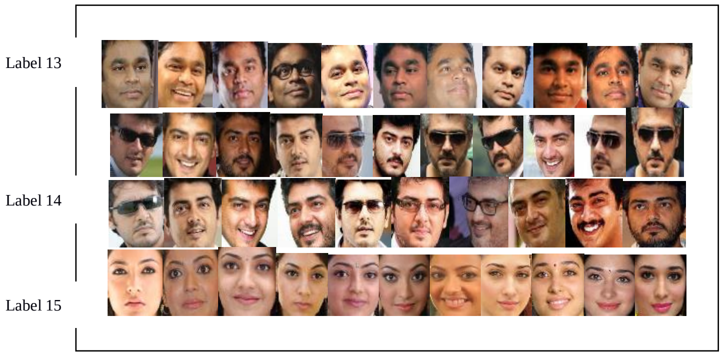


Fig.1. Sample training images from online social networks

Fig.2. Query image 14 (Test data)

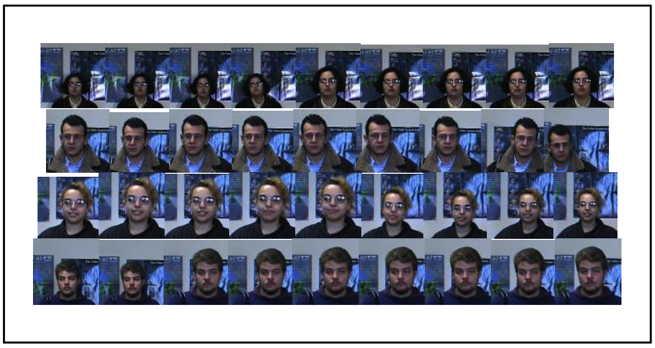


Fig.3. Sample training images from Faces96 dataset

System Framework

The basic block diagram of Image matching process flow is shown in fig.4.The system consists of a Hadoop YARN (Yet another Resource Navigator) framework for distributed processing and scalability. The HIPI framework is used for processing images in large scale and processes it with other Hadoop ecosystem tools. Initially, all the images from different social networks are stored in HDFS as Hadoop image bundles (Hib) using the HIPI framework. Those bundles are used further for processing using Hadoop and its tools.

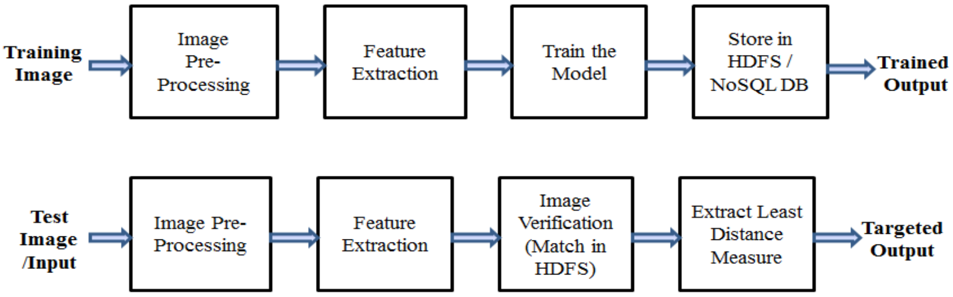


Fig.4. General process flow of image matching

Images are also extracted from standard image datasets Faces96, Yale face databases and other online social networks such as Google Images, Flickr, Twitter and Facebook using API's. All the images are scaled equally for the training purpose. The size of the images will be of an immense size and hence the image matching computation is implemented using Map Reduce parallel programming framework. All the images are compressed, and the compressed file is converted into a sequence file to be processed by Hadoop. Once it is converted, the images as sequence files are stored in HDFS. Now, the machine learning model is to be built using supervised learning approach. The images are labeled for each person with their different representations.

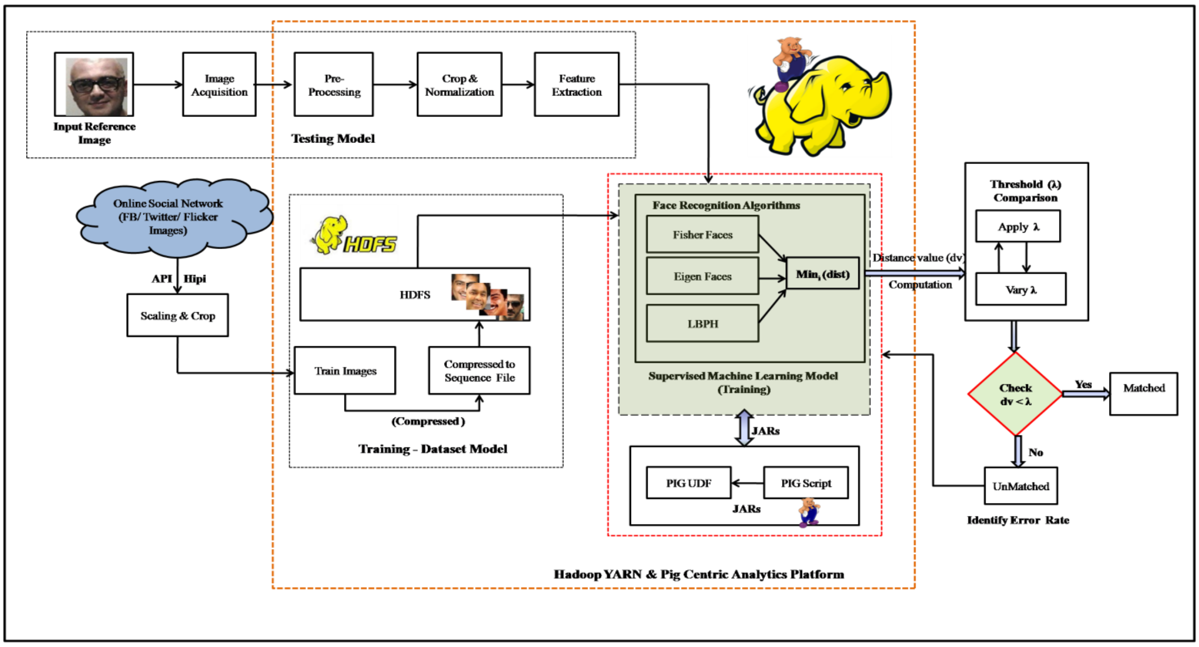
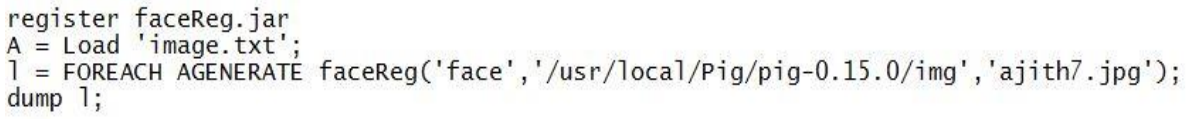


Fig.5. System framework of scalable image matching on Hadoop and pig-centric analytics platform

The machine learning model is built using face recognition algorithms namely Eigenface, Fisher face and Local Binary Patterns Histograms (LBPH). The Pig-centric analytics platform is used for operating on image data inside Hadoop. The algorithms are compiled with Pig User Defined Functions (UDFs) and registered. The pig scripting is executed which is shown below in Fig. 6, initiates the minimal number of MapReduce jobs needed for the process to be completed in optimized time and scalable way.



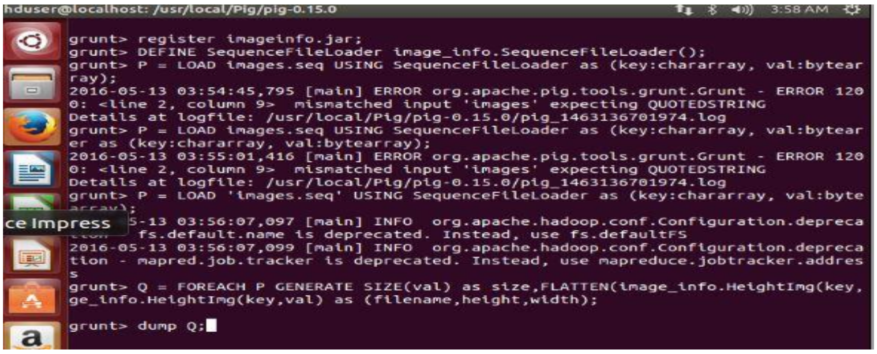
```
register faceReg.jar
A = Load 'image.txt';
l = FOREACH AGENERATE faceReg('face','/usr/local/Pig/pig-0.15.0/img','ajith7.jpg');
dump l;
```

Fig.6.Pig script execution of face recognition

Once it is calculated over the face recognition algorithms, the distances between the vectors are computed, and the matching images with the distance less than the threshold (λ) are fetched. The false acceptance rate and false rejection rate are calculated, and the training model is further fed with error rate to train it better. This results in identifying the existence of similarity. These results will help further to identify the anomaly user.

Experiments and Results

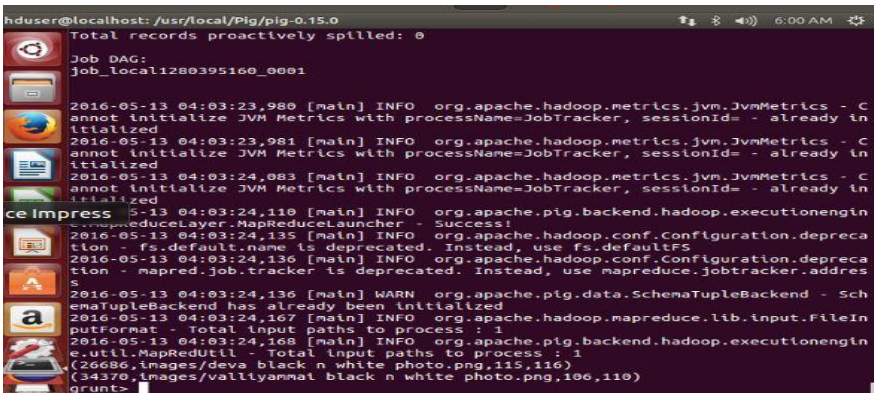
Experiments are done by using a single node Hadoop cluster. The large scale images are easily handled using Hadoop framework. Once the images are obtained, pre-processing is done to train the model. As shown in Fig.7, the height and width of images are calculated and scaled equally using Pig on Hadoop, which reduces the processing time for a huge volume of images. As shown in Fig.8, the resulting information about the image provides the image filename, type, height, and width. All the images are then compressed and converted to sequence file format and stored using sequence file loader in HDFS. NOSQL databases can also be used for storing the sequence images.



```
hduser@localhost: /usr/local/Pig/pig-0.15.0                                3:58 AM
grunt> register imageinfo.jar;
grunt> DEFINE SequenceFileLoader image_info.SequenceFileLoader();
grunt> P = LOAD images.seq USING SequenceFileLoader as (key:chararray, val:bytear
ray);
2016-05-13 03:54:45,795 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 120
0: <line 2, column 9>  mismatched input 'images' expecting QUOTEDSTRING
Details at logfile: /usr/local/Pig/pig-0.15.0/pig_1463136701974.log
grunt> P = LOAD images.seq USING SequenceFileLoader as (key:chararray, val:bytear
er as (key:chararray, val:bytearray);
2016-05-13 03:55:01,416 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 120
0: <line 2, column 9>  mismatched input 'images' expecting QUOTEDSTRING
Details at logfile: /usr/local/Pig/pig-0.15.0/pig_1463136701974.log
grunt> P = LOAD 'images.seq' USING SequenceFileLoader as (key:chararray, val:byte
array);
ce Impress 5-13 03:56:07,097 [main] INFO  org.apache.hadoop.conf.Configuration.depreca
tion - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-05-13 03:56:07,099 [main] INFO  org.apache.hadoop.conf.Configuration.depreca
tion - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addres
s
grunt> Q = FOREACH P GENERATE SIZE(val) as size,FLATTEN(image_info.HeightImg(key,
ge_info.HeightImg(key,val) as (filename,height,width);

grunt> dump Q;
```

Fig.7. Pig script executing the image information



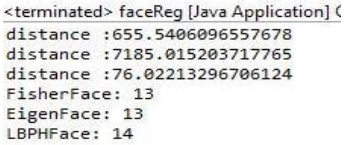
```
hduser@localhost: /usr/local/Pig/pig-0.15.0                                6:00 AM
Total records proactively spilled: 0

Job DAG:
job_local1280395160_0001


2016-05-13 04:03:23,980 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics - C
annot initialize JVM Metrics with processName=JobTracker, sessionId= - already in
itialized
2016-05-13 04:03:23,981 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics - C
annot initialize JVM Metrics with processName=JobTracker, sessionId= - already in
itialized
2016-05-13 04:03:24,083 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics - C
annot initialize JVM Metrics with processName=JobTracker, sessionId= - already in
itialized
ce Impress 5-13 04:03:24,110 [main] INFO  org.apache.pig.backend.hadoop.executionengin
e.mapReduceLayer.MapReduceLauncher - Success!
2016-05-13 04:03:24,135 [main] INFO  org.apache.hadoop.conf.Configuration.depreca
tion - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-05-13 04:03:24,136 [main] INFO  org.apache.hadoop.conf.Configuration.depreca
tion - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addres
s
2016-05-13 04:03:24,136 [main] WARN  org.apache.pig.data.SchemaTupleBackend - Sch
emaTupleBackend has already been initialized
2016-05-13 04:03:24,167 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileIn
putFormat - Total input paths to process : 1
2016-05-13 04:03:24,168 [main] INFO  org.apache.pig.backend.hadoop.executionengin
e.util.MapRedUtil - Total input paths to process : 1
(26686,images/deva black n white photo.png,115,116)
(34370,images/valliyammai black n white photo.png,106,110)
grunt>
```

Fig.8. Image information obtained from the Pig script

The machine learning model is built using the training images and the three face recognition algorithms.The given query image labeled 14 is compared with image sequence in HDFS. The similarity distance is measured between the eigen vectors computed using respective recognition methods. The threshold point is set as λ =100. As shown in Fig.9, the Fisher face and eigenface has their distance values greater than λ. But LBPH distance measure is less than λ and also it correctly matches with subject series labeled 14.



```
<terminated> faceReg [Java Application] (
distance :655.5406096557678
distance :7185.015203717765
distance :76.02213296706124
FisherFace: 13
EigenFace: 13
LBPHFace: 14
```

Fig.9. Distance computation results of face recognition algorithms

The size of the training images is then varied to numbers 1500 and 1920 of standard image datasets and similarity measure obtained are compared in Fig 10. The figure clearly shows that performance of LBPH over Hadoop produces possible accurate image matching results when compared to other two methods. The obtained accuracy information is also shown in Table I.

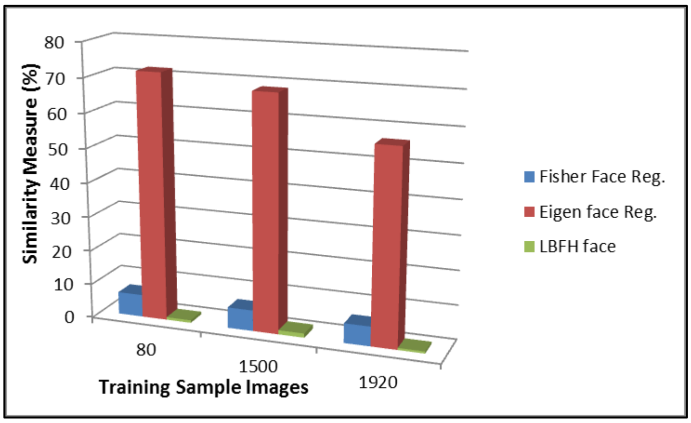


Fig.10. Comparison of image matching results of face recognition algorithms

Table I. Image matching Accuracy of the model

S.No	Face Recognition and Matching	Accuracy
1	Fisherface	55%
2	Eigenface	48%
3	LBPH	99%

.

Conclusion and Future Work

The images are compared to identify the similarity using the large scale parallel processing available in Hadoop framework and Pig. The proposed system is capable of processing a large number of images in an efficient way. The obtained results of similarity can be used for identifying the feature of anomaly users in social networks who use other users pictures to avoid revealing their original identity. Processing capability and cost of processing is made efficient by making use of available tools in Hadoop environment. The same process will be utilized as a part of future work for matching the signatures received from the user and the signature in the database to verify a user in case of any suspicion on multi-node Hadoop cluster.

Acknowledgement

The authors gratefully acknowledge DST, New Delhi for providing financial support to carry out this research work under PURSE scheme. One of the authors, Ms.Devakunchari Ramalingam, is thankful to DST, New Delhi for the award of DST PURSE fellowship.

References

Ding, C., Xu, C., & Tao, D. (2015). Multi-task pose-invariant face recognition. *IEEE Transactions on Image Processing*, 24(3), 980-993. doi: 10.1109/TIP.2015.2390959.

Face Recognition with OpenCV. Retrieved February 20, 2016, from http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#fisherfaces

Fang, Q., Sang, J., Xu, C., & Hossain, M. S. (2015). Relational user attribute inference in social media. *IEEE Transactions on Multimedia*, *17*(7), 1031-1044.

Gao, S., Zhang, Y., Jia, K., Lu, J., & Zhang, Y. (2015). Single sample face recognition via learning deep supervised autoencoders. *IEEE Transactions on Information Forensics and Security*, *10*(10), 2108-2118. doi: 10.1109/TIFS.2015.2446438

Guo, L., Zhang, C., & Fang, Y. (2015). A trust-based privacy-preserving friend recommendation scheme for online social networks. *IEEE Transactions on Dependable and Secure Computing*, 12(4), 413-427.

Huang, Z., Shan, S., Wang, R., Zhang, H., Lao, S., Kuerban, A., & Chen, X. (2015). A benchmark and comparative study of video-based face recognition on COX face database. *IEEE Transactions on Image Processing*, *24*(12), 5967-5981..doi: 10.1109/TIP.2015.2493448

Liang, X., Li, X., Zhang, K., Lu, R., Lin, X., & Shen, X. S. (2013). Fully anonymous profile matching in mobile social networks. *IEEE Journal on Selected Areas in Communications*, *31*(9), 641-655.

Najaflou, Y., Jedari, B., Xia, F., Yang, L. T., & Obaidat, M. S. (2015). Safety challenges and solutions in mobile social networks. *IEEE Systems Journal*, *9*(3), 834-854.

Nguyen, H. T., & Caplier, A. (2015). Local patterns of gradients for face recognition. *IEEE Transactions on Information Forensics and Security*, *10*(8), 1739-1751.doi:

10.1109/TIFS.2015.2426144.

Squicciarini, A. C., Lin, D., Sundareswaran, S., & Wede, J. (2015). Privacy policy inference of user-uploaded images on content sharing sites. *IEEE transactions on knowledge and data engineering*, *27*(1), 193-206.

Welcome to Apache Hadoop! Retrieved February 20, 2014, from http://hadoop.apache.org/

Welcome to Apache Pig! Retrieved January 2, 2013, from https://pig.apache.org/

Xia, F., Liu, L., Li, J., Ma, J., & Vasilakos, A. V. (2015). Socially aware networking: A survey. *IEEE Systems Journal*, *9*(3), 904-921.

Yale Face Database, Retrieved Jan. 18, 2016, from http://vision.ucsd.edu/content/yale-face-database.

Zhao, W., Chellappa, R., Phillips, P., and Rosenfeld, (2003). A. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)* 35(4), 399–458.

AUTHOR BIOGRAPHY

C. Valliyammai *M. Tech., Ph.D.,* is the Senior Grade Assistant Professor at the Department of Computer Technology, Anna University (Madras Institute of Technology campus), Chennai, India. She received her Ph.D. in computer science and engineering at Anna University. She has 15 years of teaching experience. Her areas of interest include Cloud computing, Big Data, Network management, Grid computing and Mobile agents. She has published around 25 papers in National and International conferences and journals. Email: cva@annauniv.edu.

R. Devakunchari *M. Tech.* is a Research scholar at the Department of Computer Technology, Anna University (Madras Institute of Technology campus), India. Her research interests include large-scale social network analysis and big data analytics. She has published papers in journals and conferences. She is currently pursuing her Ph.D. in Social Network Security, Anna University, Chennai, India. Email: devakunchari.r@gmail.com.