Clustering of Paired Image Datasets for High Dimensional Data Using Deep Learning

Mauzzam Siddiqui[1], Dr. Preetam Suman[2]
[1]M Tech Scholar,Department of Computer Science and Engineering,Integral University, Lucknow, Uttar Pradesh, India.
[2] Assistant Professor,Department of Computer Science and Engineering,Integral University, Lucknow, Uttar Pradesh, India.

ABSTRACT
The foremost computationally costly portion of numerous computer vision calculations comprises of looking for the most comparative matches to high dimensional vectors, also referred to as closest neighbour coordinating. Having an efficient algorithm for performing expeditious most proximate neighbour matching in immensely colossal data sets can bring speed amendments of several orders of magnitude to many applications. In this paper we exploit two algorithms for matching high dimensional features, first the K-nearest neighbour (KNN) and second SVM support learning technique using KNN. We also illustrate that the finest nearest neighbour algorithm and its parameters rely on the data-set features and portray an automatic configuration method for discovery of preeminent algorithm to search a particular data set.

KEYWORDS: *High dimensional data, KNN, Support Vector Machine, Face Recognition*

Date of Submission: 07-05-2019 Date of acceptance:24-05-2019

I. INTRODUCTION

The foremost computationally costly portion of numerous computer vision calculations comprises of looking for the most comparative matches to high dimensional vectors, also referred to as closest neighbour coordinating. Having an efficient algorithm for performing expeditious most proximate neighbour matching in immensely colossal data sets can bring speed amendments of several orders of magnitude to many applications. Examples of such issues incorporate finding the best matches for neighbourhood picture highlights in huge information sets, clustering nearby highlights into visual words utilizing the k-means or comparative calculation, worldwide picture featuring matching for scene acknowledgement, human posture estimation, coordinating deformable shapes for question acknowledgement or performing normalized cross-correlation (NCC)[1] to compare image patches in huge information sets.

It has been appeared that utilizing expansive preparing sets is key to getting great real life execution from numerous computers vision strategies. Nowadays the web may be a tremendous asset for such preparing information, but for huge information sets the execution of the calculations utilized rapidly gets to a key issue. When working with tall dimensional highlights, as with most of those experienced in computer vision applications (image patches, neighbourhood descriptors, worldwide picture descriptors), there's regularly no known nearest-neighbour look calculation that's correct and has satisfactory execution. To get a speed change, numerous commonsense applications are constrained to settle for an inexact look, in which not all the neighbours returned are correct, meaning a few are surmised but ordinarily still near to the precise neighbours. In practice it is common for inexact closest neighbour look calculations to supply more than 95 percent of the proper neighbours and still be two or more orders.

To get a speed change, numerous practical applications are constrained to settle for an approximate search, in which not all the neighbours returned are exact, meaning a few are surmised but typically still near to the precise neighbours.

In numerous cases the closest neighbour look is fair a part of a bigger application containing other approximations and there's exceptionally small misfortune in performance from utilizing inexact or maybe than exact neighbours.

1.1 K-Nearest Neighbor

The KNN algorithm is very straightforward and very effective. The model depiction for K-Nearest Neighbor (KNN) is the complete training dataset.

Forecasting is prepared for a new-fangled data point by hunting through the whole training dataset for the K most analogous instances (the neighbors) and then output variables are summarized for those K instances. This might be the mean output variable for regression tribulations, while for classification tribulations this might be the mode (or most common) class value[2].
The trick is in how to determine the similarity between the data instances. The straightforward method if your attributes are all of the same scale (all in inches for example) is to utilize the Euclidean distance, a number you can evaluate unswervingly based on the differences between each input variable.

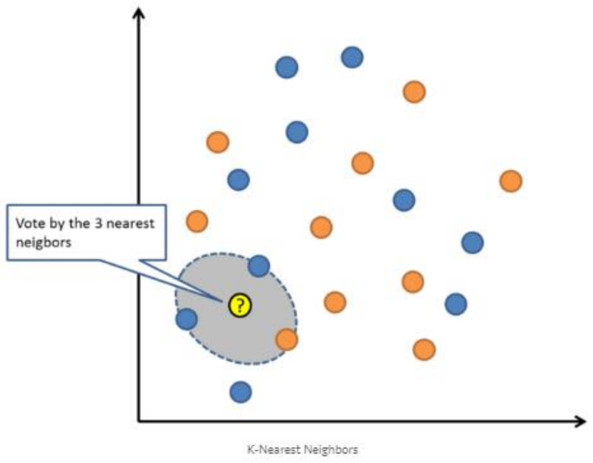


Figure 1: K- Nearest Neighbour

1.2 Support Vector Machine (SVM)

Support Vector Machines (SVM) are based on the conception of decision planes that delineate decision boundaries. A decision plane is one that divides between a set of objects having dissimilar class memberships. A graphic paradigm is illustrated below. In this illustration, the objects belong either to class GREEN or RED. The extrication line delineate a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any novel object (white circle) falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the extrication line).

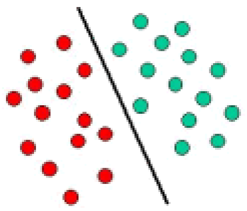


Figure 2: Linear Classifier

The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., appropriately classify novel objects (test cases) on the basis of the examples that are available (train cases). This circumstance is portrayed in the illustration below. Compared to the preceding diagram, it is clear that a full severance of the GREEN and RED objects would necessitate a curve (which is more complex than a line). Classification tasks based on drawing extrication lines to discriminate between objects of diverse class memberships are recognized as hyperplane classifiers. SVMs are chiefly suitable to handle such tasks[3].

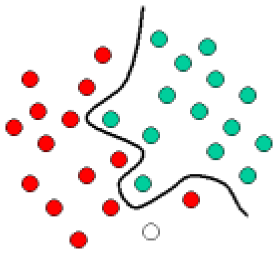


Figure 3: Complex Classifier (Curve)

The illustration below demonstrates the fundamental inspiration SVMs. Here we observe the imaginative objects (left side of the schematic) mapped, i.e., rearranged, utilizing a set of mathematical functions, recognized as kernels. The process of rearranging the objects is recognized as mapping (transformation). Note that in this novel setting, the mapped objects (right side of the schematic) is linearly distinguishable and, thus, instead of creating the multifaceted curve (left schematic), all we have to do is to find an most favorable line that can split the GREEN and the RED objects.

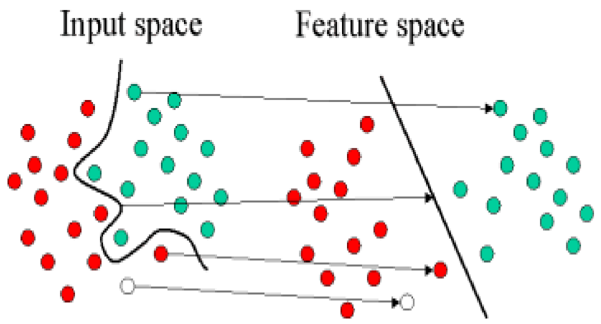


Figure 4: Support Vector Machine

II. LITERATURE REVIEW

For many computer vision and machine learning problems, large training sets are key for good performance. Nonetheless, the most computationally high-priced part of many computer vision and machine learning algorithms consists of finding adjacent neighbour matches to lofty dimensional vectors that symbolize the training data. Marius Muja et. al. [4] proposed new algorithms for approximate nearest neighbour matching. For matching lofty dimensional characteristics, they discovered randomized k-d forest as well as priority search k-means tree algorithms to be the largely competent. They also proposed a novel algorithm for matching binary characteristics by searching multiple hierarchical clustering trees and revealed that it do better than methods typically used in the literature. With the intention of scaling to very bulky datasets that would otherwise not fit in the memory of a single machine, they proposed a distributed nearest neighbor matching framework that can be utilized with any of the algorithms.

Marko Arsenovic et. al.[5] exploited deep convolution neural networks (CNNs) to proposed a novel deep learning based face identification attendance system. Their model comprises numerous indispensable steps developed using today's most advanced techniques: CNN cascade for face detection and CNN for generating face embeddings. The chief objective of this research work was the practical employment of these state-of-the-art deep learning approaches for face identification tasks. The proposed face identification model could be incorporated in another system with or without some slight alternations as a supporting or a main component for monitoring purposes.

Edgar Osuna et. al.[6] investigated the application of SVMs in computer vision. They offered a decomposition algorithm that promised global optimality, and can be utilized to prepare SVM's over extremely bulky data sets. The chief thought behind the decomposition is the iterative solution of sub-problems and the evaluation of optimality conditions which are used both to produce improved iterative values, and also establish the stopping criterion for the algorithm. They presented experimental outcomes and shown the practicability of proposed approach on a face recognition predicament that involves a data set of 50,000 data points.

Guodong Guo et. al.[7] partitioned face images into different qualities, and evaluated the recognition performance, using the state-of-the-art deep networks. Some fascinating results are obtained, and their studies show directions to encourage the deep learning methods towards towering accurateness and realistic use in solving the rigid predicament of unconstrained face identification.

Okfalisa et. al.[8] analyzed the comparison of KNN and MKNN algorithms to classify the data of Conditional Cash Transfer Implementation Unit (Unit Pelaksana Program Keluarga Harapan) which consist of 7395 records. Comparative analysis is based on the accuracy of both algorithms. Before classification, K-Fold Cross Validation was done to search for the optimal data modeling resulted in data modeling on cross 2 with accuracy of 93.945%.

III. PROPOSED APPROACH

Step 1:

First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We also require setting an ID (it may be a number or the name of the person) for every picture, so the algorithm will exploit this information to distinguish an input picture and give you an output. Images of the same person must have the same ID.

Step 2:
The first computational step of the Algorithm is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**.

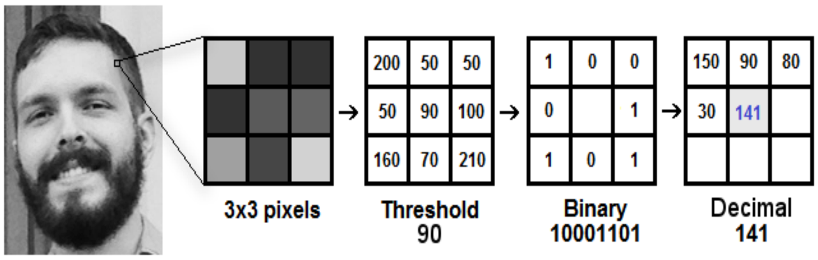


Figure 5: Procedure of conversion from pixel to decimal

Based on the image above, let's break it into several small steps so we can understand it easily:

- Suppose we have a facial image in gray-scale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We require concatenating each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101).
- Then, we translate this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (KNN procedure), we have a new image which represents better the characteristics of the original image.
- Note: The KNN procedure was expanded to use a different number of radius and neighbors, it is called Circular KNN.

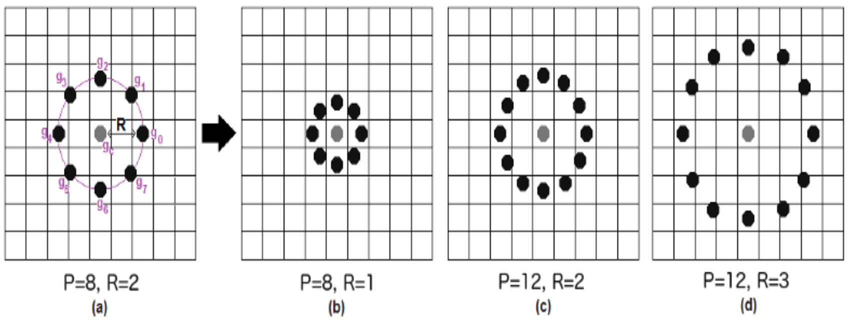


Figure 6: Circular KNN

It can be done by using **bilinear interpolation**. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the novel data point.

Step 3:
Histograms Extraction: The image generated in the last step, is divided into multiple grids, by exploiting the **Grid X** and **Grid Y** parameters. This can be seen in the figure 7.

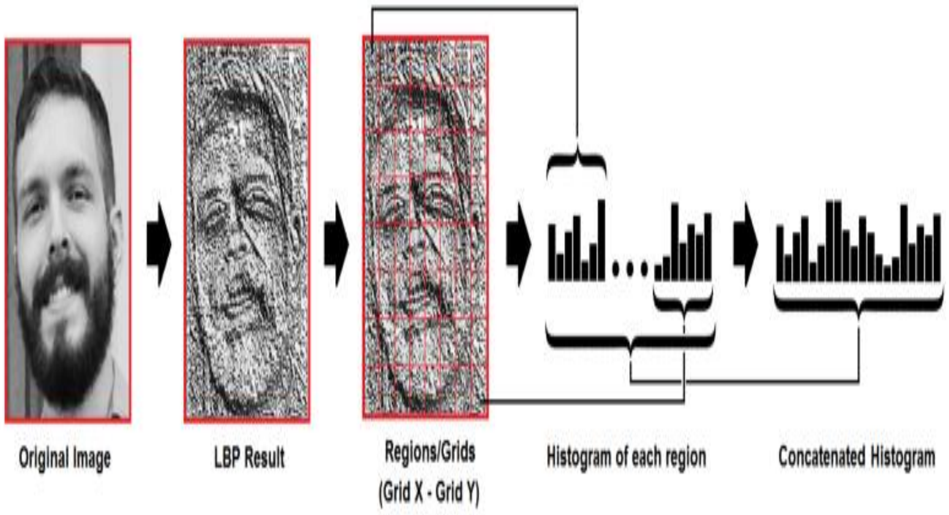


Figure 7: Histograms

Histogram of each region of image can be extracted as follows:

- As we have an picture in gray-scale, each histogram (from each grid) will restrain only 256 positions (0~255) representing the incidences of each pixel concentration.
- Then, we require concatenating each histogram to generate a novel and improved histogram. Presume we have 8x8 grids, we will have 8x8x256=16.384 positions in the finishing histogram. The finishing histogram represents the features of the original picture.

Step 4:

In this step, the algorithm is previously trained. Each histogram produced is exploited to symbolize each picture from the training dataset. So, given an input picture, we carry out the steps again for this novel image and generate a histogram which characterizes the picture.

- So to discover the picture that matches the input picture we just require comparing two histograms and returning the picture with the closest histogram.
- For comparing the histograms, can employ diverse approaches like **Euclidean distance**, **chi-square**, **absolute value**, etc. can be exploited. In this research, we have exploited the Euclidean distance as follows:

D= $\sqrt{\sum_{i=1}^{n}(\text{hist1}_i - \text{hist2}_i)^2}$

- So the algorithm output is the ID from the picture with the closest histogram. The algorithm should also return the computed distance, which can be used as a '**confidence**' measurement.
- We can then employ a threshold and the 'confidence' to automatically guesstimate if the algorithm has accurately recognized the picture. We can assume that the algorithm has lucratively recognized if the confidence is lesser than the threshold delineated.

The flow diagram and system architecture is shown in figure 8 and 9.

IV. EXPERIMENTAL EVALUATION

For implementing our proposed approach we used tensorflow (version: 1.12.0)by using opencv (version:4.0.0)and python (version 3.6.7). We also used dlib to make my work more efficient. We resized the datasets to 400x400 pixels. Pickle encoding of images are done to process the datasets. Snapshots of our experimentation is shown in figure 10 and 11.

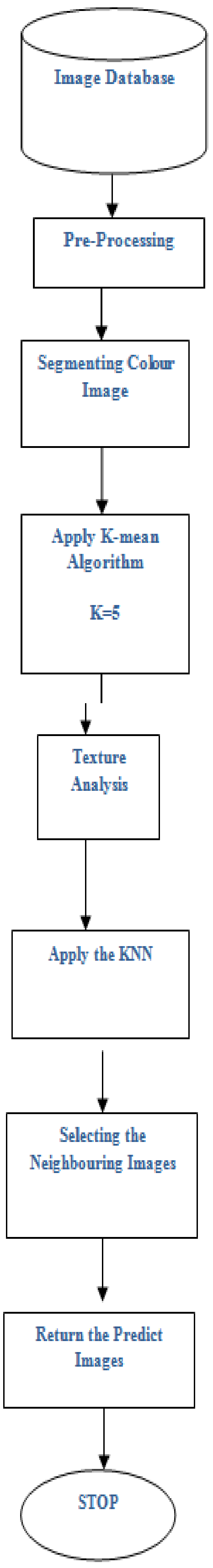


Figure 8: Flow diagram of proposed approach

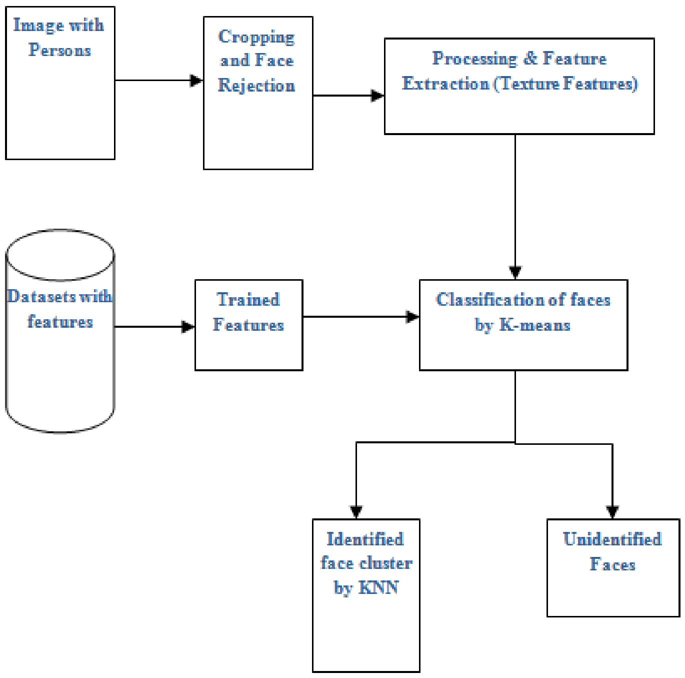


Image with Persons
Cropping and Face Rejection
Processing & Feature Extraction (Texture Features)
Datasets with features
Trained Features
Classification of faces by K-means
Identified face cluster by KNN
Unidentified Faces

Fig 9: System Architecture

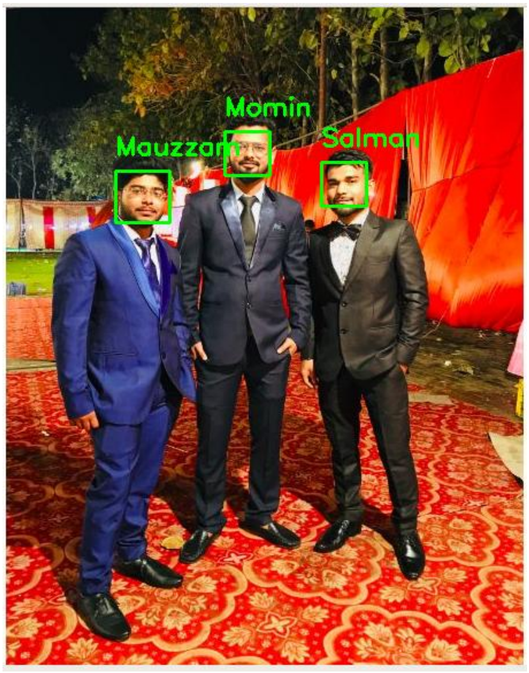
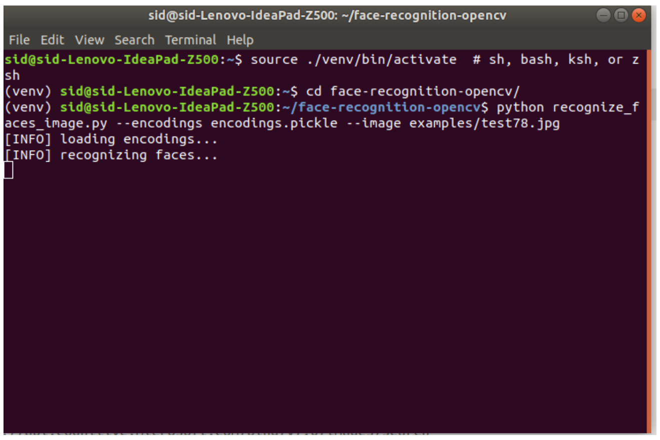


Figure 10: Screenshot of Face recognition in proposed approach



```
sid@sid-Lenovo-IdeaPad-Z500: ~/face-recognition-opencv
File  Edit  View  Search  Terminal  Help
sid@sid-Lenovo-IdeaPad-Z500:~$ source ./venv/bin/activate  # sh, bash, ksh, or z
sh
(venv) sid@sid-Lenovo-IdeaPad-Z500:~$ cd face-recognition-opencv/
(venv) sid@sid-Lenovo-IdeaPad-Z500:~/face-recognition-opencv$ python recognize_f
aces_image.py --encodings encodings.pickle --image examples/test78.jpg
[INFO] loading encodings...
[INFO] recognizing faces...
```

Figure 11: Screenshot of working implementation of proposed approach

V. RESULT ANALYSIS

The result of my proposed approach by using KNN algorithm to detect & recognize the face images with an accuracy of 89%.

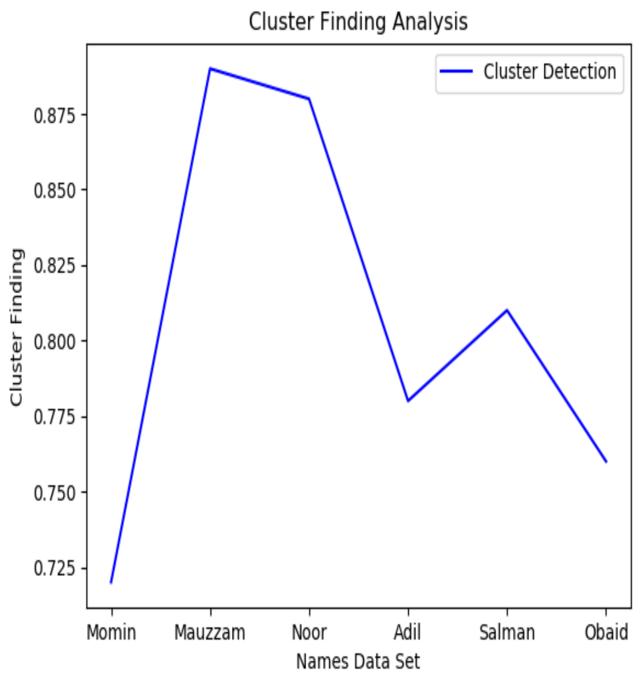


Figure 12: Cluster finding analysis

The result of my proposed approach by using KNN and SVM to detect and recognize face images is having accuracy of 93%

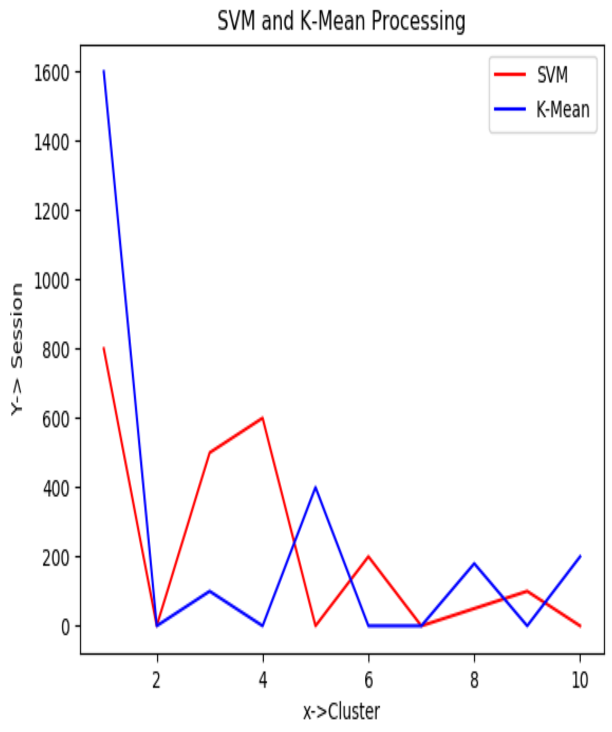


Figure 13: SVM and K-Means processing

VI. COMPARATIVE ANALYSIS

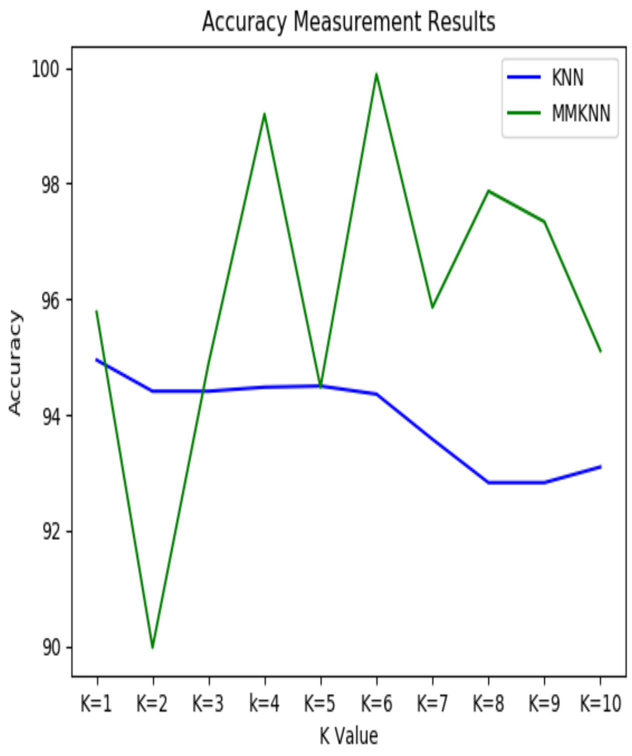


Figure 14: Accuracy Measurement Results

On comparison with the previous KNN with my modified KNN algorithm (MMKNN), the accuracy increases by 2%, which was 93.94%. The accuracy of MMKNN is 95.94%.

VII. CONCLUSION

Our main objective is to detect a candidate from a group also using KNN and SVM algorithms. When a candidate who is either a criminal, missing person or someone else whom we've to find from a group, then we're going to search the person with the help of our trained cameras and datasets. KNN helps in finding the nearest neighbor datasets and SVM helps by providing accuracy and efficiency to the trained datasets by its experience. By this, we can increase the accuracy and efficiency of face detection.

REFERENCES

[1]. Tong, Mingsi, et al. "Valid data based normalized cross-correlation (VDNCC) for topography identification." Neurocomputing, 308, pp. 184-193, 2018.
[2]. Karamizadeh, Sasan, and Abouzar Arabsorkhi. "Skin Classification for Adult Image Recognition Based on Combination of Gaussian and Weight-KNN." International Journal of Information and Communication Technology Research 10.2, pp 56-62, 2018.
[3]. Thanh Noi, Phan, and Martin Kappas. "Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using Sentinel-2 imagery" Sensors 18.1, No. 18, 2018.
[4]. Marius Muja, David G. Lowe, "Scalable Nearest Neighbour Algorithms for High Dimensional Data", IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. 36, NO. 11, 2014.
[5]. Marko Arsenovic, Srdjan Sladojevic, Andras Anderla, Darko Stefanovic, "FaceTime – Deep Learning Based Face Recognition Attendance System", IEEE 15th International Symposium on Intelligent Systems and Informatics, 2017.
[6]. Osuna, Edgar, Robert Freund, and Federico Girosit. "Training support vector machines: an application to face detection." In Computer vision and pattern recognition, 1997. Proceedings, 1997 IEEE computer society conference on, pp. 130-136. IEEE, 1997.
[7]. Guo, Guodong, and Na Zhang. "What Is the Challenge for Deep Learning in Unconstrained Face Recognition?." In 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pp. 436-442. IEEE, 2018.
[8]. Comparative Analysis of K-Nearest Neighbor and Modified K-Nearest Neighbor Algorithm for Data Classification, Okfalisa, Mustakim, Ikbal Gazalba, Nurul Gayatri Indah Reza, 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE).

Mauzzam Siddiqui" Clustering of Paired Image Datasets for High Dimensional Data Using Deep Learning" International Journal of Computational Engineering Research (IJCER), vol. 09, no. 5, 2019, pp 63-72