# Smart Image Segregation using Face Recognition

Raj Shaiwalla and Arindam Chaudhuri

April 18, 2021

# Smart Image Segregation using Face Recognition

Raj Shaiwalla
Mukeh Patel School of Technology Management and Engineering NMIMS University
Mumbai India
raj.shaiwalla1721@nmims.edu.in

Arindam Chaudhuri
Mukesh Patel School of Technology Management and Engineering NMIMS University
Mumbai India
arindam.chaudhuri@nmims.edu

**Abstract:** This paper discusses a method of classifying and segregating images using facial recognition. This study would define the most critical features for evaluating within a model that can distinguish one face from another. Extraction and collection of features are critical measures to better distinguish people from one another. Extraction of features is the method of extracting different properties from a set of results. Selection of features is the method that follows extraction, where the most important features are chosen to represent each sample. Once the appropriate features are selected, they are added as potential inputs to the neural network. The image dataset used for this project has been provided to me by a relative. This dataset contains suitable .jpg files containing over 4000 images we aim to classify and segregate. The model in this research differentiates between types of people based on their faces. Diverse features are tested to determine which elements performed better. The project mainly uses dlib and face recognition libraries in order to provide the functionality.

## 1 Introduction

Face analysis and identification are two such pathways that have a promising and beneficial impact on society. Artificial intelligence and machine learning advances have advanced these technologies significantly over the years. Computer vision is another area where technology is extremely useful. These innovations are in high demand in our daily lives these days.

While there is a wide range of research in related to facial recognition, there is relatively little work on applying these concepts to the real word for image segregation. The development of several different libraries and APIs for face recognition has resulted from such study and development of such problems. These libraries can be customised to meet our needs in order to solve our problem. Facial recognition is also one of the most common and most explored tasks in the field of computer vision. It can provide useful information for both image understanding and video content analysis.

Classification of people is a common problem with many practical applications in computer vision. Another application could be to automatically organize a huge image dataset and tag every person by their name. These images can be used for finding similar images of the person.

Convolution neural networks in the field of computer vision is considered to deliver great outcomes. We've seen a lot of improvement in this field for several years before we now achieve better than human accuracy for the classification of pictures. Such networks learn to recognize specific input features, and learn more complex features when stacked one after the other. Through the years some improvements have been implemented, such as dropout to prevent overfitting, normalization of batches to make weight initialization not a problem etc. For this model, libraries such as face recognition [2], [3] and dlib [1] were used in order to solve our problem. In this case, computer vision neural network ideas for image recognition were applied.
.
Face recognition has recently received a lot of attention as one of the most promising applications of image analysis and understanding, particularly in the last few years. This development can be attributed to at least two factors: the broad spectrum of commercial and law enforcement applications, and the availability of viable innovations after 30 years of study. Even though current machine recognition systems have achieved a certain level of maturity, the conditions imposed by many real-world applications hinder their performance. Image classification is a fundamental problem in the field of image processing. The key task is to extract characteristics from the image, and then classify which class the image belongs to. It is not just the classification that is a key problem but the automated segregation of those classes itself, which is the aim of the project.

This paper is organised as follows. In section 2 significant related works are summarised. This is followed by methodology in section 3. The results and discussions are presented in section 4. In section 5 conclusions are highlighted.

## 2 Related Work

After comparing various research papers related to image and face recognition, image and face classification available through resources various similarities has been observed when it comes to approach and motivation behind these various works. Keeping this in view we describe here few related significant research works. These works have described process used to do perform face recognition and classification and some related components [8].

Zeiler et al [7] proposed a model which consists of multiple interleaved layers of convolutions, non-linear activations, local response normalizations, and max pooling layers. We additionally add several $1 \times 1 \times d$ convolution layers inspired by work of Lin et al **[**11**]**.

Szegedy et al [9] recently implemented an inception model as the winning approach for ImageNet 2014. These networks use mixed layers that run several different convolutional and pooling layers in parallel and concatenate their responses. We have found that these models can reduce the number of parameters by up to 20 times and have the potential to reduce the number of FLOPS required for comparable performance. There are vast corpus of face verification and recognition works. We review here few significant ones only viz Sun et al [10], Taigman et al [12]**,** Zhu et al [13]. In these works, complex system of multiple stages has been employed which combines output of deep convolutional network with PCA for dimensionality reduction and an SVM for classification.

Zhu et al [13] employed a deep network to warp faces into a canonical frontal view and then learn CNN that classifies each face as belonging to a known identity. For face verification, PCA on the network output in conjunction with an ensemble of SVMs is used.

Taigman et al [12] proposed a multi-stage approach that aligns faces to a general 3D shape model. A multi-class network is trained to perform the face recognition task on over four thousand identities. The authors also experimented with a so-called Siamese network where they directly optimize the L1-distance between two face features. Their best performance on LFW (97.35%) stems from an ensemble of three networks using different alignments and color channels. The predicted distances (non-linear SVM predictions based on $\chi 2$ kernel) of those networks are combined using a non-linear SVM.

Sun et al [10] proposed a compact and therefore relatively cheap to compute network. They use an ensemble of 25 of these networks each operating on a different face patch. For their final performance on LFW (99.47 %) authors combine 50 responses (regular and flipped). Both PCA and a Joint Bayesian model that effectively correspond to a linear transform in the embedding space are employed. Their method does not require explicit 2D/3D alignment. The networks are trained by using a combination of classification and verification loss. The verification loss is similar to the triplet loss we employ in that it minimizes the L2-distance between faces of the same identity and enforces a margin between the distance of faces of different identities. The main difference is that only pairs of images are compared, whereas the triplet loss encourages a relative distance constraint.

Rybchak et al [15], [5] describe the actual methods and technologies for all stages of the development of the recognition system, since in the field of recognition, a huge number of unique solutions have been developed. The researchers introduce the method of face recognition using the SVM which significantly improve the speed and efficiency of the process of comparison of faces. Scholars have introduced the facial landmark estimation algorithms and techniques which are used to position the faces in the frame. It in turn helps the model in identifying the faces more efficiently and increase the overall quality of the system we are trying to achieve. Based on the study of documentation and analysis we have come to a conclusion that there no single pathway to conduct a facial recognition rather there are many methods and ways one could achieve this goal. But, pertaining to this article we find it easier to use face recognition library.

## 3 Methodology

The primary objective for this project is to use this wedding dataset to perform image segregation using facial recognition.

The dataset involves over 4000 .jpg images. The images are taken over the occurrence of three wedding functions, each having different locations, different number of photos taken per function, different amount of people attending the wedding, different illumination due to the different locations.

The project setup environment includes following significant artefacts. Processor: Intel® Core™ i7-8750H CPU @ 2.2GHz (12 CPUs); RAM: 16384MB RAM; Operating Systems: Windows 10 64-Bit (Build 18363); Graphic Card: NVIDIA GeForce GTX 1070 with Max-Q Design; Programming Language: Python (Version 3.7) with libraries os, cv2, pickle, face_recognition, imutils, numpy, pandas, shutil etc.

Now we present all the steps in building a face recognition and segregating system and implement with help of libraries as mentioned above.

The first step to make a smart image segregation system is the step of identifying the face itself in the image [6]. Whether any face is not identified, or if any other entity is viewed as a face, the device we are implementing will be ineffective, and the results will be unsatisfactory. To solve this problem, we employ one of the most widely used algorithms built by Dalal et al [14] for detecting faces in images called histogram of oriented gradients (HOG).

Fig. 1: Grayscale Image

Since we don't need color data to find faces in a picture, we'll start by converting it to black and white as seen in Fig. 1. Then we'll go over each and every pixel in our picture one by one. We want to look at the pixels that are immediately surrounding each pixel. Our aim is to determine how dark the current pixel is in comparison to the pixels in its immediate vicinity. Then we'll draw an arrow indicating which way the picture is darkening. Hence, we end up with arrows called gradients that show the flow of light from light to dark across the entire image.

We'll later divide the image into 16 × 16 pixel squares with each block being replaced by the strongest arrow path in that block. The final product would be a very basic representation of the face caught in the picture as shown in Fig. 2. The image is then compared to the HOG pattern, which was extracted from a collection of training faces and marked as the most common pattern with the identified pattern. As a result, we'll be able to identify the faces in image dataset we choose to segregate.

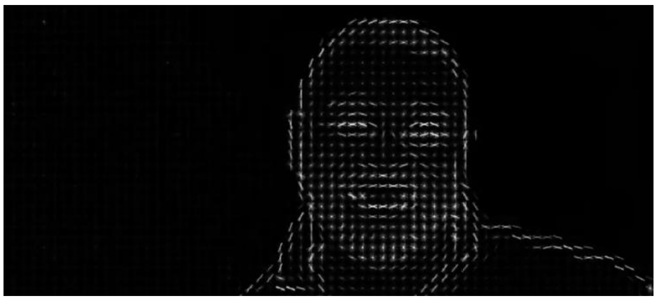


Fig. 2: HOG based image

After finding the face in the picture, the next issue we face is face positioning; in most images, the face is not centre placed as the algorithm requires; otherwise, the algorithm's efficiency and accuracy would suffer. We use face landmark estimation by Kazemi et al [16] in order to solve this problem. The basic concept is that we can identify 68 distinct points (known as landmarks) on any face, such as the top of the chin, the outside edge of each eye, the inner edge of each forehead, and so on as seen in Fig. 3. Then, on any face, we'll train a machine learning algorithm to find these 68 unique points.

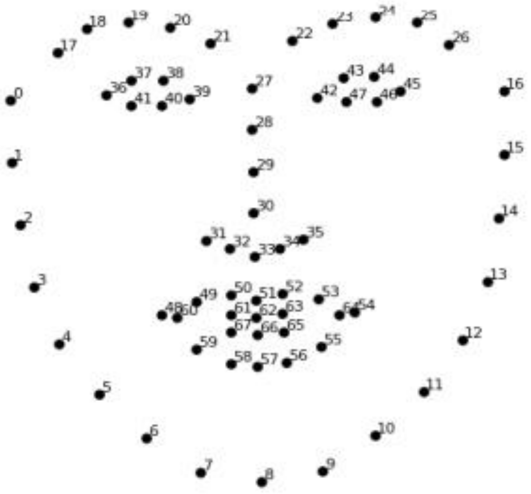


Fig. 3: The 68 landmarks we will locate on every face

Now that we know where the mouth and eyes are, we'll rotate, scale, and shear the picture to focus the eyes and mouth as much as possible. By completing this step i.e., centering face we will be able to improve the accuracy and efficiency of the next step.

The next step will be to learn how to distinguish between various types of faces. This can be accomplished by taking simple measurements from each forehead. Then we could use the same method to calculate our unknown face and find the known face with the nearest measurements. We might, for example, take measurements of the size of each ear or the distance between our eyes. So, we need to consider here correct metrics. Another issue we may encounter is that, with such a large database of faces, the comparison process will take a long time, making it inefficient. At first glance, we might believe that the main characteristics of the face, such as the distance between the eyes, the distance between the ears, or the length of the nose, are appropriate for the main measurements of the face, but as we dig deeper, we discover that the machine does not view the face as a whole, but rather considers the pixels of the picture. To solve this problem, it was suggested that we can use a deep convolutional network, which will be trained to identify 128 unique numerical facial features. But instead of training the network to recognize pictures objects like we did last time, we generate the 128 measurements called embeddings: The training process works by following steps:

(a) Load training face image of a known person
(b) Load another picture of the same person
(c) Load a picture of a different person

The deep convolutional network will change the results from images 1 and 2 so that the 128 characteristics obtained are as similar as possible, while the image loaded in the third stage is as different as possible.

The next step is to train the deep convolutional neural networks to produce unique numerical values of the 128 characteristics from a large number of faces in the databases, which requires a significant amount of computational power. When the neural network is equipped, it can take the input of a face that has never been seen before and instantly produce unique characteristics. This makes the move the most critical of the others we've discussed; insufficient preparation can lead to unsatisfactory results and inefficiency. If you don't want to train a model, you can always use one that has already been trained and incorporate it into your algorithm to generate the features. In this project, we use the same strategy by using a pre-trained model to drastically minimize our job and economic costs.

The algorithm's next step will be to compare the faces to available faces, which means that the available 128 features from the previous step will be compared to the data we have. The task at hand is to figure out how to compare the faces, and for this, we've chosen to use SVM. We train the classifier with the available features; the more homogeneous the features are, the better we can determine the face [4].

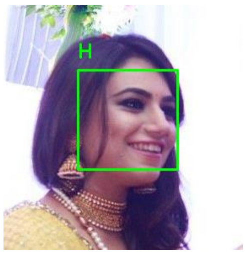
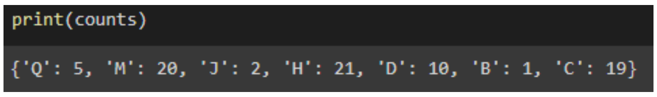


Fig. 4: Example of Face Recognition on singular face

We proceed to run the model and the face matches are calculated by the use of embeddings. Internally compare_ faces function is computing the euclidean distance between the candidate embedding and all faces in our dataset. If the distance is below some tolerance (the smaller the tolerance, the more strict our facial recognition system will be) then we return True, indicating the faces match**.** Otherwise, if the distance is above the tolerance threshold, we return false as the faces do not match.



```
print(counts)

{'Q': 5, 'M': 20, 'J': 2, 'H': 21, 'D': 10, 'B': 1, 'C': 19}
```

Fig. 5: Counts

Given our matches list we can compute the number of votes for each name (number of True values associated with each name), tally up the votes, and select the person's name with the most corresponding votes as shown in Fig. 5. This system is used to identify and individual person as shown in Fig. 4 and an image containing multiple people in it, and it is done for each person as shown in Fig. 6.

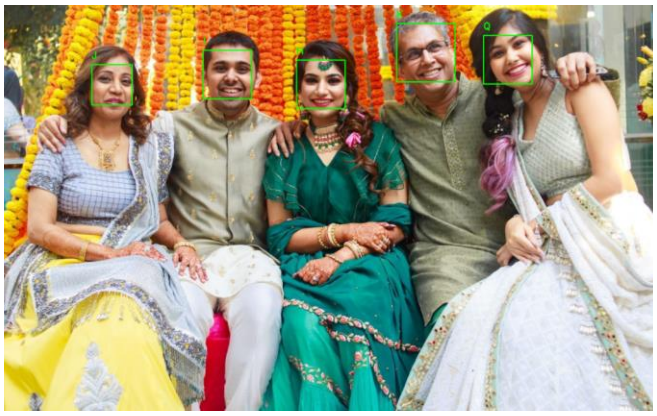


Fig. 6: Example of Face Recognition on multiple faces

If there are any True votes in matches, we need to determine the indexes of where these True values are in matches. We do that by creating a simple list. And we then create a dictionary that holds the name of the person as the key and the list of images the person is present in as the values of that key. Using this dictionary, we export all the images i.e., the values corresponding to the name, to a folder holding the person's name as the name of the folder, with all of the images of the person as the content [7].

## 4 Results and Discussions

As we can see from Fig. 7 algorithm has correctly classified the images based on the face and exported the respected images to a google drive folder.

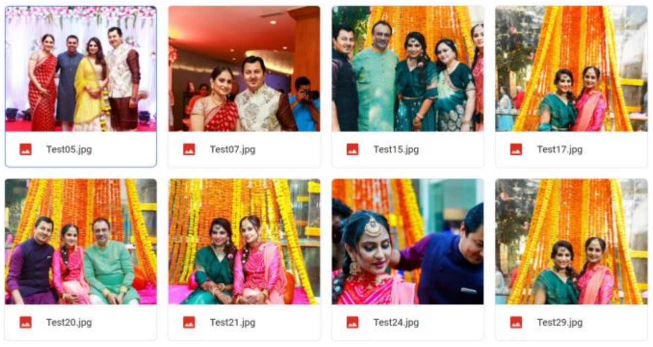


Fig. 7: Final Results

We can see that our model has a great accuracy in terms of detecting the image, classifying it and segregating it. In today's age this project right here is a stepping stone for image segregation, as it cannot be only pertained to a wedding but any social gathering. And it drastically reduces the manual effort of finding one's personal photos in a huge image collection.

## 5 Conclusion

In this report, all main aspects of face detection, classification and segregation have been covered. The growth of classification systems driven by AI suggests a future for a personalized user experience of fluid. It will be one of the most effective ways of delivering content that is context-conscious. This paper described the development of a simple image segregation system, and since the usage of pretrained models available, we were able to utilise them and implement these models in a real-world use case application. The future prospects of this project can be to classify people not only in images and segregate those images but to detect them in videos and export them along with the images.

## References

[1] http://dlib.net/python/index.html
[2] https://github.com/davisking/dlib/tree/master/python_example
[3] https://github.com/ageitgey/face_recognition
[4] https://scikit-learn.org/stable/modules/svm.html
[5] https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/
[6] https://www.superdatascience.com/blogs/opencv-face-recognition
[7] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013.
[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4), 541–551, 1989.
[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014.

[10] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. CoRR, abs/1412.1265, 2014.
[11] M. Lin, Q. Chen, and S. Yan. Network in network. CoRR, abs/1312.4400, 2013.
[12] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In IEEE Conf. on CVPR, 2014.
[13] Z. Zhu, P. Luo, X. Wang, and X. Tang. Recover canonical view faces in the wild with deep neural networks. CoRR, abs/1404.3543, 2014.
[14] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision and Pattern Recognition, CVPR 2005, 886–893.
[15] Z. Rybchak and O. Basystiuk, Analysis of computer vision and image analysis technics, ECONTECHMOD: an international quarterly journal on economics of technology and modelling processes, Lublin: Polish Academy of Sciences, 6(2), 79–84, 2017.
[16] V. Kazemi and J. Sullivan. One Millisecond Face Alignment with an Ensemble of Regression Trees. International Conference on Computer Vision and Pattern Recognition, CVPR 2014, 1867–1874.