EXAMINING THE STATE OF THE ART IN FACIAL RECOGNITION SYSTEMS

A PREPRINT

Debargha Ganguly
Department of Computer Science
Ashoka University
Haryana, India - 131029
debargha.ganguly_ug20@ashoka.edu.in

Prajwal Seth
Department of Political Science
Ashoka University
Haryana, India - 131029
prajwal.seth_ug20@ashoka.edu.in

December 1, 2019

ABSTRACT

In this paper, we discuss how the approaches taken for Facial Recognition have changed across the years, in addition to the current state of the art used in production. Building on Google's landmark FaceNet paper, we implement one such model with pipeline, and test its performance with respect to facial verification, facial recognition and automated facial clustering. With reference to our results, we discuss the challenges of designing and implementing such systems. Apart from this, we also look into how we can improve the entire pipeline, as well as current concerns with these systems and ways to mitigate them, as scope for future work.

***K*eywords** Facial Recognition · FaceNet · Machine Learning

1 Introduction and Background Work

As a part of the problem statement, we have to be able to identify or verify a person from a digital image or a video frame. There are multiple methods in which facial recognition work, but, they generally end up comparing a set of facial features from a given image to the same set of facial features with-in the database of people. Therefore a unique identification of an individual can be established by analysing the patterns of their facial shape and textures.

Back in the 1960's Bledsoe, Wolf and Bisson started work on recognition systems for human faces. Their initial approach, involved manually marking some pre-determined landmarks for one's face, such as the center of the eyes, the distance between them, the mouth etc. With the help of compute power, these were rotated to account for pose-variation. The distance between these landmarks were compared and used to determine their identity. [1]

More than half a century back, in 1966, Bledsoe described the difficulties to be stemming from 'great variability in the head rotation and tilt', 'lighting intensity and angle', 'facial expression' and 'aging'. He particularly described that the method of pattern matching in unprocessed optical data used by the researchers, would yield very little correlation between, situations such as where the faces were turned in different directions. [1]

After this, the systems that were made for facial recognition can be broadly classified into, biometric techniques, template matching, statistical approaches and neural network based approaches. These biometric techniques, as discussed above try to match the faces based on the lengths and angles between facial features. The biggest issue with such systems is that it's not easy to obtain such lengths with a suitable error bar. Template matching schemes work well when the faces are accurately centred and have similar illumination. [2]

Other approaches on top of this include eigen-faces and neural networks. In the eigenfaces approach by Turk and Pentland, the face image is represented by its projections on the eigenvectors of the training image set. This low dimensional representation can then be used for classification. Notably, the FaceNet system we implement later in this paper, bear remarkable similarity to what's been proposed with the EigenFace system. Anyhow, this demonstrates

that the neural networks have been used for learning on both the original facial data from optical sensors and the representations created of the faces.

With the recent explosions in the amount of compute power, and the availability of clean data due to the rise of social networks, much deeper neural networks have been used for feature engineering on much larger datasets, which build on the success of facial recognition using classical machine learning methods. [5] [6] [7]

[8] uses complex systems that combine a CNN with PCA for dimensionality reduction and a SVM for classification. It uses 25 low-on-compute networks in ensemble methods, each operating on their own facial patch.

2 Approach taken to the problem

Our pipeline consists of the following steps :

- Sensor data → Image pre-processing / Calibration → Non-distorted image
- Non-distorted Image → Facial Extraction → Cropped Image
- Cropped Image → Image Standardisation → 160*160 Face Image
- Face Image → Pretrained Resnet Inception - v1 → Facial Embedding
- Facial Embedding → Clustering/ kNN/ Thresholding → Facial Recognition
- Or Facial Embedding → PCA → Representative Components
- Representative Components → Clustering / kNN/ Thresholding → Facial Recognition

2.1 Camera Calibration

Firstly, the biggest prerequisite is that the visual sensory input needs to be free from optical distortion. Although cameras have been around for a long time, most of the cameras we use with embedded systems and even low cost laptops are pinhole cameras. Unfortunately, these pin-hole cameras are susceptible to significant amounts of distortion. However, these image matrices have some constants (radial and tangential), which can be used to calibrate and remap the pixels into our non-distorted requirements. This can be done by clicking pictures of black-and-white chess boards, that can in turn be recognised by OpenCV to calculate the distortion matrix. With the help of this distortion error, we can calculate the re-projection error, resulting in the non-distorted error.

We eventually found a way to replace the camera we were using with non-distorted ones, therefore resulting in reduced complexity for the pipeline.

2.2 Facial Extraction

Facial detection is a fundamental step in any of the operations carried out in the face recognition pipeline. These algorithms need to consistently perform with high accuracy, and high speed, as every it's going to be used (multiple times) on every single image in the data.

2.3 HOG frontal face detection

A haar cascade classifier is based on an object detection framework proposed by Viola et al in their paper "Rapid object detection using a cascade of simple features". A single classifier is trained using each feature of the training dataset. However, since a single classifier won't really produce high accuracy, we ensemble multiple weak classifiers (Adaboost). Together, the frontal face detection accuracy approximates 95 percent. [10]

Images are used as convolutional kernels, such that features can be extracted from them. Each of the features can be generated by subtracting the sum of pixels under the lighter feature rectangle from the sum of the pixels under the darker feature rectangle. For example, that the eyes are darker than the bridge of the nose. Each feature throughout the entire training set is used to calculate the threshold with which the positive and negative will be differentiated.

In simple terms, we choose the features that best classify the difference between images with a face inside it and images that don't have one. With AdaBoost, we ensemble together all of these weak classifiers to generate a strong classifier, who's accuracy has been described before.

In practice, due to the computational complexity of the training procedure, we're using a pre-trained classifier that came with the paper, to simplify the procedure.

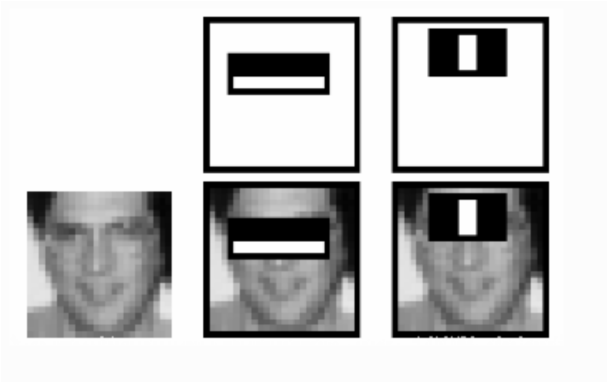

Figure 1: Features for HOG classifier. Taken from 'Face Detection using Haar Cascades' on OpenCV's website

2.4 MTCNN (Multi-task Convolutional Neural Networks) for Facial Detection

The Multi-task CNN is special because it has three neural networks, the P-net, R-net and O-net that allow it to outperform many of the facial detection benchmarks while retaining real-time performance ability. If described in words, the neural network can be described to work in the following way.

Lots of copies of the same image are created with different scalings, which are fed into the P-Net. From the output of the P-Net, the bounding boxes with low levels of confidence are removed. A series of steps converts the kernel coordinates to unscaled image coordinates, followed by non-maximum suppression on first the kernels of each scaled image, and then all kernels. The bounding boxes are then created, which are then shaped as squares.

The boxes produced from the previous step that were out of bound are padded, and the scaled images are fed into the R-Net. With the result of the R-Net, we again delete the boxes with low confidence levels, and do non-maximum suppression for all of the boxes. The bounding boxes are then again un-scaled and reshaped into squares.

Finally, the out of bound boxes are again padded, and the images are fed into the O-Net. With the output of the O-Net, we delete the bounding boxes with low levels of confidence. Bounding boxes are un-scaled and non-maximum suppression is again applied for all of the boxes.

After passing through the three networks, the bounding box with the best confidence level gives the resultant answer. [8]

2.5 Image Standardisation

After the steps described before, we know that the aspect ratio of the images will be the same due to the bounding boxes, however now since we are going to input this into the neural network, we need to make sure, that we standardise the images to consistent dimensions. Going by the results in the FaceNet paper, we choose 160*160 as the appropriate size for the image that is to be passed. This size doesn't lead to features being lost either, while preserving performance and accuracy.

Normalisation and adding central weights to the images weren't performed, because the FaceNet paper talks about being invariant to facial position changes and lighting changes. This leaves the network to be able to learn the feature representation of these different scenarios, as it has been trained to do so.

2.6 FaceNet Model for Feature Engineering

FaceNet is a deep convolutional network. We're running a pretrained Resnet Inception(v1) type network, trained on the VGG Face Dataset. With the given model details, the FaceNet paper treats the model as a black box, with the important part of the approach being with the end-to-end learning of the whole system. It uses a triplet loss function, which directly reflects what needs to be achieved for face verification, recognition and clustering. Therefore, with the embedding function $f(x)$ from an image x, we map it into a feature space of R^d, such that the squared distance between all faces, independent of imaging conditions, of the same identity is small, and the squared distance between a pair of face images from different identities is much larger.

The motivation for this triplet loss function is that other loss functions make all faces of the same identity are project into a single point in the embedding space, however the triplet loss enforces a margin between each pair of faces, from

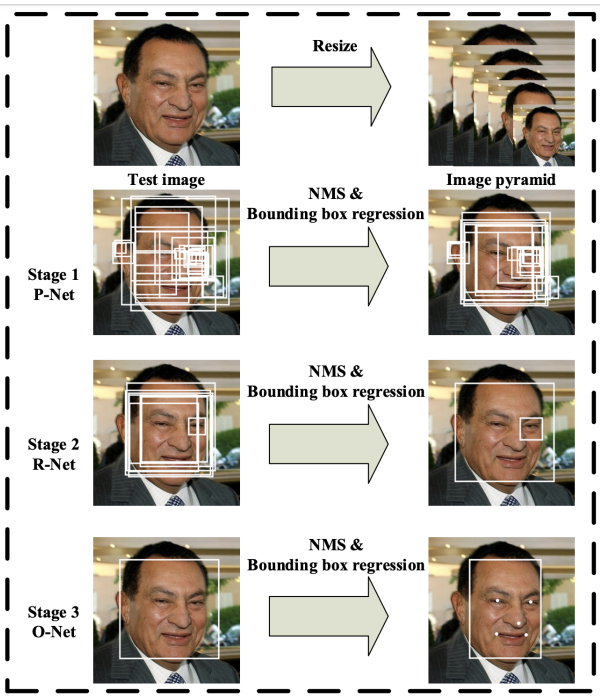

Figure 2: Facial Detection with MTCNN

one person to all other faces. [FaceNet] This allows the person with the same identity to be discriminable from other identities, while being able to identify two different faces with same identities.

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \,, \tag{1}$$

$$\forall\, (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathcal{T} \,. \tag{2}$$

$$\sum_i^N \Big[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\Big]_+ \,. \tag{3}$$

Figure 3: Triplet loss function from Facenet paper [8]

The embedding is represented by $f(x) \in$ & $\mathbb{R}$. It embeds an image x into a d-dimensional Euclidean space.

Additionally, we constrain this embedding to live on the d-dimensional hyper sphere, i.e. $||f(x)_2|| = 1$. [8]

2.7 Facial Verification

Since our pipeline will output similarity value between faces, and the verification problem is inherently binary, we have to use a threshold function to be able to differentiate.

A dynamic thresholding method is used to differentiate, on the basis of the Cosine Distance (Euclidean distance also tried out) between same identities in different angles, and different identities in the same angles.

2.8 Facial Recognition

To recognise patterns, the k-nearest neighbours algorithm is used, which is a non-parametric method. With the k closest training examples in the feature space, the output would be a class membership. The given embedding of the testing

face, can be classified by the plurality vote of its neighbours, with the object being assigned the class most common amongst it's k nearest neighbours.

2.9 Facial Clustering

Facial clustering is the usage of unsupervised learning for the classification of faces amongst unlabelled faces. It's an important metric in understanding the performance of our model and embedding.

K-means is one of the simplest unsupervised learning algorithms to solve this problem, with the number of k clusters fixed apriori, we define the k centroids for each cluster. With random allocation of the centroids, we find the clusters from the closest points to the nearest centroid. This centroid then becomes the center point of the cluster, and this process is looped.

DBScan is also used for clustering, as it's a density based clustering non-parametric algorithm. Given all the embeddings in space, the DBScan algorithm is used to cluster together all the points that are closely packed together, and marks the points that lie alone in low-density regions as outliers.

3 Our dataset

Figure 4: Sampled images in our own training dataset

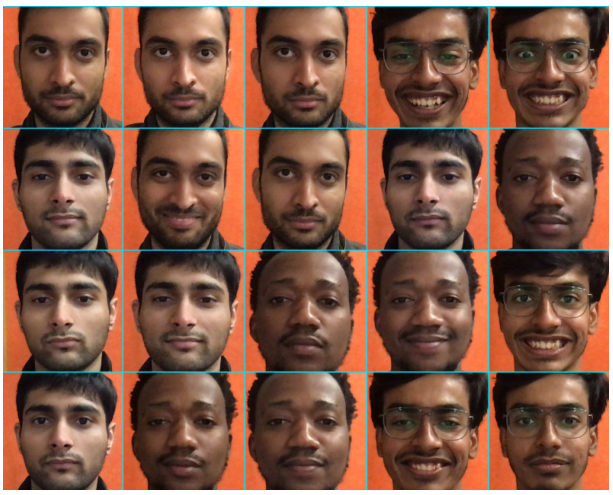

Figure 5: Sampled images from faces extracted from our dataset

4 Results

4.1 Facial verification

Table 1: Accuracy of pipeline for face detection

	Our dataset	**5 celebrities dataset**
Frontal Face HOG	90%	67%
MTCNN	100%	100%

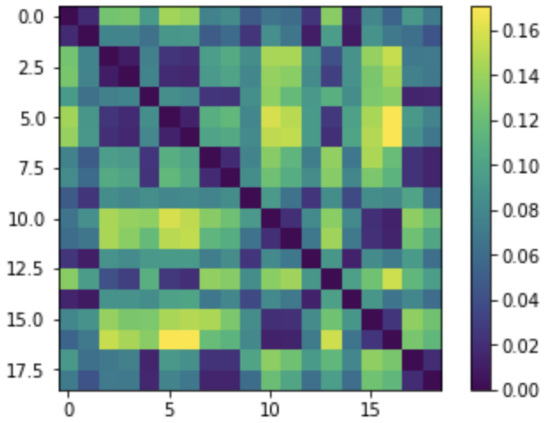

Figure 6: Confusion matrix for cosine distances in training dataset. Although variance looks high, the optimal threshold is around 0.07.

Table 2: Facial verification on our dataset

Pairwise testing sets	**Accuracy**
1 set	100%
3 sets	100%
5 sets	100%

Figure 7: **Angles and lighting variance.** A threshold value of 0.07 makes sure that images of the same person from different angles are correctly verified, while images of other people are not. This is an example of the 3 $\times$ pairwise test.

4.2 Facial recognition

Table 3: Facial recognition using K-nearest-neighbors (on our own dataset)

k	Accuracy without PCA	PCA accuracy
k = 1	0%	NA
k = 3	33	NA
k = 5	40	NA

4.3 Facial clusering

Table 4: Facial clustering

	k-means	DBSCAN
Training accuracy	30% $\pm$ 10%	30% $\pm$ 10%
Testing accuracy	25% $\pm$ 8%	50% $\pm$ 0%

5 Discussion and Conclusion

These are very interesting findings. With our Facial Verification accuracy being at 100% inspite of facial pose changes and illumination changes, we can say that our system definitely works well.

MTCNN provided 100% facial detection accuracy, for our pipeline, however the Haar Cascade classifier was simpler to use and faster, therefore was used. For a better overall performance, we'd love to have tested with MTCNN on the Labelled Faces in the Wild dataset, which would have been the absolute state of the art test, however much more computationally intensive.

We hoped for better results on the facial recognition and facial clustering tasks, however we didn't get them, due to our testing dataset being small, and the curse of dimensionality leading to the points being scattered very close in some dimensions, which are to indicate similarity, however the distances being moderate in all the rest of the 128 dimensions, resulted in increasing the distances, therefore the mixed results.

In any case, our facial verification task was based on the cosine similarity, which is a better representation in this hyperspace, than euclidean distance, which is why we managed to get such accuracy. If we were to do this again, we would definitely use the Curvilinear Component Analysis method to be able to better extract the important dimensions from the results.

By adding PCA (Principal Component Analysis) to the mix, we were able to generate lower dimensional representations, however from the few tests we ran, we were not able to get satisfactory enough results, as they were again based on euclidean distances. We were running out of time, and running PCA to the pipeline would mean having to change our code base significantly therefore, the testing wasn't completed in time.

We also wrote programs to deconstruct the neural network, by generating the image of the filter in the neural network to get a better look at what features were actually being detected, at different stages of the network. Unfortunately, the number of the parameters was so extensive that even with GPU enabled computation, 8 hours of continuous processing on Google's Collaboratory didn't get us even to 10% of the total computation we had to do.

Finally, there's so much more scope for discussion about improving facial recognition technology, since we found a significant bias against classifying identities of African origin in our data set. Since we didn't have enough targeted tests to verify our hypothesis, the results for the same have not been included.

It would definitely be interesting to deconstruct this model and check what the learnt features were, and do the same with another model trained on a dataset with no racial bias. This would be a good scope for future work as an experiment towards identifying bias in facial recognition systems.

References

[1] de Leeuw, Karl; Bergstra, Jan (2007). The History of Information Security: A Comprehensive Handbook. Amsterdam: Elsevier. pp. 264–265. ISBN 9780444516084.

[2] R. Brunelli and T. Poggio " Face recognition : Features versus Templates". https://ieeexplore.ieee.org/document/254061.

[3] Lotlikar, R and Kothari, R : "Facial Recognition using Curvilinear Component Analysis". https://ieeexplore.ieee.org/document/687126.

[4] M. Turk and A. Pentland, "Eigenfaces for recognition". https://www.mitpressjournals.org/doi/10.1162/jocn.1991.3.1.71.

[5] Parkhi, et al. "VGG Face Descriptor". https://www.robots.ox.ac.uk/~vgg/software/vgg_face/

[6] OpenFace. https://cmusatyalab.github.io/openface/

[7] Sandberg, David. Face Recognition using Tensorflow. https://github.com/davidsandberg/facenet.

[8] Kalenichenko, et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering". https://arxiv.org/pdf/1503.03832.pdf.

[9] "Face Detection using Haar Cascades." https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html.

[10] Zhang, et al. "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks." https://kpzhang93.github.io/MTCNN_face_detection_alignment/paper/spl.pdf.