Face augmentation using Facebook Graph Data

Vaibhav Aggarwal (vaibhavg@stanford.edu)

Abstract

In this project I experimented with different techniques to identify people in a camera frame. I tried feature matching and image embedding techniques. I found that image embedding based on Siamese network had the highest accuracy and was most mobile friendly. The technique yielded in 82% face identification accuracy. It took less than 5 millisecond to identify a person from a database of 1 million images.

Introduction

The objective of this project is to propose a computer vision technique which can be used to identify faces from a database of faces. This is usually known as face identification or face verification problem. This problem is different from face detection problem which intends to just find all human faces in a given image (which is already a largely solved problem with most commercial cameras being able to do it with reasonable accuracy). This problem takes the detected faces and classifies them as one of the human face. In a way this is an N-Way classification problem where N is the number of people in the database. The goal of this project is to also go one step further and optimize the solution for a mobile platform which is more restricted in cpu, memory and network bandwidth availability.

This algorithm can be used to implement an Android application which will augment the faces of people in an image with their names in real time. A user would be able to use this application to augment the incoming camera frame with facebook user info in real time. This application is also useful to augment the memory of people with not just their visual appearance but with other information like names, relationship etc. Figure 1 illustrates an example of what is possible with such an application.

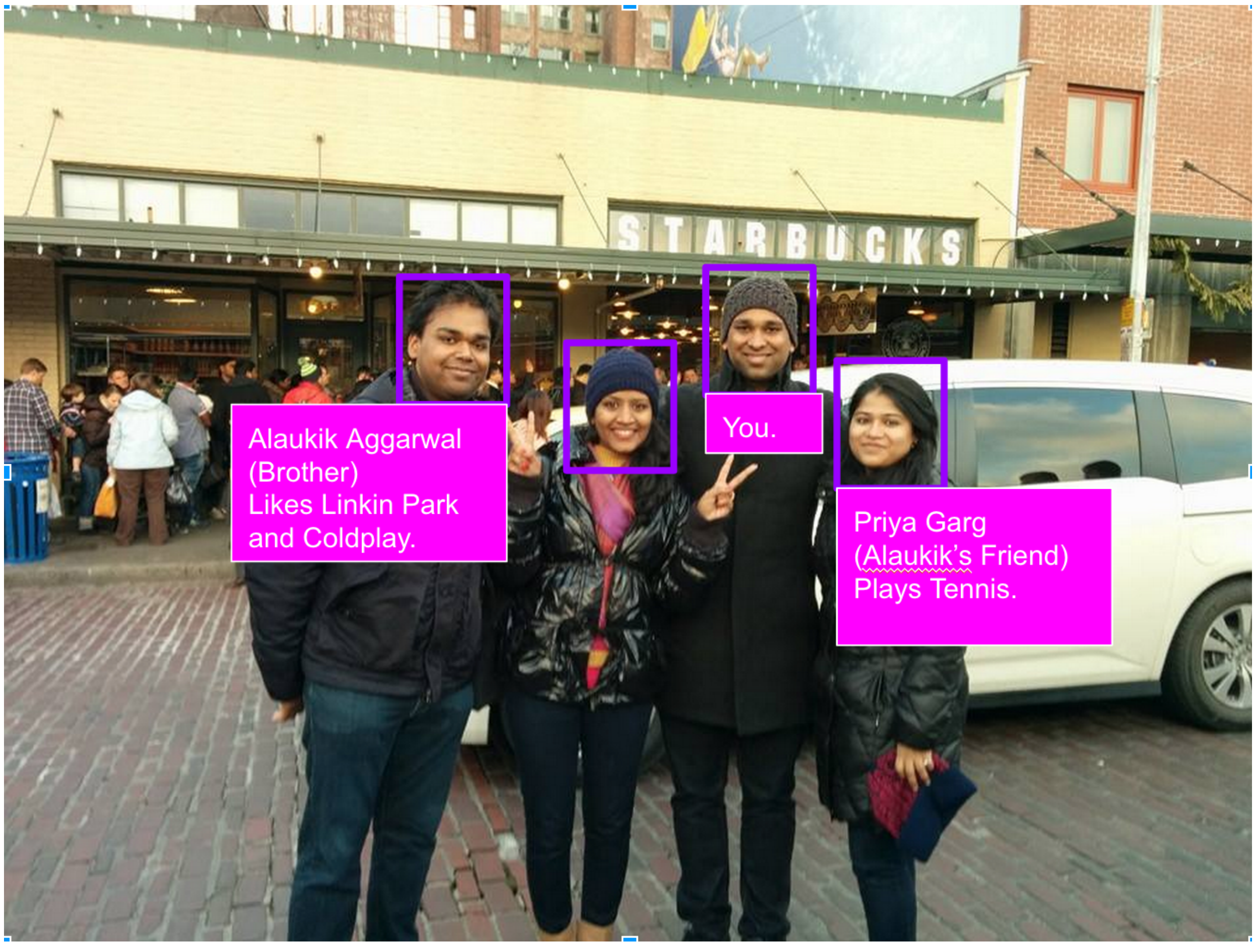


Figure 1

Previous Work

The idea of face identification is very popular and there are a number of researches which are focus on solving this problem. The traditional techniques involve using SIFT [12] and other feature extraction techniques to match similar objects. One shortcoming of these approaches is that they are sensitive to lighting conditions, geometric transformations of the input images (shift, scaling, rotation) and to other variabilities (changes in facial expression, glasses, and obscuring scarves). More recent techniques attempt to use neural networks for classification. A very popular paper by Facebook on Deep Face [3] explores using deep networks to learn representation directly from the pixels of the face. The features do not come from prior knowledge about the task but they are learned from data. When used with a convolutional network as the mapping function, the proposed method can learn a wide range of invariances present in the data. The problem with such a network is that it requires myriad of data to train and avoid overfitting small data sets. Our approach is somewhat similar to that of [1] which uses a siamese architecture for signature verification. The difference is that the network is simpler and easier to train. A paper by Google on FaceNet [2] extends this technique to generate N-Dim neural networks which are again very deep in nature. They also use a large dataset of labelled for training.

Technical Details

Summary

The main focus of this project is to implement techniques to accurately predict the person in picture. I have outlined the following initial algorithm:

Offline Process:

1. Retrieve the database of images.
2. Compute SIFT/ORB features from the profile pictures.
3. Create a database of features mapped to profile information.
    a. Explore algorithms to build a single unique feature for a person combining features from multiple images.
4. Explore indexing schemes for fast searching this database based on feature similarity.

Online Process:

1. Use Android API to detect faces in current camera frame.
2. Send the image across to search server.
3. Compute features from the faces.
4. Apply feature matching or image embedding to compute the most likely classification for the image.

DataSet

There are a number of different datasets available for face data. But there are some particular properties that were required for this problem. We focus on similarity between profile pictures which tend to be closeup view of a person with clearly visible facial features.

The following parameters were looked at before choosing a dataset:

1. The dataset should be fairly recent to match the quality of pictures found on facebook.
2. The data should have thousands of images if not hundreds of thousands in order to support training deep neural network.
3. The data set should have multiple different images of same person in order to allow training and evaluating face similarity of a person.

I researched a number of different datasets Yale [10], CMU [11], FaceScrub [9] etc. I used FaceScrub because it had a large collection of good quality images (approx. 100,000 images) and they were very close to what a profile picture would look like.

Feature Similarity

The first approach I tried was to use feature similarity. I followed the following Algorithm:

1. Extract SIFT/ORB features from training images.
2. Use k nearest neighbor matcher to get feature matches for query image.
3. Discard the features with do not pass the ratio test with threshold of 0.8.
4. Compute homography using RANSAC from those features.
5. Compute the number of inliers from the matching featured.
6. Use that as the score to rank image matches.

Image Embedding using Siamese Network

The second approach I using Siamese network for training and classification of images. Its called a zero one classification where zero means the two images match and one means they do not match. The idea is so take a pair of image like X1 and X2 and feed them to two separate convolutional towers. The two towers actually share the same set of weights and these convolutional towers are used to learn features from two images. Then these extracted features are passed to a fully connected layer and eventually to a loss function. The most important part here is the loss function. We used contrastive loss function which maximizes the distance between dissimilar images and minimizes the distance between similar images. This technique produces an n dimensional vector representation of an image in Euclidean space. Hence it is quite easy to simply use L2_NORM to compare distance between 2 images. The following table explains the neural network used for training.

Table 1

Layer	Output	Kernel Size	Stride	Bias Type	Pool Type
Img 1 Data					
Img 1 Convolution 1	20	5	1	Constant	
Img 1 Pool 1		2	2		Max
Img 1 Convolution 2	50	5	1	Constant	
Img 1 Pool 2		2	2		Max
Img 1 Fully Connected	500				
Img 1 ReLU					
Img 1 Fully Connected	10				
Img 1 Fully Connected	2				
Img 2 Convolution 1	20	5	1	Constant	
Img 2 Pool 1		2	2		Max

Img 2 Convolution 2	50	5	1	Constant	
Img 2 Pool 2		2	2		Max
Img 2 Fully Connected	500				
Img 2 ReLU					
Img 2 Fully Connected	10				
Img 2 Fully Connected	2				
Contrastive Loss					

Experiments

Feature Matching

I tested feature matching based approach using ORB and SIFT. Even though this technique works well for a lot of computer vision tasks, it is not very well suited for face identification. Two main problems with this approach are:

1. The input images have very similar local features like the corner around the eye, lips and hairs etc.
2. If you take the similar features and do the geometric verification they turn out to me geometrically coherent because most people have a very similarly placed local features like eyes ears lips etc .

Figure 2 illustrates an example of a bad match. You will notice that eyes, hair and nose feature got matched with the person on the right. It yielded a very high inlier count even though the two people are different.
I also tried two different approaches, the first one was to use the full image for feature matching and the second was to identify faces in the image and use the cropped image. In all cases cropped face image yielded better results.

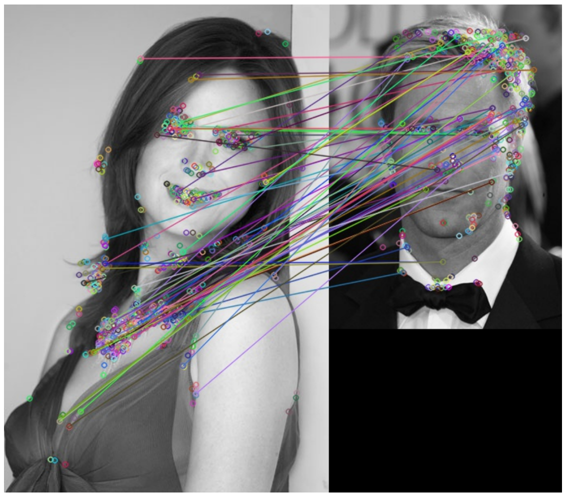


Figure 2

I used two different metrics to measure quality:

Classification Accuracy

This was measured by classifying test images against a database of images. For identification the matching image with highest inlier score (or least distance for later techniques) was used. Please note that we typically have more than one image of a person in the database. This is essentially measuring N-way classification accuracy where N is the number of distinct people in database.

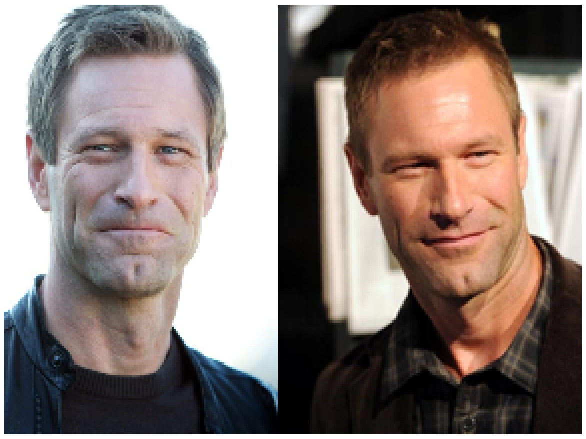
Pair Wise Matching Accuracy

This was measured by computing 2-way classification accuracy (of image pairs) as same or not.

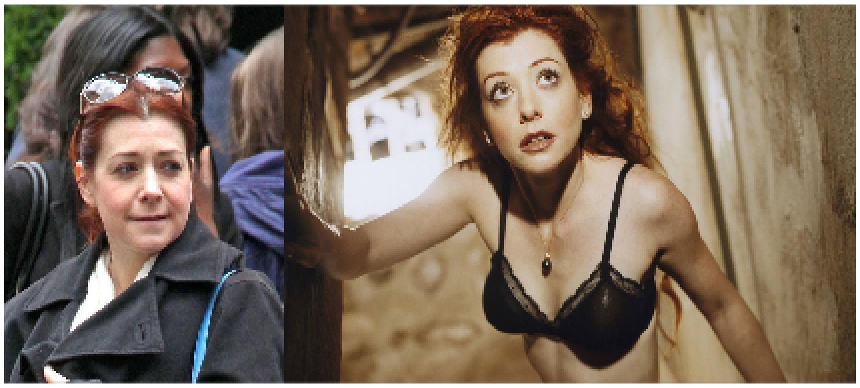
SIFT had 33% classification accuracy and 76% pair-wise accuracy. ORB had 40% classification accuracy and 60% pair-wise accuracy. For this project classification accuracy is more important than pair-wise accuracy. The details are described in figure 5 and 6.

Image Embedding using Siamese Network

The network was trained on 10,000 image pairs and tested on 1,000 image pairs. This yielded much better classification accuracy of 82% and pair-wise accuracy of 77%. Some of the results are shown in figure 3. You can notice that the same person in very different situations is also classified correctly.

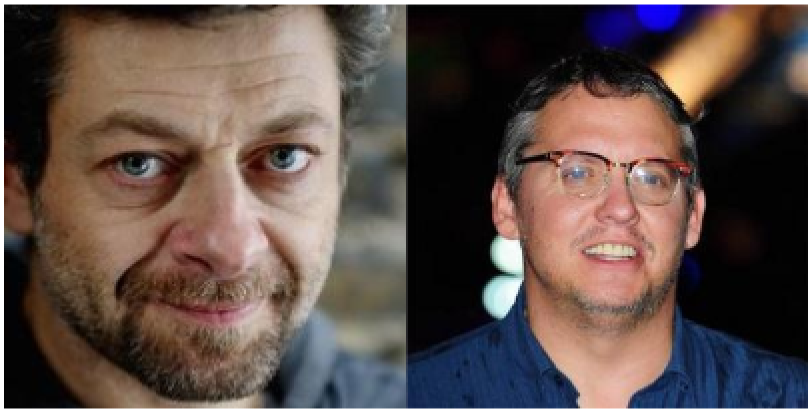


Good match even with different lighting conditions.



Good match even with pictures taken in different setting at different age.

Figure 3



Bad Match

Figure 4

The following chart illustrates classification accuracy comparison.

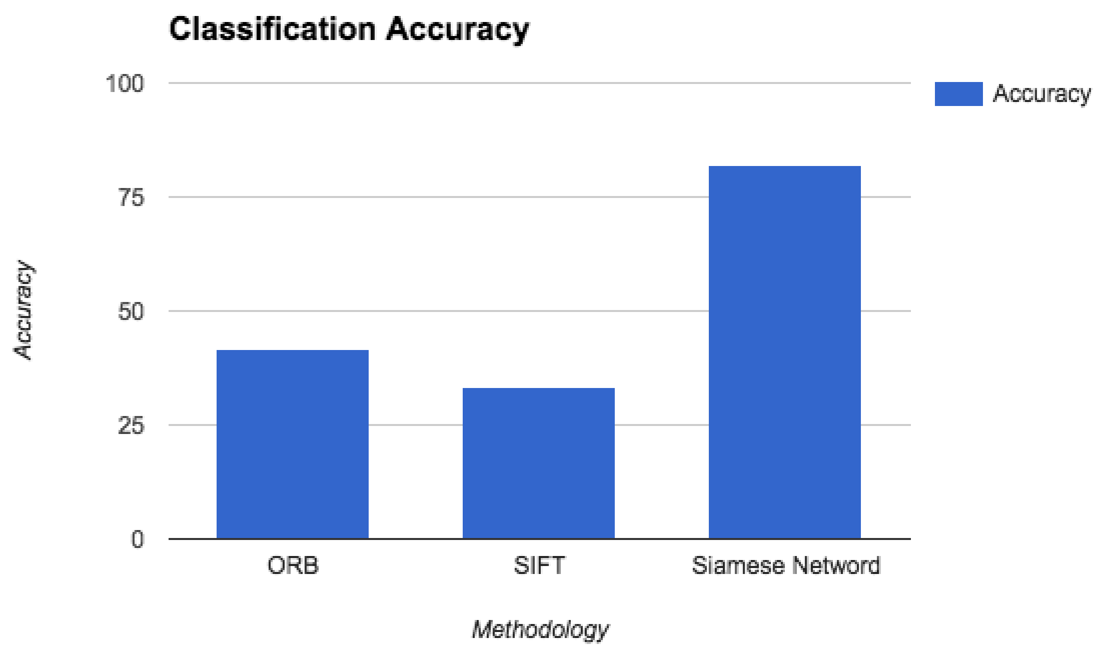


Figure 5

The following chart illustrates pair-wise matching comparison.

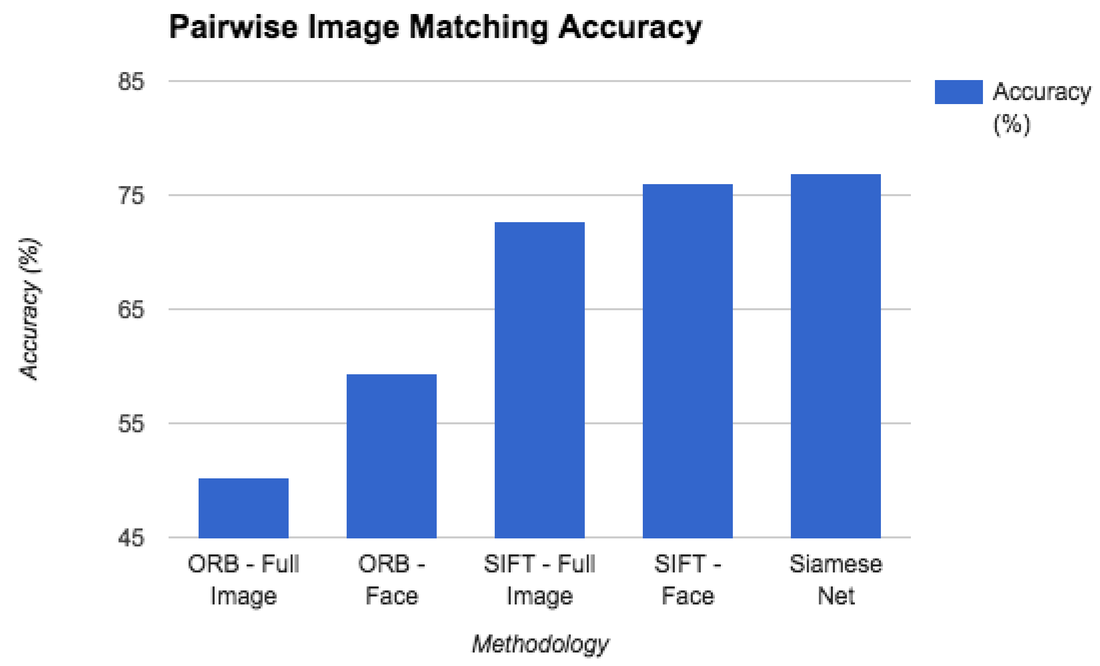


Figure 6

Usage Statistics

I also collected a number of usage statistics to make sure that the techniques are viable for mobile platform.

Database Creation Latency

The database creation latency which is a one time process is reasonably low for all techniques. Siamese network, once trained, does particularly well on this. Furthermore it can be parallelized to multiple servers.

Technique	Num Database Images	Time to process (sec)
ORB	10000	800
SIFT	10000	800
Siamese Net Inference CPU	10000	332
Siamese Net Inference GPU	10000	291

Inference Latency

The inference latency is reasonably low for all techniques. It is particularly low of Siamese network.

Technique	Num Images	Latency per Query (msec)
ORB	1	8
SIFT	1	8.5
Siamese Net	1	0.005

Database query latency and Memory Usage

The database query latency is very high for ORB and SIFT (order of minutes) even for 10K images. Hence it did not make sense to test them on larger dataset. It makes them unsuitable for database query in real-time (unless a vocabulary tree is used). The inference latency of database query built using image embedding is order of 3-9 millisecond. The memory usage us also reasonably low.

Num Database Images	Latency (msec)	Memory Usage (MB)
100K	3.438	19.2
1M	4.381	93.19
10M	9.678	806.02

Conclusions

In this project I presented 2 different techniques of face verification namely feature matching and image embedding using Siamese Network. Out of these two approaches Siamese network definitely seems to be better suited for face verification. First of all it proved to be quite accurate (with scope of further improvement using deeper networks and triplets described in FaceNet paper [2]). We were able to get 82% classification accuracy with our model. Also it can be used to embed images in euclidean space which makes it very efficient to query for similar images. Hence it is well suited for mobile applications. The one advantage traditional feature matching based approach still has is that it requires lot less data to train. Overall I showed that it is possible to do face verification on a mobile platform.

References

1. Learning a Similarity Metric Discriminatively, with Application to Face Verification
   http://yann.lecun.com/exdb/publis/pdf/chopra-05.pdf
2. FaceNet: A Unified Embedding for Face Recognition and Clustering
   http://www.cv-foundation.org/openaccess/content_cvpr_2015/ext/1A_089_ext.pdf
3. DeepFace: Closing the Gap to Human-Level Performance in Face Verification
   http://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf
4. Training Siamese network with Caffe
   http://caffe.berkeleyvision.org/gathered/examples/siamese.html
5. Hierarchical classification of images by sparse approximation
   http://cvgl.stanford.edu/papers/imavis13_sparse.pdf
6. CNN and Random Forest
   http://web.stanford.edu/class/cs231m/lectures/lecture-8-machine-learning.pdf
7. Opencv face recognition.
   http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html
8. Android face recognition
   http://developer.android.com/reference/android/media/FaceDetector.Face.html
9. FaceScrub dataset
   http://vintage.winklerbros.net/facescrub.html
10. Yale dataset
    http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/download.html
11. CMU dataset.
    http://vasc.ri.cmu.edu/idb/html/face/frontal_images/
12. Distinctive Image Features from Scale-Invariant Keypoints
    http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf