Supervised Deep Learning
For Real-Time Face Recognition

Amir Sotoodeh AASOTOODEH@CPP.EDU
California Polytechnic University: Pomona
3801 W Temple Ave
Pomona, CA 91768, USA

Abstract

Facial recognition is a prominent topic of discussion as an application to a variety of systems pertaining to security measures. In terms of machine learning, face recognition has always been an abstract and seemingly complex task with respect to computer vision given that facial landmarks must first be accurately examined and observed.

Although there have been many recent advancements in facial recognition, the greatest challenge to overcome has been the means of finding an efficient and quick ways to compute facial similarities or matches. A typical deep feed-forward neural network can be quite computationally expensive in its forward and backward propagation of data, requiring costly hardware to keep up with its capabilities. Hence, the task of real-time facial recognition for security purposes is a difficult task to complete with respects to limited hardware and efficiency.

This research paper looks into various state-of-the-art techniques that can potentially be utilized for facial recognition via hardware such as webcams and/or security cameras in real time. Additionally, it will discuss a history of face recognition and provide an overview of the task in how it is accomplished.

Keywords: Deep Learning, Supervised Learning, Convolutional Neural Networks

1. Introduction/Background

There are many uses for facial recognition; namely, in video surveillance and aiding in search of specific criminals from available cameras in public. Additionally, it can also be applied as a safety/security measure for high authority officials to enter restricted premises. As such, it is an extremely active field in research by many data scientists and ML enthusiasts.

Given that there exists a surplus of data from the proliferation of social media in recent years, powerful machine learning and statistical models could be built to an alarmingly high accuracy rate. This technology can even be seen in Apple's or Facebook's photo face recognition feature when searching for a name; it is commonly asked to tag a specific individual after detection of a face in a photo.

This goal of this project is to research the potential usage of machine learning models such as deep convolutional neural networks to solve the task of facial recognition in real-time video feed. Additionally, research will be conducted to obtain a greater or more efficient performance of this task by using state of the art statistical models while training on large public/open source dataset.

2. Review of Literature

The several works discussed in this project will be based on the papers within the references section. According to [3], In 2014, DeepFace had acquired the state-of-the-art accuracy on the famous LFW benchmark, and had later been dramatically improved to 99.80% in the next three years. When referring to Google's FaceNet, it became apparent that larger datasets contribute

largely to the accuracy of such models; additionally, no one other than Google had access to the 260 million images that were used to obtain its high accuracy. As such, this inspired varying architectures that required less data. In this project I will refer to and utilize several different deep learning architectures mentioned in [1][2][3], namely, variants of Inception (v1 and v2). In addition to these models, I will also discuss the history of the progress of facial recognition and the several different approaches taken to complete the task.

3. Conceptual Outline

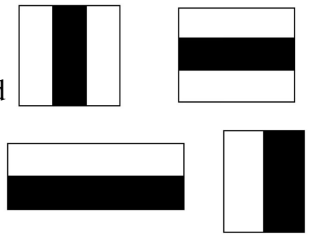
The task of facial recognition is typically broken down into three steps:

1. Facial Detection
2. Facial Alignment*
3. Facial Recognition

Each of these tasks are uniquely difficult in their own nature because it is difficult to define and detect the complex features of a human face, let alone a matrix of pixels. Facial alignment is not discussed in detail but is mentioned in Facebook's implementation of DeepFace [3]. However, the goal of this paper is to provide an introduction into computer vision processing in dealing with human faces. As such, the aforementioned topics are briefly discussed for educational purposes in demonstrating what I have learned in the span of thirteen weeks.

Face Detection

The task of face detection will seem relatively similar to the nature of how convolution works. Several face detection techniques include: Histogram of Oriented Gradients SVM, Haar Cascade classifiers, and Cascading CNNs. With respect to performance in speed, Haar Cascade and HOG methods are preferred due to its ability to reduce unnecessary computations on regions of the image in which no facial features are detected.



Haar features are a set of filters (much like convolution filters) that can be used to detect lines and edges. The two figures on the right are such features, in which are applied to a specified region of an image, to which the Viola-Jones algorithm is applied. The goal is to take the sum of the corresponding positions of the black pixels in the original image and subtract it from the sum of the white pixels. This becomes an order of O(n*m) computation where n is the number of rows and m is the number of columns of pixels.

Viola Jones Algorithm: $\Delta = black - white = \frac{1}{n}\sum_{black}^{n} I(x) - \frac{1}{n}\sum_{white}^{n} I(x)$

However, to optimize this computation and understand why computing Haar features is extremely fast, we must first compute the integral image **I** for a given image. The corresponding integral image is produced when every pixel of the original image is the sum of the current pixel value and every pixel above and to the left of the image as displayed in *Figure 1*.

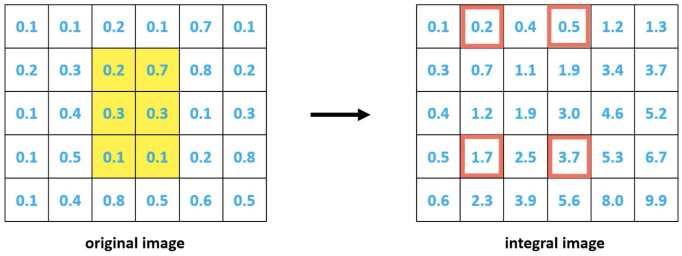


Figure 1

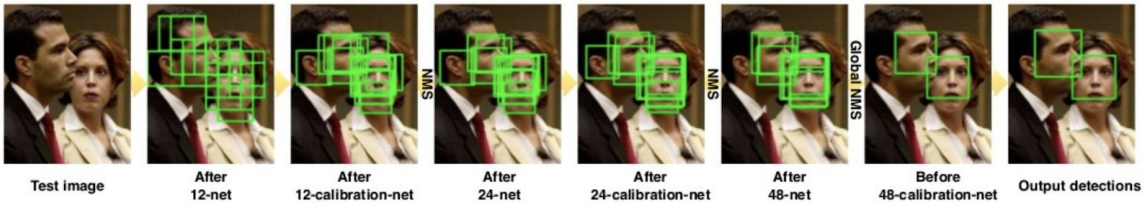
To get the sum of the yellow shaded pixels of an image, we can utilize the integral image values at 4 specific points. Using the following formula, the sum of the shaded region is now a trivial calculation:

$$\sum_{(x,y) \in ABCD} I(x,y) = I(D) + I(A) - I(B) - I(C)$$

where I is the integral image.

As a proof of concept, the shaded region of the previous image is as followed: S = 3.7 + 0.2 - 0.5 - 1.7 = 1.7. Using this as our new algorithm, we begin by applying multiple Haar features of different scales to a sliding-box window across our image. Training a classifier as with each Haar feature as a "weak classifier" ultimately contributes to a final decision of whether or not a feature contributes to a facial feature. The features that have the strongest correlation to a human face are learned by the classifier. Additionally, regions of the image in which have little to no Haar features are quickly skipped over and are not necessary for future computations.

Furthermore, as its name suggests, a cascading CNN is an architecture composed of a series of CNNs that produce a bounding box prediction of the face's region. *Figure 2* displays the pipeline of how a large set of bounding boxes are refined into a few accurate bounding boxes through forward passes throughout the networks.



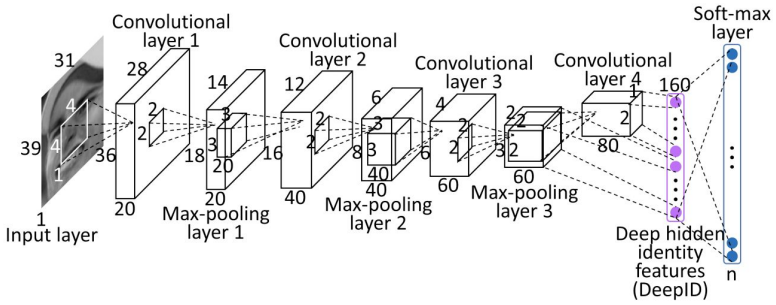
[8] Figure 2

The input of each respective network is a cropped patch of size X (with respect to the network input size, i.e 12-net) from the original image, where the output is a binary classification to represent a facial feature, or not a facial feature. This approach has proven to be very accurate with respect to various conditions of the initial face pose.

In summary, the CNN approach is preferred for uses in which face detection is required to be much more robust and expected to handle faces in varying conditions. On the contrary, for performance oriented tasks, HOG SVM and Haar classifiers are preferred for their faster algorithmic runtimes. HOG SVM and Haar Classifiers have their respective pros and cons: Haar has faster performance but poorer accuracy for detecting human faces in varying conditions, whereas HOG is slower but provides a more reliable accuracy. For this project, a HOG SVM is utilized and is compared to a Haar classifier with respect to accuracy and performance.

4. Related Works

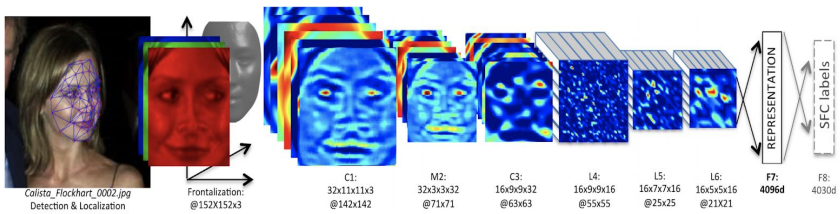
The study of face recognition with deep neural networks had made significant progress with the publication of University of Hong Kong's paper published in 2014 [6]. This paper proposed a series of convolutional neural networks in which suggested translating specific landmarks into a series of 160-dimensional vectors, also known as an individual's DeepID. These landmarks included cropped regions that are centered around an individual's facial features including: left eye, right eye, nose, left mouth corner, right mouth corner. The architecture had also proposed that the target output would be the individual's corresponding identities of which included n>10,000 unique individuals.



[6] Figure 3

Following the development of the DeepID model came improvements to the architecture and ultimately produced DeepID2, DeepID2+, and DeepID3. Each of these models made modifications to how patches or landmarks are selected for processing as well as used different facial landmark detectors. The DeepID models ultimately reached a respectable performance of 99.53% accuracy on the LFW, however began to face problems as the number of classes began to increase. With a significant increase in classes and less face images for each class, the authors noted "the classifier outputs [became] diverse and unreliable, therefore cannot be used as features". This was likely attributed to the chosen loss function of the architecture; the model had no way of formalizing facial features within similar faces, but rather, was training to optimize which class to predict within a specific set of identities. As such, it could be expected that using the DeepID for face verification outside if the ten thousand classes would be much less reliable than that of a DeepID used for someone inside of the dataset.

Following the publication of the DeepID models, Facebook released DeepFace; they had claimed it was a model in which reached human level accuracy of facial recognition. The particularly unique aspect of FaceNet was the face alignment step taken to improve the model's robustness. By performing 2-D and 3-D alignment on detected faces, it is guaranteed that every face will be analyzed in the same orientation and perspective, allowing for a much more consistent analysis of facial features. However, this technology is very difficult to implement and the model required as massive dataset that only Facebook has access to in order to train. Ultimately, the model's architecture was set to produce a 4096-d representation vector in which is trained with cross entropy loss for predicting probability of associated classes.



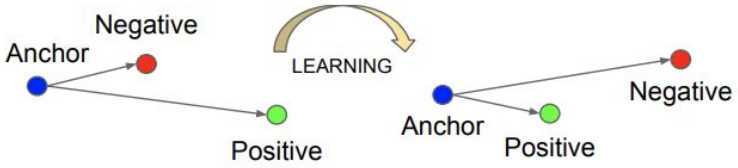
[9] Figure 4

5. Methods Overview

Given the discussion of the related works models, it became clear that the facial recognition task had already been solved with respect to accuracy and most of the available datasets at the time. However, none of the models were robust enough to allow for facial recognition outside of the available training/testing dataset.

For a company to implement a facial recognition system for security, the current model would require a training session every time a new employee were hired. Additionally, a single forward pass of an image would be computationally expensive and would not allow for real-time face recognition. By training machine learning models while using a cross entropy loss function, the model will not train to quantify new facial features into an encoded vector, but rather train to maximize the correct number of classifications within a specific dataset. These are two crucial problems that would need to be solved in order to create a very robust statistical model with high performance.

The aforementioned problems were solved by Google researchers, Philbin et al. with the introduction of a loss metric known as a triplet loss [1].



[1] Figure 5

In contrast to the previous models utilizing a softmax cross entropy loss function, a metric loss function using triplets allows for a quantifiable way to compare and contrast facial features. A triplet consists of three components:

1. Anchor - Image of an individual
2. Positive Exemplar - Another image of the same individual
3. Negative Exemplar - An image of an individual with similar facial features.

Computing a triplet loss:

$$L = \sum_{i}^{N} \left[\left\| f(x_i^a) - f(x_i^p) \right\| - \left\| f(x_i^a) - f(x_i^n) \right\| + \alpha \right]$$

Simplifying into $$L = \sum_{i}^{N} d(a,p) - d(a,n) + \alpha$$

d(a,p) = Euclidean distance of vector encodings of positive and anchor
d(a,n) = Euclidean distance of vector encodings of negative and anchor
α = Hyperparameter to push d(a,p) and d(a,n) apart.

With a triplet loss function, the neural network will find the optimal weights and bias values in which maximize the distance between images of different individuals, while minimizing the distance between images of the same individuals. Once the face is detected and an image of face is cropped, recognition becomes a trivial task with a network trained on triplet loss. A forward pass into such a model will produce a 128-dimensional vector in which corresponds to the individual's unique identity. By computing the L2 (Euclidean) distance between the 128-dimensional identity vector of two images, this value will represent how similar two faces are to one another. Additionally, specifying a threshold value will determine how sensitive or how different a face must be in order to be considered a match or not a match. With respect to

recognition, comparing the 128-d feature vector to a set of known encodings until another vector is found within the threshold accomplishes this task with ease.

FaceNet is proposed with two models, Zeiler & Fergus and Inception/GoogleNet, each of which produce the 128-dimensional vector mentioned with different input sizes.:

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

[1] Figure 6 - Zeiler & Fergus (left) Inception/GoogleNet (right)

6. Objectives

Upon completion of this project, I hope to have successfully experimented with available state-of-the-art machine learning models composed for accomplishing the facial recognition task. With this in mind, the model should ideally accurately recognize my face and any additional specified faces with greater than 90% accuracy in real time. Additionally, the model should be relatively compact and produce an acceptable frame rate when run in real-time. I chose to work on this topic because I found the topic to have many use cases in the industry as well as personal use. Furthermore, this project will be great practice for gaining more experience in the growing field of Machine Learning and Artificial Intelligence.

7. Technical Approach

I began by researching state of the art models with high accuracy with respect to its performance. I prioritized models in which have faster performance and are smaller in size relative to its accuracy; this will allow for a model that can perform adequately in a real-time environment. Once a model or several models had been chosen, I researched for tweaks to the model's architecture as well as any impactful preprocessing steps that may contribute to the overall performance/efficiency of the models. By utilizing face_recognition_models, a python library suited for this exact task, I was able to experiment with networks trained with triplet loss as well as had access to pre-trained face detection models via dlib.

Additionally, it is important to consider the dataset that will be used. Based on the papers referenced, it seems that the LFW (labeled faces in the wild) public dataset is ideal for this task given that they are rotated and cropped in the proper aspect. Lastly, I will attempt to tweak the model's hyper-parameters and/or apply regularization/dropout to obtain a greater accuracy. If the opportunity of training a model myself is not possible, I will explore techniques that can be used to optimize/improve the performance of the face recognition process via state of the art machine learning models.

How can we improve the current state of the art models? From a security standpoint, it is important to consider false instances of a face such as providing an image via an external screen or paper with a face on printed on it. This can be solved by accounting for depth in a CNN model using a stereo-camera or depth sensor. With this CNN model for detection, the model will only run facial recognition when a RGB (with depth) face is detected, rather than an RGB without depth image.

From an accuracy standpoint, we could potentially use GANs to generate effective triplets by creating faces with similar, yet different feature vectors but different labels. This can be extremely beneficial for researchers and/or enthusiasts because due to lack of access to the amount of data that companies like Google or Apple are able to use in the training of their state-of-the-art models.

8. Timeline/Plan

Week 1-2: Research architecture of choice and highest performing techniques used.
Week 3-7: Determine appropriate DNN and detection algorithms
Week 8-9: Model testing/tweaking
Week 9-11: Deployment and Integration
Week 12: Complete research paper and finalize project.

9. Deliverables

Python Code: High quality python code containing pre-processing functions, deep learning models via tensorflow.
https://colab.research.google.com/github/excisionhd/CS5990_Face_Recognition_Assignment/blob/master/CS5990_Face_Recognition_Assignment.ipynb

Github Page: Github page for my peers and fellow computer scientists containing instructions for running the code as well as an in-depth explanation of the methods/architectures used.
https://github.com/excisionhd/CS5990_Face_Recognition_Assignment

10. Resources

Given the complicated and expensive nature of Deep Neural Networks, the use of a GPU will be necessary to train the models that will be used. Additionally, software dependencies will be required:

CPU: Intel i7 7700k
GPU [High Performance]: MSI Geforce GTX 1060 6GB
Python 2.7+
Tensorflow
Numpy
OpenCV
Dlib
face_recognition_modelss

11. Acknowledgements

Thank you to Dr. Hao Ji for the inspiration and opportunity to enter to exciting and amazing field of Machine Learning. Additionally, thank you to all the authors of the following papers and any other references I have used to learn about this revolutionary task.

References

[1] Florian, S., Kalenichenko, D., Philbin, J., (March 2015) FaceNet: A Unified Embedding for Face Recognition and Clustering. Obtained From: https://arxiv.org/pdf/1503.03832.pdf

[2] Kulkarni, H., (May 2018) Deep Learning for Facial Recognition. Obtained From: https://www.researchgate.net/publication/325071878_Deep_Learning_for_Facial_Recognition

[3] Wang, M., Deng, W., (Sep 2018) Deep Face Recognition: A Survey. Obtained From: https://arxiv.org/pdf/1804.06655.pdf

[4] Cen, K, Study of Viola-Jones Real Time Face Detector. Obtained From: https://web.stanford.edu/class/cs231a/prev_projects_2016/cs231a_final_report.pdf

[5] Sun, Y., et al. Deep Convolutional Network Cascade for Facial Point Detection. Obtained From: https://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Sun_Deep_Convolutional_Network_2013_CVPR_paper.pdf

[6] Sun, Y., et al. Deep Learning Face Representation from Predicting 10,000 Classes. Obtained From: http://mmlab.ie.cuhk.edu.hk/pdf/YiSun_CVPR14.pdf

[7] Li, H., et al. A Convolutional Neural Network Cascade for Face Detection. Obtained From: http://users.eecs.northwestern.edu/~xsh835/assets/cvpr2015_cascnn.pdf

[8] Taigman, Y., et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. Obtained From: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf