An Approach to Facial Recognition Technology

Aarushi Upadhyaya

Abstract
This project focuses on creating facial recognition technology using advanced Artificial Intelligence concepts. Similar facial recognition technology has been implemented successfully in Google and Facebook. My approach involves identifying key facial points, implementing them in a similarity transform, extracting patches of the transformed images, and using the patches to confirm the identity of the face with Convolutional Neural Networks (CNNs). My technology was able to achieve a 75 percent accuracy rate. In the future, I plan to add more subjects to the image dataset, as with more variance and increased training, the CNN will become more adept at distinguishing between subjects. This technology can be incorporated into a project idea of creating a facial recognition-based attendance system and can easily be altered to accept image input through a video feed.
Keywords
Key points — Similarity Transform — CNN

Contents

1 **Introduction** 1
2 **Methods** 1
2.1 Identify key points . . . 2
2.2 Implement a similarity transform . . . 2
2.3 Extract patches . . . 2
2.4 Use Convolutional Neural Networks to confirm face's identity . . . 3
Definitions of Machine Learning Concepts • Structure of the confirming CNN • Formatting dataset for input into CNN
3 **Results and Discussion** 4
3.1 Ensuring accuracy throughout the procedure . . . 4
Changing number of dlib points used • Data issues
3.2 CNN results and comparison to current technology 4
4 **Conclusions and Open Questions** 4
5 **Acknowledgments** 4
References 4

1. Introduction

In recent years, facial recognition technology has made groundbreaking strides and some facial recognition programs today even have success rates that rival that of a human. An example of this is Facebook's DeepNet algorithm has a 97.25 percent accuracy, while Google's FaceNet has an accuracy of 95 percent, both of which are less than 3 percent away from the human accuracy rate of which is approximately 97.53 percent. [1] The creation of such astounding technology has numerous applications in our society and has the potential to revolutionize everyday life as we know it. Such technology has even begun to appear in our society, through implementations, such as in the iPhone X, where a user can unlock their phone using their face; social media automatic tagging, where sites, such as Facebook, provide suggestions for tagging users in uploaded pictures; Mastercard's plan for online bio metric payment, where users can pay online, substituting their face as their credit card. Specifically, an implementation of such technology which I was interested in was a facial recognition-based attendance system. My intent for this project was to create the necessary facial recognition technology which will eventually be implemented into an attendance system.[2]

Similar technologies have been attempted before, the majority of which deal with machine learning algorithms. A previous solution to this is the DeepID method, implemented by researchers Yi Sun, Xiaogang Wang, and Xiaoou Tang [3] [4]. The method I created incorporates some of the earlier steps involved in that method. However, my method is unique in that I chose to combine parts of the DeepID method [5] with a simpler face classification algorithm, to ensure the technology could be developed under my projected timeline of 10 months. Throughout this process, I intended to familiarize myself with advanced Artificial Intelligence concepts, such as Comvoluntional Neural Networks (CNNs), and with handling image data.

2. Methods

Acquiring Image Data To create my technology, first, I needed to acquire data to test my program on. I discovered the Facial Recognition Database [6], which stored hundreds of image sets that could be used in various facial recognition-based endeavours. After obtaining permission from the Olivetti Research Laboratory, I downloaded and began experimenting with their AT and T Labs' ORL face database. The dataset contained 400 images total in 92x112 dimensions and was

organized such that each of the 40 subjects' folders had 10 images. The images were already conveniently in Black and White, which made its future implementation simpler, as I didn't need to gray-scale them.

Use of an Anaconda Environment Furthermore, throughout the year, all of my code was developed and tested inside of an anaconda environment. I decided to upon this as, early on in the year, my initial attempts at using the images outside of the environment were plagued with compilation errors due to clashing versions of python libraries and it became cumbersome to continuously resolve them. After implementing the anaconda environment, the problems ceased to interfere with the execution of my code and allowed for an overall smoother implementation of my code.

Approach The approach I took to solving this problem is mainly divided into 4 steps, outlined below:

1. Identify key points
2. Implement a similarity transform
3. Extract patches
4. Use Convolutional Neural Networks to confirm face's identity

2.1 Identify key points

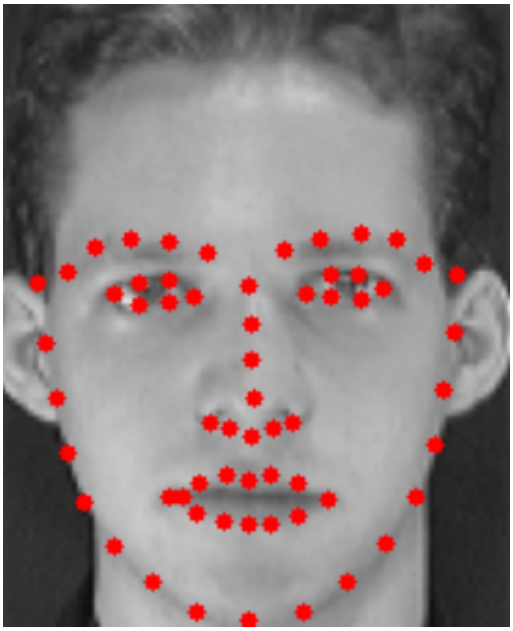

Figure 1. Key points identified in red on a sample image

In this first step, I want to identify certain key points (shown in the image above) which are focused around easily distinguishable regions of the face, such as the eyes, nose and mouth. These points were identified through the implementation of a pre-trained facial landmark predictor, which I found and downloaded during my research on this topic. The implementation of this predictor involved the use of a python library, dlib which mainly deals with image processing and computer vision. After finding the points, I stored them for later use, in the next step of this approach.

2.2 Implement a similarity transform

In this step, the main goal is to have a template image, with the key points identified, and to transform a sample image, such that its key points match up to that of the template image. I was able to create this code, using parts of an implementation of a similar transformation project, which I found on GitHub while researching [7]. The key points that i found in the previous step were crucial in the implementation of this transform. The method I used had four main steps:

1. Subtract key points by mean of points
2. Divide key points by Standard Deviation
3. Use linear algebra method Singular Value Decomposition (SVD) to find transformations, scale factors, and translations necessary to convert the image to the template based on key points
4. Use OpenCV's WarpAffine() to transform matrix with results from SVD

This method was applied to all 400 of the images in my database and the resulting transformed images were stored in a folder structure identical to that of the dataset I downloaded, with 40 folders and 10 images per folder. The images were transformed to fit a sample image with the subject facing the camera straight (See Figure 1 or 3). An example of the similarity transform working is shown below in Figure 3.

2.3 Extract patches

Figure 2. The patches extracted from the bottom left image

After the similarity transform has been performed on all the images, all steps occurring after will only reference the new, transformed images. This step involves extracting predetermined patches of the image and converting them into new images. When extracting the patches, using the similarity transformed images were very convenient in that I only had to set the bounding values for each patch once, as the locations of the key features are in approximately the same location.

I chose to extract 7 additional patches of varying dimensions, using OpenCV, from the facial image. When combined with the original image, this results in 8 total patches. A visual representation of this is included above in Figure 2. Each patch that was extracted focused on a different aspect of the

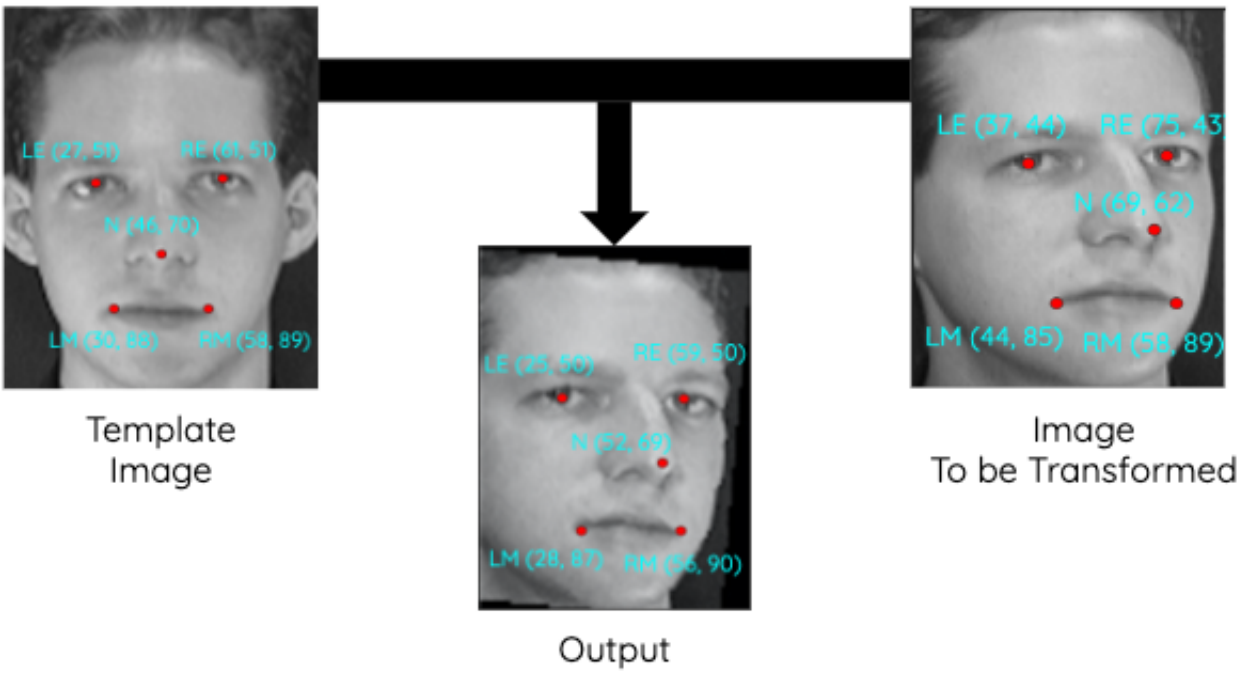

Figure 3. The similarity transform applied to convert a sample image to the template and a visual of the result

face and a different set of key points, such as the nose, or multiple sets of key points, such as an eye and the nose. This allowed for more variance to occur and also ensured that each section of the face was being represented equally.

After extraction, the patches were stored in a new folder structure which was similar to the old one, storing each patch from each of the 10 images in a folder for each subject.

2.4 Use Convolutional Neural Networks to confirm face's identity

2.4.1 Definitions of Machine Learning Concepts

Before delving into the structure and the details of the implementation of the confirming CNN, a few Machine Learning terms must first be defined.

Convolutional Neural Network (CNNs) A deep learning algorithm that receives input and, after a training process, can analyze and make deductions regarding the input

Convolution Layer A layer in the CNN that receives input, in the the case of this study, image data, and has a kernel traverse through the image and extracts high level features from it

Activation Layer A layer that typically follows a Convolution Layer and evaluates the weights of the neurons in the preceding layer

Flatten Layer Typically precedes a Dense layer and converts the image data from a 2D array to a 1D array

Dense or Fully Connected Layer The final layer(s) in a CNN. Evaluated all the Neurons in every Convolution Layer and hones in to the weights to improve overall accuracy

In this study, CNNs were chosen and implemented over their 2-Dimensional counterpart, Neural Networks which are normally quite effective, due to the prevalence of image data in the implemented dataset.

2.4.2 Structure of the confirming CNN

The structure employed in the confirming CNN is 4 Convolution layers, each followed by a "Relu" activation layer. "Relu" or Rectified Linear Unit is a type of activation layer [8]. I decided upon it due to its rapid learning rate and high accuracy. Next in the CNN is a flatten layer followed by two dense layers. Then, code is included that fits my image dataset to the CNN model, and trains and tests it for 10 epochs, or 10 times.

2.4.3 Formatting dataset for input into CNN

Before fitting the dataset to the model, the dataset needed to be formatted in a way that ensured that the dimensions were consistent between the images and the CNN model. The images also needed to be sorted into separate categories: testing or training data. The images were separated such that 70 percent were for training, while the other 30 percent went for testing, a standard data division practice in the AI industry. The images were converted into numpy arrays, all of which were stored in larger, either the training or testing, numpy arrays.

In addition, two numpy arrays were created which stored the identity of the image in the same index in the corresponding image arrays. At each index in these numpy arrays, an integer from 1 to 40 was present, each number representing a different subject in the image dataset.

3. Results and Discussion

3.1 Ensuring accuracy throughout the procedure

This study was conducted in stages, in which the results of the following steps was dependant on the success of preceding steps: the key points identified using dlib were essential in transforming the images using the similarity transform, the transformed images were used in the patch extraction, and the extracted patches were used in the confirming CNN. Thus, alterations to the earlier steps were crucial to achieving a high accuracy value later on.

3.1.1 Changing number of dlib points used

One significant change made to the procedure earlier on in the project that significantly affected the results was going from only using dlib to identify 5 significant facial points: the centers of the eyes and nose, and the corners of the mouth to identifying 61 points. This decision was made mainly after considering how accurate the similarity transform would be when using a limited point set versus a larger point set.

3.1.2 Data issues

An unexpected problem that I faced occurred during a later step in the methodology. While I was implementing the confirming CNN, my code faced multiple compiling issues involving the structure of the input data being incorrect. I struggled with this issue for a significant period of time, attempting various data structures and solutions and my timeline for progress was slightly set back. Eventually, I was able to overcome this issue using the numpy python library's data structures and methods.

3.2 CNN results and comparison to current technology

The main results finding and accuracy determination occurred at the end on the project. Ultimately, after training for 10 epochs, the developed technology resulted in about 95 percent accuracy rate and around a 0.17 loss rate - a value measuring inconsistencies between the identity generated by the CNN and the image's actual identity - on previously seen data, and approximately 75 accuracy rate and about a 1.5 loss rate on new data which the CNN hadn't seen before. In this case, the measured loss is decreasing with each epoch, which is to be expected, and by the 10th epoch, the loss is relatively low. Furthermore, in the context of the future application of this technology, the 75 percent statistic is more relevant as new pictures will have to be taken in and corresponded with its identity.

After finishing the accuracy calculations on the CNN, I also wanted to extend this technology beyond the previously downloaded dataset that was mostly used over the course of the study to ensure the technology could be successfully applied to other subjects. To test this, A 41st subject was added to the existing dataset - 10 images of Vasudha Upadhyaya were obtained, with her permission. After exacting all the steps of the procedure on the new images and retraining the CNN to account for the new images, an image of subject 41 was inputted into the trained model. The output verified that the CNN could identify the image as belonging to subject 41. This verified the original belief on the further potential applications of this technology.

The results are quite similar to that of the current average success rate of facial recognition technology available [1]. However, there is still a large margin for improvement in my method, as many algorithms, such as those developed by Google and Facebook mentioned in the Introduction of this paper, have almost perfect accuracy.

4. Conclusions and Open Questions

The context surrounding the results is the purpose of developing this technology, which was for the eventual incorporation into a facial recognition-based attendance system. Thus, this study accomplished its original goal, which was to create facial recognition technology through the implementation of CNNs. Throughout the course of this study, I was able to gain valuable experience in dealing with image data and learning more in-depth about machine learning concepts. I gained similar results to what I expected with this method. This method contributes to the field of Machine Learning as it provides results on a novel approach to facial recognition technology and combines aspects of two methods, the DeepID method and image classification.

In the future, I plan to expand upon this technology by adding many more subjects to the image dataset, as with more variance and increased training, the CNN will become more adept at distinguishing between subjects' features. This technology can be incorporated into the project idea discussed in the Introduction - an automated attendance system - and can easily be altered to accept image input through a video feed.

5. Acknowledgments

I would like to thank my Syslab director, Mr. White, for his mentorship and support throughout the development and execution of this project. I would also like to thank Dr. Gabor, Dr. Torbert, and Dr. Zacharias for their guidance during the previous academic year, in helping me develop this idea and create my unique approach. I would also like to acknowledge Olivetti Research Laboratory, for allowing the use of their dataset in this project and Vasudha Upadhyaya, for allowing images of her to be incorporated into this study.

References

[1] The top 7 trends for facial recognition in 2019, 2019.

[2] Patil S. Thakare R. Wagh P. Chaudhari, J. Attendance system based on face recognition using eigen face and pca algorithms.

[3] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes.

[4] Qianli Liao. A summary of deep models for face recognition.

[5] E. R. Davies. *Computer Vision - 5th Edition*. Academic Press, 2017.

[6] Face recognition homepage, 2019.

[7] Switching eds: Face swapping with python, dlib, and opencv, 2015.

[8] Danqing Liu. A practical guide to relu, 2015.