Final Project: Face Recognition

Artificial Intelligence

Kshitij Sharma
Worcester Polytechnic Institute
100 Institute Rd, Worcester MA
ksharma@wpi.edu

Saurabh Kashid
Worcester Polytechnic Institute
100 Institute Rd, Worcester MA
skashid@wpi.edu

Jose L. Figueroa
Worcester Polytechnic Institute
100 Institute Rd, Worcester MA
jlfigueroa@wpi.edu

Ravi Teja Alapati
Worcester Polytechnic Institute
100 Institute Rd, Worcester MA
ralapati@wpi.edu

Abstract

The goal of this final project is to comprehend how face recognition works and has evolved over the years. We approach this issue by creating several methodologies and putting them to the test on various subjects and case studies. Python and its libraries were used to create the two tested methods. We take into account the scenario of Siamese neural networks with two inputs in each attempt, which gives us promising outcomes, for our final attempt. For our project, we were able to locate various data sets, and we used data augmentation to give our models more training data. We employ a collection of photographs that were taken with our personal web cams for our final case. The last method was very successful with the data we had. This last approach worked great for our own data set. This is a sound strategy for implementing face recognition more effectively, but it has limitations that are described in this report.

1. Introduction

Facial recognition is the process of identifying or verifying a person's identification by using their face. Facial recognition algorithms can be used to recognize individuals in real time, on-screen, or in images.

Although there is growing interest in other applications such as transactions, population analysis, and attendance, face recognition is mostly used for security and law enforcement.

Finding a face in a frame and detecting it is the initial stage in face recognition. Recognizing several faces in a crowd, in various lighting conditions, and from various angles is the best strategy for this. We then study the face once we've located it in the frame. We rely on two-dimensional (2D) color images with three channels to help us identify facial features. These characteristics could include various gaps between the eyes, the forehead and hair, edges, and a variety of other things. To have reliable landmarks to identify a face, it's critical to be able to recognize these essential characteristics in each individual. [1]

Finally, we develop a matching method that evaluates the stored and observed information and determines if a match or a close match exists. The face is categorized to the nearest face in the initial approach, and in the second approach, we provide a binary response for a positive match or a negative match.

1.1. Data

We filled our data with a collection of images from five distinct celebrities dataset in order to present different subjects to the algorithm. This data was extracted from the Kaggle.com website. After a few tries, we realized that the data was too small, so we utilized a data set called "labelled faces in the wild." The majority of the images in this data set were collected from YouTube films, which in most cases aided our methodology when taking into account the lighting and size conditions. This data set also included faces of many more participants.

Finally, we made the choice to use a unique data set. To classify the data into positive and anchor examples, we gathered approximately 2400 photographs captured using our own web cameras. For the negative examples, we used the faces of other group members. We experimented with several lighting setups and viewing positions to more effectively test our theories.

1.2. Data Augmentation

We increased the photos in our dataset to improve the accuracy of our algorithms. We developed a code that randomly enlarged the images and generated 31 images from a single image. The following augmentations were carried out:

- rotation range (0-360 degrees)
- width shift range (20%)
- height shift range (20%)
- shear range (20%)
- zoom range (20%)
- horizontal flip (20%)
- fill mode (reflection)

Figure 1. Sample of Image Augmentation.

2. Image Classification and Detection with Random Forest

The first approach we attempted is known as VGG16 and VGG19; this approach is more concerned with the number of layers than the hyperparameters. To extract characteristics from photos, this approach is trained using the ImageNet dataset. The final layer of the VGG16 is given to a random forest using our technique, which then initiates a face recognition procedure using image classification.

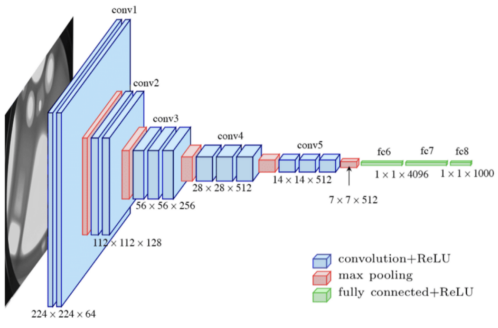

Figure 2. VGG16 Architecture.

The 64-channel input pictures are the same size as used in the Siamese network research paper as the starting point for the VGG16 algorithm, and nonlinearity is implemented using a ReLu activation function. Then, to decrease the height and width of our image, we implemented max pooling layer. And as we move forward, the layer's depth increases which causes the tensor's height and width to diminish. To tackle this issue we implemented same padding with a stride of 2. We linearly lower the layer size in this network. Every time a layer is max pooled, its size is reduced by 50%. We chose VGG since it offers a greater number of trainable parameters. [**?**]

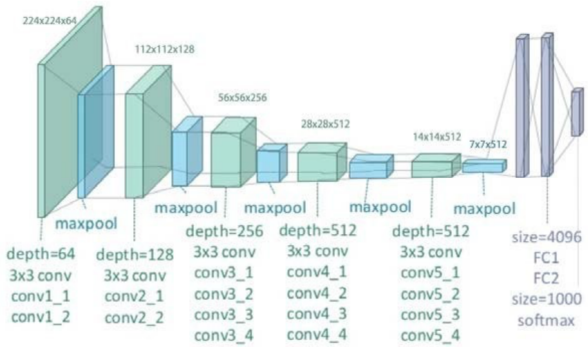

Figure 3. VGG19 Architecture.

Following VGG16, we provide VGG19, which adds three more trainable parameters and raises the number of layers by 3. Additionally, the design has changed, and we now have vertical arrays rather than horizontal arrays, which provide additional information to the final levels. [3]

2.0.1 Methodology

For this strategy, we established several test and training folders, and inside of those folders, we formed various classes and gave the folders names based on the names of the individual classes. In order to have a normalized input, we then go through each folder and each image, and scale them to 256*256. As previously mentioned, we also augmented the data set for each subject. The photos were then transformed into a three-dimensional matrix and stored in an array. We used the pre-processing encoders to turn the folder names into numbers after storing them in the list in accordance with the images. We performed a one hot encoding, which outputs a value of 1 for each image label if the picture belongs to the specified class and a value of 0 otherwise.

The characteristics from each of the photos were then extracted using the VGG16 that we loaded. We were able to detect the topic and recognize each person's characteristics by setting the trainable parameter to false.

The output layer of the VGG16 is then scaled to a vector. The random forest model was then fitted using that vector and the single hot-encoded class value. The photographs in the test folder were then used for the testing. the forecast and accuracy matrix was printed.

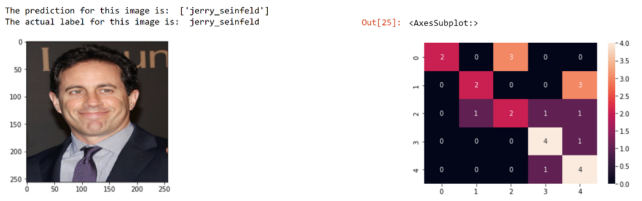

Figure 4. Celebrity Data Set Example.

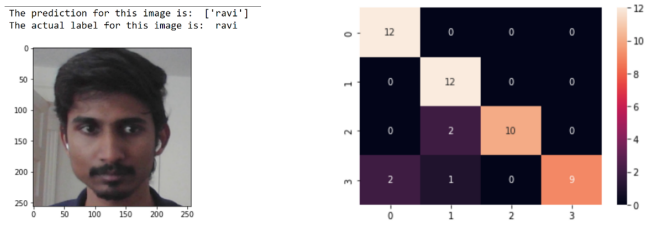

Figure 5. Our Own Data Set Example.

The data above show that our initial strategy is effective. We then test our method using the entire amount of data we have.

2.0.2 Confusion Matrix

A confusion matrix describes how many of a classifier's predictions were accurate and, in cases where they weren't, where they went wrong. The rows in the confusion matrices above indicate the actual class labels, while the columns display anticipated class labels. The diagonal values show how frequently the anticipated label matches the actual label. The values in the other cells show cases when the classifier incorrectly labelled an observation; the row and column in each case indicate the correct label.

The confusion matrix of the five celebrity dataset, which comprises five classes, is shown In Figure 4 right side picture. Images from classes 3 and 4 are shown here; four predictions were accurate and one was erroneous. It was anticipated to be in class 4, but which actually belongs in classes 3 and 4, despite being projected to be in class 4. It predicted two right numbers and three wrong ones for classes 0, 1, and 2.

The confusion matrix from our own dataset is shown on the right side of Figure 5. Class 0 and 1 photos correctly predicted each and every image. Ten of the photographs belonging to class 2 were accurately predicted, while two were successfully forecasted as images belonging to class 1. Similarly, it accurately predicted 9 photos that belong to class 3 and incorrectly forecasted 2 images that belong to class 0 and 1 that belong to class 1.

2.1. Results

The accuracy for our initial approach as shown in the following Table 1:

DataSet\Model	VGG16	VGG19
Celebrity Dataset	56%	51%
Own Dataset	75%	89%

Table 1. Accuracy

We can see that our data set's accuracy is higher than that of the celebrity data set because, according to our analysis, there are fewer images to extract the features from in the celebrity dataset's five stars, while the test dataset contains images taken in a variety of settings, lighting conditions, and angles, some of which also included additional subjects that made it difficult for us to identify the subject in some instances. The celebrity data set also contained images from various years, therefore the age of the individuals also had an impact on the result. On the other hand, when we tested our own data set, we noticed that we had only taken pictures of people's faces, and that because they were taken at varied angles and in varying lighting, the algorithm had an easier time identifying the faces.

3. One shot Recognition

One shot image verification is a common and widely used technique in the field of image verification where we can make classification based on one look at the picture of a person and pick up facial features and compare it with another image it has been presented with and provides a value that shows either how similar or different the images are from each other as shown in the following Figure 6. [2]

In order to perform one shot recognition we have to train a neural network and a special type of neural network at that called a Siamese Neural Network that allow us to have a computation between two inputs.

3.1. Siamese Neural Network

A Siamese network has two parallel same neural networks where different inputs are given to each of the neural networks that output an embedding vector of the facial features and then the vectors from both the input images is compared using another fully connected layer that calculates the distance between the two and output a label based on that result. A basic Siamese neural network appears as the one shown in Figure 7. [4]

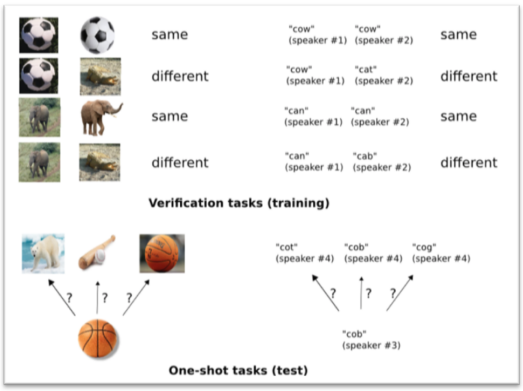

Figure 6. One shot Recognition

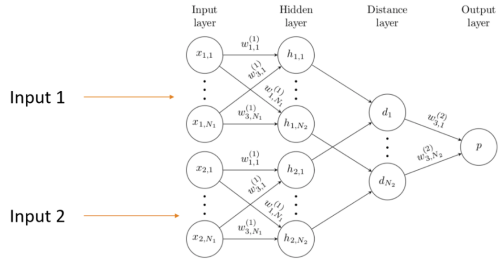

Figure 7. Siamese Neural Network Architecture

Our second approach introduces Siamese Neural Networks that has the architecture shown in Figure 8

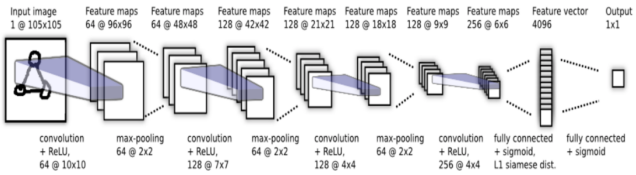

Figure 8. Siamese Neural Network

3.1.1 Methodology

The Siamese network that we implemented in this project included the following components in each neural network:

- Input image of size 100 x 100 x 3
- Convolution layer with 64 filters of size 10 x 10 with ReLu activation
- Max pooling layer with 64 filters of size 2 x 2 with same padding to keep the size on the output of this layer intact
- Convolution layer with 128 filters of size 7 x 7 with ReLu activation
- Max pooling layer with 64 filters of size 2 x 2 with same padding to keep the size on the output of this layer intact
- Convolution layer with 128 filters of size 4 x 4 with ReLu activation
- Max pooling layer with 64 filters of size 2 x 2 with same padding to keep the size on the output of this layer intact
- Convolution layer with 256 filters of size 4 x 4 with ReLu activation
- Fully connected layer 4096 neurons with Sigmoid activation

Finally, we combine the output of 2 neural networks and pass it through another fully connected layer with Sigmoid activation to get an output label.

Figures 9 and 10 show the basic implementation of the siamese neural network in our project.

Our aim is to train the Siamese neural network to learn the similarities and differences between a set of images. Therefore, the neural network is fed with two sets of images either anchor and positive or anchor and negative and the neural network must tune the parameters in such a way that the difference between the embedding of an anchor and positive image is very less and the difference between the embedding of a negative and an anchor image must be substantial.

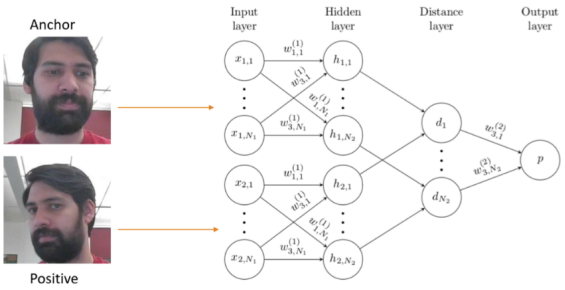

Figure 9. Anchor and Positive Data Set example

3.2. Training Parameters

There were multiple parameters that contributed to the loss, precision and recall evaluation metrics and are shown below:

1. Dataset - varying (900, 6000, 12000, 40000 images)
2. Train/Test Split - 70% / 30%
3. Optimizer - Adam Optimizer

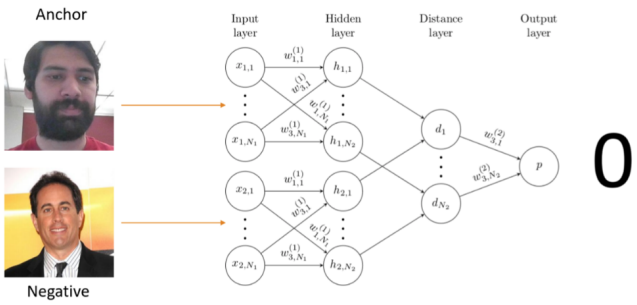

Figure 10. Anchor and Negative Data Set example

4. Learning Rate - 0.0001
5. Loss Function - Binary Cross Entropy Loss
6. Epochs - 50
7. Batch Size - Varied from 16 to 64 on account of varying datasets
8. Number of trainable Parameters in the neural network - 39 million parameters

These training parameters can be varied and the results may vary as well. For example we can use the triplet loss function used in FaceNet and that may change the result or we can increase the dataset and talk about in millions of images etc..

3.3. Results

We performed the training on 900 images of 1 person and trained the neural network on CPU first because of an error that we were facing and not being able to use the GPU in the machine. The 900 images dataset consisted of all the images taken from webcam and were of mediocre quality. It took us 1 hour to train the 900 images dataset on CPU and the results were incredible. We calculated the precision, recall and loss. After enough debugging we fixed the GPU setting for our results and we increased the dataset with augmentation and increased the number of people. We started with 2 people, 3 people and finally 4 people. We calculated the precision, recall and loss for the largest dataset of 4 people. The times spent on the CPU and GPU with the different datasets is as shown in the table 2 below:

No. Subjects	No. Images	CPU Time	GPU Time
1	900	1 hour	120 Seconds
2	6000	5.5 hours	35 minutes
3	12000	11 hours	1 hour
4	40000	20 + hours	5 hours

Table 2. CPU and GPU times

The final results that we were able to generate using the siamese neural network and one shot image verification are shown in the images below:

- In Figure 11 we can see the results that were generated by the siamese neural network and can be seen that the Siamese Neural Network performed very well and was succesful in verification of the input image of Saurabh and determined if it was Saurabh or it was not Saurabh.
- In Figure 12 we see that the results where the neural network failed to recognise the image of Ravi on the right side and incorrectly recognised the image Saurabh on the left. This is the example where the Neural Network being perfectly capable of recognizing 1000s of examples can also falter some times and give wrong results.

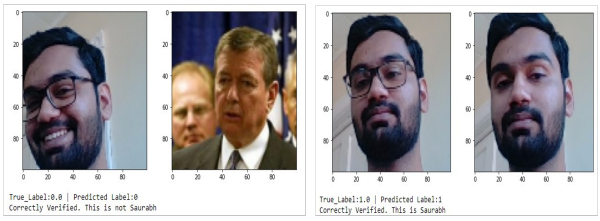

Figure 11. Correct Result Siamese Neural Network

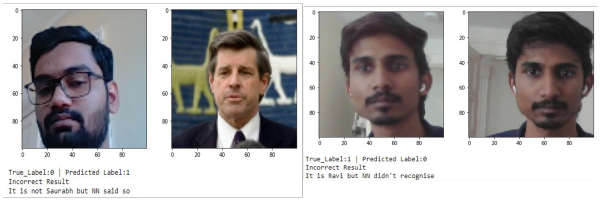

Figure 12. Incorrect Result Siamese Neural Network

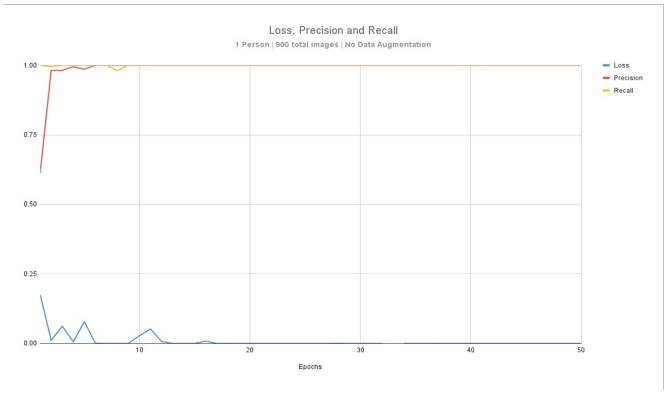

Figure 13. Loss, Precision and Recall for 900 Images

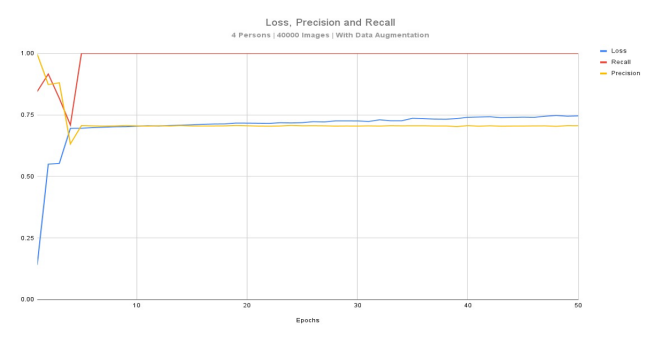

Figure 14. Loss, Precision and Recall for 40000 Images

As shown in figure 13, the result for using just one subject and a training set of 900 images allowed us to have a precision of almost 1, however, at the moment on introducing more subjects this precision drops to around 0.7 as shown in Figure 14, this is a sign that the method is working and it is not over fitting, or just memorizing features in each train image.

4. Discussion

After implementing a facial recognition system, we have encounter that computer vision and object detection relies heavily on the quantity and quality of the data. In most cases we were able to perform a recognition, however, in a small portion or our experimentation the network fail to recognized a face; the majority of the fail attempts occurred for subjects that had a big variance in the input and testing data. In the other hand, the results that we obtain using our own image data set perform smoothly and it was able to recognized each member of the group. The approach that we introduce using Siamese Neural Networks is novel and can perform more efficiently if the right computer power is available. Another factor that we analyzed was the image size, we choose to use low resolution images and have more samples, than having high resolution images with fewer samples.

5. Limitations and Future Work

5.1. Real-time Face Detection

In our project, we were only dealing with recognizing faces in images. We would like to extend our work to enable it to recognize faces in a real-time video stream. We did some preliminary work trying to implement it.

The most common way to detect a face (or any other object), is using the "Haar Cascade classifier". Object detection using Haar feature-based cascade classifiers is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. For face detection, a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. If we do not want to create our own classifier, OpenCV already contains many pre-trained classifiers for detecting faces and also specific features of a face. We could detect a face in a real-time video but unfortunately failed to integrate our trained model to recognize whose face is it.

5.2. Comparison with FaceNet

FaceNet is a deep neural network used for extracting features from an image of a person's face, published in 2015 by Google researchers Schroff et al. The netwrok has been trained with datasets of millions of images, and it is said to have an accuracy of over 99 percent. We tried to use the FaceNet to recognize our faces in our own dataset, but could not succeed in implement it. It would be been interesting to compare our trained model with the FaceNet for our dataset. This is the first limitation of the project and we would like to extend even this task as our future work.

5.3. Training with Google Colab

Google Colaboratory is a cloud-based notebook environment which is said to have a very good training time for neural networks. We would like to train our model on Google Colab and compare it with the times taken to train the model on our computer's CPU and GPU, and add the results to the Table 2.

5.4. Testing with Additional Features

We would also want to try to add additional features to our images, such as a cap or goggles and try to recognize the person with our model. We also wanted to create images adding a beard, or blonde hair to our faces and check whether our trained model would be able to recognize the person.

References

[1] Face detection system based on viola - jones algorithm. *International Journal of Science and Research (IJSR)*, 5(4):62–64, 2016. 1

[2] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 3

[3] Rohit Thakur. Step by step vgg16 implementation in keras for beginners, Nov 2020. 2

[4] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2016. 3