Facial Recognition

Radhika Mattoo
rm3485@nyu.edu

Mohammad Afshar
ma2510@nyu.edu

December 22, 2016

1 Introduction

Our objective in this project was to achieve a relatively low error in face recognition by exploring and training variations of existing successful architectures. Our models were influenced specifically by Facebook and Google's *DeepFace* and *FaceNet* architectures, respectively, as well as the winning architecture from a 2011 Recognition Benchmark. We used a subset of Stefan Winkler's FaceScrub dataset to give us 41.6 thousand training images across 321 classes and tested our models using a subset of Labeled Faces in the Wild (LFW). Despite resource and time constraints, we created, trained, and tested three different architectures in order to compare and analyze the resulting rates in face recognition.

2 Problem Summary

The problem we aimed to solve using our models was facial recognition with 'same-not-same' classification. Thus, we trained a convolutional neural network on our dataset, and used Cosine similarity during testing to determine if two images passed through our trained network were of the same face.

3 Baseline Method

There are several experiments that have previously been done in facial recognition. Our benchmark for comparison was a deep learning model consisting of an ensemble method of over twenty neural nets. The training data used by this model is a subset of the LFW dataset using a generative

approach initially to generate more training samples. The misclassification error this experiment achieved was 91.75%.

4 Architectures

The overall pipeline of the model we used is based on that used in *DeepFace*. We first scrape the data from the given links and place them in class-based directories. The coordinates of the face is already given in every image, so we sample the image to only contain the face. We then use 3-D facial alignment to only consider the face, and down-sample to 70x70. Once this is done, we feed the images into our ConvNet for training. We trained on the entire training set, and once the network was trained, we cut off the last layer and used the network as a feature extractor for the LFW dataset. For each two pairs of same/not same images given, we found the cosine distance between the two and placed a threshold at 0.7. Although we could have further done training to determine the optimal threshold as well, we simply decided to use the results of the experiments done already. The non-linearity we used was ReLU and the loss function for all models is the CrossEntropy loss, which were both suggested in *DeepFace*. A batchsize of 128 is also consistent in our approach, with a learning rate of 0.01. In our experiments, we discovered that any change in the learning rate decreased performance. These parameters were also the same as the ones used in the DeepFace architecture.

4.1 Model 1

The architecture consists of 3 sets of convolutional layer, non-linearity layer, pooling layer, and dropout followed by one linear layer with 400 hidden units and then the output layer. The number of feature maps we used was based on a recognition architecture that used a high number of filters to detect several features in the faces. The number of feature maps goes from 100 with size 7x7 to 150 with size 4x4 to 250 with size 4x4 from each convolutional layer, with 2x2 Max Pooling all throughout. In the dropout layers, we dropout the weights based off a 0.2 probability.

4.2 Model 2

This architecture mimics the DeepFace network, with some modifications. Torch does not implement local convolutional layers, where each filter used is unique. Therefore, we replaced local convolutional layers with

regular convolutional layers. The network consists of a convolutional layer with 16 feature maps and filter size of 4x4, then a non-linear layer and a 2x2 max pooling layer. This is followed up by another convolutional layer with 32 feature maps and a 4x4 filter, followed by a non-linear layer and no max pooling. This is fed into a final convolutional layer with 128 feature maps and a filter size of 7x7, followed by a non-linear layer, 2x2 max pooling, and dropout. This is followed by a linear layer with 1000 hidden units and then an output layer.

4.3 Model 3

The final model we explored was inspired by a paper by Fei Fei Li. It consists of 2 sets of convolutions, non-linearity, convolution, non-linearity, and max pooling. We also used batch normalization to help with the convergence rate of SGD. The feature maps in the convolutional layers are as follows: 16, 32, 64, and 128 each with a filter size of 4x4. The two pooling layers are 2x2 max pooling. This is followed by a linear layer with 4096 hidden units followed by the output layer.

5 Obstacles

5.1 Data

Our largest obstacle with respect to data was simply finding enough viable images to use for training. While Facebook's DeepFace model was trained on a private dataset of 4.4 million images, for our training dataset we were left to our own devices. After spending much time searching online for a large dataset of colored face images, we discovered Stefan Winkler's FaceScrub dataset of 100 thousand celebrity images, each including labels and a download URL. While this dataset was the best and largest we came across, downloading these images, then aligning and cropping them proved to be too time-consuming and inefficient. To preserve time for actually training our different models, we decided to download 41.6 thousand images in total from FaceScrub. Thus, already hindered on one of the most important aspects of the project, we knew our results would be considerably subpar to those of DeepFace, FaceNet, and the Recognition Benchmark.

5.2 Resources and Time

In addition to lack of data, we encountered issues with our resources and time. Initially, our plan was to use NYU's High Performance Computing (HPC) clusters for our project, thus giving us the opportunity to implement, train, and test many different models. Unfortunately, over the past month, HPC has been unreliable in its service; it would either reject or queue our jobs for long periods of time. Thus, we were forced to use our own, significantly weaker, laptops to collect and clean our data, and train and test our models. Additionally, attempting to install qlua on different computers, such as our more powerful desktops with GPUs lead to the discovery that qlua's build was broken at some point over the past month. Finally, we planned on imitating Facebook's DeepFace architecture exactly, and train it with different parameters to see its effects on our results. Torch, however, does not offer locally connected convolutional layers, which is an integral and essential component to the architecture's success. Therefore, we replaced local convolutional layers with regular convolutional layers in our experiments. With more time to collect proper data and explore different libraries, such as TensorFlow for more diversity, and access to more reliable and available computing resources, we would have the opportunity to explore more models in much more depth.

6 Results

Mohammad

7 Conclusion

8 Future Works

Mohammad

9 References