Machine Learning
Term Paper

Facial Recognition Using Neural Networks

Raghav Mittal
Ashoka University
raghav.mittal_ug18@ashoka.edu.in

Shourya Shukla
Ashoka University
shourya.shukla_ug18@ashoka.edu.in

8th May, 2017

Abstract

The field of machine learning and neural networks has witnessed a paradigm shift in the past few decades. From simple human versus computer games to reading unconstrained handwritten strings, we have come a long way. In a world where almost everything, from handheld devices to supercomputers, bank account details to Facebook passwords and from the microphones attached to our laptops to the recent Oculus Rift, is vulnerable to cyber attacks, one thing that holds exclusive and fundamental to everyone is their face. One could argue that a person's fingerprint, retina scan or heartbeat is relatively distinctive than one's face, however, due to technological and economical barriers, most people do not have access to fingerprint or retina scanners. Given such a scenario, a face recogniser could do wonders to the mess regarding authentication. A well trained face recogniser could help us mark attendance for lectures and void the system from proxies. A strong database could further help put name to damaged bodies with minimal to no documentation (say identity cards or a driving licence). One could it use for financial transactions at a grocery store and help make our economy go cashless. This paper wishes to describe the problem of recognising people from their facial images and thereby provide an algorithm which can be deployed for, yet not limited to, the purpose of authentication.

Introduction

Face recognition is not child's play. Human beings, by far, outperform even the most advanced machines at both quality and time efficiency. It is prudent to realise that a neither does a face have rigid structure nor does it have a rigid orientation. The images in our collection set are likely to be different from what a webcam might capture. A person, for instance, could have tilted her face in either of the directions, both vertically and horizontally. This makes the task of a machine, a non conscious entity, extremely complex. In our journey of educating machines and making them capable of learning, we train and test our software on various datasets. To demonstrate an accurate working model of the application and implement it on our immediate environment, that is, on our companions at Ashoka, we feature people from various batches, cultural and topographical backgrounds and further, people performing non-academic backgrounds. Bearing the comfort zone of students and the lack of confidentiality that our project submission entails, it was difficult for us to incorporate everyone for our dataset.

Database

Our software was primarily trained on the dataset: Face Recognition Data- University of Essex, UK.
The database comprises of students (mainly undergraduates, also consists of older people) from different races and gender orientations. A decent number of people in our database wore glasses and maintained beards. (Spacek, 2007).

A Detailed Dataset Description

Technical Aspects

1) The pictures were clicked used a stationary camera.
2) All participants made of a contribution of twenty images apiece.
3) Contributors were obliged to move a step forward towards the lens in order to produce head variations.
4) We witness a time lag of one half of a second between sequential frames.
5) 180 * 200 resolution pictures.

Variations

1) Red curtains are used for background. Shadow variations help facilitate background changes.
2) Significant head variations by scaling and other techniques which entail turns, tilts and slants along with other minor variations.
3) Translational effects produced in images.
4) Lighting variations caused by movement and artificial lightings.
5) Minor expression variations.
6) Negligible hair variations for the images were taken in a single session.

Some sample images:

Our secondary test dataset, comprising of students at Ashoka (The Ashokan Dtabase), entails the following features:

1) Taken next to a white wall to maintain stark contrast with the contributor's face and other bodily features.
2) Facial variations include turning, tilting and other mechanisms in form of expressions.
3) Our database includes people adhering to different ethnicities and genders.
4) The average image size is 3MB and 4096 * 2304 pixels (taken by a hand held Lenovo K-3 note to maintain non-uniformity and thereby make our model more adaptable).
5) People in our datasets incorporates people with varying beard designs and eye colours. Further, it includes people with spectacles and distinctive attire.

Basic Model

Based on our learnings from online sources and work related to face recognizers from men and women of distinguished scientific backgrounds, we implemented the following model post a few hit and trials after resizing the images from (300,200) to (200,180).:

The Initial Model:

```
model = Sequential()
model.add(Conv2D(32, (5, 5), padding='valid', input_shape=(200, 180, 3),activation='relu'))
model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(72, activation='relu'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=10, batch_size=32)
```

On running this model, we noticed a lot of fluctuation both in the loss and the accuracy, thus arriving at the conclusion that the training data has too many local minimas. Therefore we reduced the learning rate considerably. We also realized that the number of feature maps in our convolutional layers was constant, which in itself makes it a faulty model.

Based on the above learnings from implementation of previous algorithms studied in-depth by this paper, we incorporate the following changes to our design:

1) Our model now comprised of four convolution layers with feature maps 3, 16, 32 and 64 respectively.
2) We changed our learning rate from 0.001, that is, the default value for an Adam optimizer, to 0.00001 in order to decrease the rate of learning and thereby try and avoid local minimas.

The Changed Model:

```
model = Sequential()
model.add(Conv2D(3, (2, 2), padding='same',kernel_initializer = 'normal', input_shape=(200, 180, 3),activation='elu'))
#model.add(Dropout(0.3))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(Dropout(0.3))
#model.add(MaxPooling2D(pool_size=(2, 2)))
#model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer = 'normal'))
model.add(Dense(6, activation='relu', kernel_initializer = 'normal'))
sgd = optimizers.Adam(lr=0.00001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=15, batch_size=32)
```

We then tried training our model for 100 epochs:

```
Using Theano backend.
Train on 1008 samples, validate on 432 samples
Epoch 1/100
1008/1008 [==============================] - 208s - loss: 9.5126 - acc: 0.0109 - val_loss: 8.5531 - val_acc: 0.0093
Epoch 2/100
1008/1008 [==============================] - 210s - loss: 8.2441 - acc: 0.0198 - val_loss: 7.7205 - val_acc: 0.0139
Epoch 3/100
1008/1008 [==============================] - 193s - loss: 7.5095 - acc: 0.0149 - val_loss: 6.6901 - val_acc: 0.0093
Epoch 4/100
1008/1008 [==============================] - 192s - loss: 6.8411 - acc: 0.0129 - val_loss: 6.0332 - val_acc: 0.0139
Epoch 5/100
1008/1008 [==============================] - 191s - loss: 6.3815 - acc: 0.0208 - val_loss: 5.7350 - val_acc: 0.0162
Epoch 6/100
1008/1008 [==============================] - 194s - loss: 6.0538 - acc: 0.0208 - val_loss: 5.9092 - val_acc: 0.0301
Epoch 7/100
1008/1008 [==============================] - 195s - loss: 6.2376 - acc: 0.0208 - val_loss: 5.6830 - val_acc: 0.0324
Epoch 8/100
1008/1008 [==============================] - 194s - loss: 5.7931 - acc: 0.0218 - val_loss: 5.5952 - val_acc: 0.0347
Epoch 9/100
1008/1008 [==============================] - 193s - loss: 5.6117 - acc: 0.0218 - val_loss: 5.6015 - val_acc: 0.0347
Epoch 10/100
1008/1008 [==============================] - 193s - loss: 5.5125 - acc: 0.0218 - val_loss: 5.5394 - val_acc: 0.0255
Epoch 11/100
1008/1008 [==============================] - 193s - loss: 5.4701 - acc: 0.0288 - val_loss: 5.5107 - val_acc: 0.0185
Epoch 12/100
1008/1008 [==============================] - 194s - loss: 5.3764 - acc: 0.0268 - val_loss: 5.3281 - val_acc: 0.0255
Epoch 13/100
1008/1008 [==============================] - 195s - loss: 5.3190 - acc: 0.0268 - val_loss: 5.2616 - val_acc: 0.0231
Epoch 14/100
1008/1008 [==============================] - 195s - loss: 5.2236 - acc: 0.0268 - val_loss: 5.2548 - val_acc: 0.0255
Epoch 15/100
1008/1008 [==============================] - 194s - loss: 5.2127 - acc: 0.0258 - val_loss: 5.2135 - val_acc: 0.0208
Epoch 16/100
1008/1008 [==============================] - 194s - loss: 5.1516 - acc: 0.0317 - val_loss: 5.1964 - val_acc: 0.0324
Epoch 17/100
1008/1008 [==============================] - 193s - loss: 5.1173 - acc: 0.0357 - val_loss: 5.1771 - val_acc: 0.0347
Epoch 18/100
1008/1008 [==============================] - 194s - loss: 5.1133 - acc: 0.0466 - val_loss: 5.1565 - val_acc: 0.0324
Epoch 19/100
1008/1008 [==============================] - 199s - loss: 5.0964 - acc: 0.0308 - val_loss: 5.1358 - val_acc: 0.0347
Epoch 20/100
1008/1008 [==============================] - 202s - loss: 5.0235 - acc: 0.0337 - val_loss: 4.9411 - val_acc: 0.0440
Epoch 21/100
1008/1008 [==============================] - 201s - loss: 4.8603 - acc: 0.0446 - val_loss: 4.9282 - val_acc: 0.0463
Epoch 22/100
1008/1008 [==============================] - 201s - loss: 4.8401 - acc: 0.0486 - val_loss: 4.8299 - val_acc: 0.0440
Epoch 23/100
1008/1008 [==============================] - 199s - loss: 4.7319 - acc: 0.0456 - val_loss: 4.7447 - val_acc: 0.0463
Epoch 24/100
1008/1008 [==============================] - 199s - loss: 4.6857 - acc: 0.0595 - val_loss: 4.7266 - val_acc: 0.0579
Epoch 25/100
1008/1008 [==============================] - 199s - loss: 4.6674 - acc: 0.0635 - val_loss: 4.6948 - val_acc: 0.0579
Epoch 26/100
1008/1008 [==============================] - 200s - loss: 4.5993 - acc: 0.0516 - val_loss: 4.6603 - val_acc: 0.0509
Epoch 27/100
1008/1008 [==============================] - 199s - loss: 4.5761 - acc: 0.0704 - val_loss: 4.6445 - val_acc: 0.0602
Epoch 28/100
1008/1008 [==============================] - 199s - loss: 4.5859 - acc: 0.0734 - val_loss: 4.6067 - val_acc: 0.0648
Epoch 29/100
1008/1008 [==============================] - 199s - loss: 4.5258 - acc: 0.0933 - val_loss: 4.5634 - val_acc: 0.0833
Epoch 30/100
1008/1008 [==============================] - 198s - loss: 4.4877 - acc: 0.0913 - val_loss: 4.5631 - val_acc: 0.1042
Epoch 31/100
1008/1008 [==============================] - 196s - loss: 4.4021 - acc: 0.1200 - val_loss: 4.4812 - val_acc: 0.1227
Epoch 32/100
1008/1008 [==============================] - 197s - loss: 4.4457 - acc: 0.1240 - val_loss: 4.4136 - val_acc: 0.1435
Epoch 33/100
1008/1008 [==============================] - 196s - loss: 4.4462 - acc: 0.1379 - val_loss: 4.9165 - val_acc: 0.0347
Epoch 34/100
1008/1008 [==============================] - 199s - loss: 4.8112 - acc: 0.0476 - val_loss: 4.8248 - val_acc: 0.0394
Epoch 35/100
1008/1008 [==============================] - 197s - loss: 4.5439 - acc: 0.0565 - val_loss: 4.5775 - val_acc: 0.0440
Epoch 36/100
1008/1008 [==============================] - 196s - loss: 4.3899 - acc: 0.0694 - val_loss: 4.5033 - val_acc: 0.0602
```

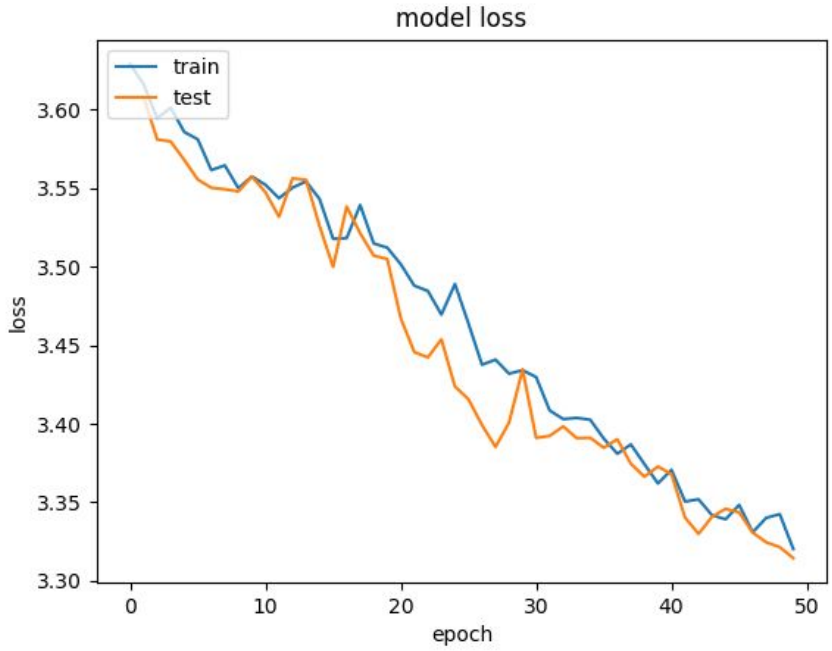
```
Epoch 34/100
1008/1008 [==============================] - 199s - loss: 4.8112 - acc: 0.0476 - val_loss: 4.8248 - val_acc: 0.0394
Epoch 35/100
1008/1008 [==============================] - 197s - loss: 4.5439 - acc: 0.0565 - val_loss: 4.5775 - val_acc: 0.0440
Epoch 36/100
1008/1008 [==============================] - 196s - loss: 4.3899 - acc: 0.0694 - val_loss: 4.5033 - val_acc: 0.0602
Epoch 37/100
1008/1008 [==============================] - 197s - loss: 4.3797 - acc: 0.1071 - val_loss: 4.5283 - val_acc: 0.1157
Epoch 38/100
1008/1008 [==============================] - 190s - loss: 4.2920 - acc: 0.1587 - val_loss: 4.4795 - val_acc: 0.1366
Epoch 39/100
1008/1008 [==============================] - 191s - loss: 4.2256 - acc: 0.1845 - val_loss: 4.2700 - val_acc: 0.1921
Epoch 40/100
1008/1008 [==============================] - 190s - loss: 4.2366 - acc: 0.1796 - val_loss: 4.2178 - val_acc: 0.2708
Epoch 41/100
1008/1008 [==============================] - 190s - loss: 4.0083 - acc: 0.2381 - val_loss: 4.0438 - val_acc: 0.2847
Epoch 42/100
1008/1008 [==============================] - 190s - loss: 4.0798 - acc: 0.2500 - val_loss: 4.5947 - val_acc: 0.0602
Epoch 43/100
1008/1008 [==============================] - 191s - loss: 4.5768 - acc: 0.0327 - val_loss: 4.7325 - val_acc: 0.0278
Epoch 44/100
1008/1008 [==============================] - 191s - loss: 4.2892 - acc: 0.0377 - val_loss: 4.5539 - val_acc: 0.0301
Epoch 45/100
1008/1008 [==============================] - 190s - loss: 4.2115 - acc: 0.0476 - val_loss: 4.6289 - val_acc: 0.0324
Epoch 46/100
1008/1008 [==============================] - 191s - loss: 4.4176 - acc: 0.0367 - val_loss: 4.4796 - val_acc: 0.0324
Epoch 47/100
1008/1008 [==============================] - 190s - loss: 4.2495 - acc: 0.0417 - val_loss: 4.3853 - val_acc: 0.0440
Epoch 48/100
1008/1008 [==============================] - 190s - loss: 4.1033 - acc: 0.0714 - val_loss: 4.3186 - val_acc: 0.0718
Epoch 49/100
1008/1008 [==============================] - 189s - loss: 4.0253 - acc: 0.1151 - val_loss: 4.2080 - val_acc: 0.1227
Epoch 50/100
1008/1008 [==============================] - 189s - loss: 3.8743 - acc: 0.2460 - val_loss: 4.1509 - val_acc: 0.3727
Epoch 51/100
1008/1008 [==============================] - 190s - loss: 3.8335 - acc: 0.3502 - val_loss: 4.0923 - val_acc: 0.3796
Epoch 52/100
1008/1008 [==============================] - 190s - loss: 3.6423 - acc: 0.3740 - val_loss: 3.9995 - val_acc: 0.4051
Epoch 53/100
1008/1008 [==============================] - 190s - loss: 3.5604 - acc: 0.4137 - val_loss: 3.9227 - val_acc: 0.4514
Epoch 54/100
1008/1008 [==============================] - 191s - loss: 3.5738 - acc: 0.4196 - val_loss: 3.7961 - val_acc: 0.4213
Epoch 55/100
1008/1008 [==============================] - 191s - loss: 3.3500 - acc: 0.4395 - val_loss: 4.0114 - val_acc: 0.4167
Epoch 56/100
1008/1008 [==============================] - 190s - loss: 3.5145 - acc: 0.4415 - val_loss: 3.7967 - val_acc: 0.5370
Epoch 57/100
1008/1008 [==============================] - 191s - loss: 3.4055 - acc: 0.3214 - val_loss: 4.1506 - val_acc: 0.1134
Epoch 58/100
1008/1008 [==============================] - 191s - loss: 3.4144 - acc: 0.3026 - val_loss: 3.9272 - val_acc: 0.3657
Epoch 59/100
1008/1008 [==============================] - 192s - loss: 3.3466 - acc: 0.4633 - val_loss: 3.9000 - val_acc: 0.3704
Epoch 60/100
1008/1008 [==============================] - 192s - loss: 3.8102 - acc: 0.1399 - val_loss: 4.2332 - val_acc: 0.0417
Epoch 61/100
1008/1008 [==============================] - 192s - loss: 3.3507 - acc: 0.2014 - val_loss: 3.9036 - val_acc: 0.1968
Epoch 62/100
1008/1008 [==============================] - 193s - loss: 3.4182 - acc: 0.2758 - val_loss: 4.7590 - val_acc: 0.0116
Epoch 63/100
1008/1008 [==============================] - 193s - loss: 3.8665 - acc: 0.0288 - val_loss: 4.0694 - val_acc: 0.0231
Epoch 64/100
1008/1008 [==============================] - 194s - loss: 3.4596 - acc: 0.1220 - val_loss: 4.0541 - val_acc: 0.1157
Epoch 65/100
1008/1008 [==============================] - 195s - loss: 3.2368 - acc: 0.2966 - val_loss: 3.9509 - val_acc: 0.2824
Epoch 66/100
1008/1008 [==============================] - 195s - loss: 3.0336 - acc: 0.4702 - val_loss: 3.6999 - val_acc: 0.5139
Epoch 67/100
1008/1008 [==============================] - 195s - loss: 3.0383 - acc: 0.5526 - val_loss: 3.8523 - val_acc: 0.5741
Epoch 68/100
1008/1008 [==============================] - 196s - loss: nan - acc: 0.5833 - val_loss: nan - val_acc: 0.0139
Epoch 69/100
1008/1008 [==============================] - 184s - loss: nan - acc: 0.0139 - val_loss: nan - val_acc: 0.0139
Epoch 70/100
 128/1008 [==>...........................] - ETA: 140s - loss: nan - acc: 0.0078
```

The model showed a gradual but considerable improvement in the accuracy and reduction in loss, but gave unexpected results once it reached the 68th epoch.

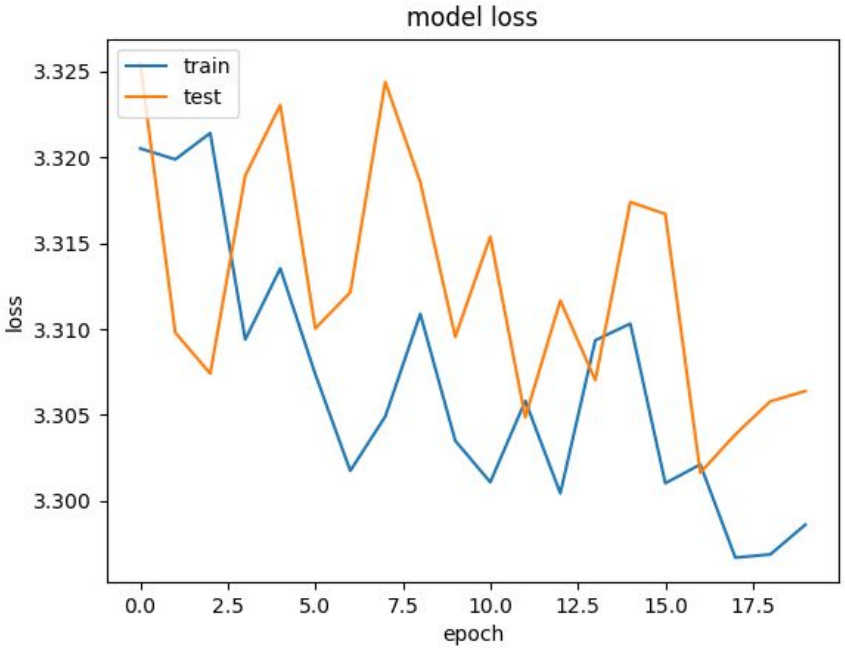
We tried changing our activation function from relu to sigmoid and further added a softmax activation function. Following was the outcome-

```
Epoch 21/50
518/518 [==============================] - 283s - loss: 3.5015 - acc: 0.1486 - val_loss: 3.4671 - val_acc: 0.1847
Epoch 22/50
518/518 [==============================] - 280s - loss: 3.4879 - acc: 0.1680 - val_loss: 3.4456 - val_acc: 0.2117
Epoch 23/50
518/518 [==============================] - 282s - loss: 3.4845 - acc: 0.1680 - val_loss: 3.4422 - val_acc: 0.2072
Epoch 24/50
518/518 [==============================] - 283s - loss: 3.4695 - acc: 0.1853 - val_loss: 3.4537 - val_acc: 0.1937
Epoch 25/50
518/518 [==============================] - 279s - loss: 3.4889 - acc: 0.1680 - val_loss: 3.4237 - val_acc: 0.2297
Epoch 26/50
518/518 [==============================] - 276s - loss: 3.4638 - acc: 0.1892 - val_loss: 3.4156 - val_acc: 0.2387
Epoch 27/50
518/518 [==============================] - 277s - loss: 3.4376 - acc: 0.2162 - val_loss: 3.3992 - val_acc: 0.2568
Epoch 28/50
518/518 [==============================] - 277s - loss: 3.4408 - acc: 0.2124 - val_loss: 3.3852 - val_acc: 0.2703
Epoch 29/50
518/518 [==============================] - 276s - loss: 3.4319 - acc: 0.2239 - val_loss: 3.4008 - val_acc: 0.2523
Epoch 30/50
518/518 [==============================] - 278s - loss: 3.4340 - acc: 0.2239 - val_loss: 3.4346 - val_acc: 0.2162
Epoch 31/50
518/518 [==============================] - 278s - loss: 3.4297 - acc: 0.2239 - val_loss: 3.3910 - val_acc: 0.2658
Epoch 32/50
518/518 [==============================] - 277s - loss: 3.4084 - acc: 0.2452 - val_loss: 3.3922 - val_acc: 0.2613
Epoch 33/50
518/518 [==============================] - 278s - loss: 3.4030 - acc: 0.2548 - val_loss: 3.3983 - val_acc: 0.2613
Epoch 34/50
518/518 [==============================] - 279s - loss: 3.4037 - acc: 0.2529 - val_loss: 3.3908 - val_acc: 0.2658
Epoch 35/50
518/518 [==============================] - 279s - loss: 3.4025 - acc: 0.2490 - val_loss: 3.3910 - val_acc: 0.2658
Epoch 36/50
518/518 [==============================] - 279s - loss: 3.3905 - acc: 0.2625 - val_loss: 3.3847 - val_acc: 0.2748
Epoch 37/50
518/518 [==============================] - 280s - loss: 3.3809 - acc: 0.2761 - val_loss: 3.3901 - val_acc: 0.2613
Epoch 38/50
518/518 [==============================] - 279s - loss: 3.3868 - acc: 0.2683 - val_loss: 3.3745 - val_acc: 0.2838
Epoch 39/50
518/518 [==============================] - 280s - loss: 3.3743 - acc: 0.2799 - val_loss: 3.3663 - val_acc: 0.2883
Epoch 40/50
518/518 [==============================] - 280s - loss: 3.3620 - acc: 0.2934 - val_loss: 3.3727 - val_acc: 0.2793
Epoch 41/50
518/518 [==============================] - 280s - loss: 3.3707 - acc: 0.2838 - val_loss: 3.3679 - val_acc: 0.2883
Epoch 42/50
518/518 [==============================] - 290s - loss: 3.3505 - acc: 0.3069 - val_loss: 3.3403 - val_acc: 0.3153
Epoch 43/50
518/518 [==============================] - 286s - loss: 3.3519 - acc: 0.3031 - val_loss: 3.3299 - val_acc: 0.3288
Epoch 44/50
518/518 [==============================] - 281s - loss: 3.3417 - acc: 0.3127 - val_loss: 3.3408 - val_acc: 0.3153
Epoch 45/50
518/518 [==============================] - 279s - loss: 3.3392 - acc: 0.3127 - val_loss: 3.3458 - val_acc: 0.3063
Epoch 46/50
518/518 [==============================] - 280s - loss: 3.3483 - acc: 0.3069 - val_loss: 3.3435 - val_acc: 0.3108
Epoch 47/50
518/518 [==============================] - 279s - loss: 3.3309 - acc: 0.3243 - val_loss: 3.3307 - val_acc: 0.3198
Epoch 48/50
518/518 [==============================] - 280s - loss: 3.3401 - acc: 0.3147 - val_loss: 3.3245 - val_acc: 0.3333
Epoch 49/50
518/518 [==============================] - 280s - loss: 3.3424 - acc: 0.3108 - val_loss: 3.3214 - val_acc: 0.3333
Epoch 50/50
518/518 [==============================] - 281s - loss: 3.3203 - acc: 0.3359 - val_loss: 3.3145 - val_acc: 0.3423
[[ 0.02583161  0.02583161  0.02583161  0.02583161  0.02583161  0.02583161
   0.02583161  0.02583161  0.02583161  0.02583161  0.02592253  0.02583161
   0.02583161  0.02583161  0.02583161  0.06997095  0.02583161  0.02583161
   0.02583175  0.02583161  0.02583161  0.02583161  0.02583161  0.02583161
   0.02583161  0.02583161  0.02583161  0.02583161  0.02583161  0.02583161
   0.02583161  0.02583161  0.02583161  0.02583161  0.02583161  0.02583161
   0.02583161]]
```

Although our model did not predict the accurate result, we see a gradual decrease in our loss function, which was desirable.



However,when run for 20 more epochs, the loss function was very unexpected:



The Final Model

Based on the above learnings from implementation of previous algorithms studied in-depth by this paper, we incorporate the following changes to our design:

1) In order to enhance accuracy (quicker), we increased our learning rate from 0.00001 to 0.0001.
2) We resized our images in the primary Face Recognition Dataset from (300,200) to (150,150).
3) This was proceeded by normalization for a meticulous yield of outputs.
4) The number of convolution layers was of brought down from four to two, with the current two layers having 3 and 16 feature maps respectively.
5) From two dropouts of size 0.3 each, our final model comprised of a single dropout of size 0.3.

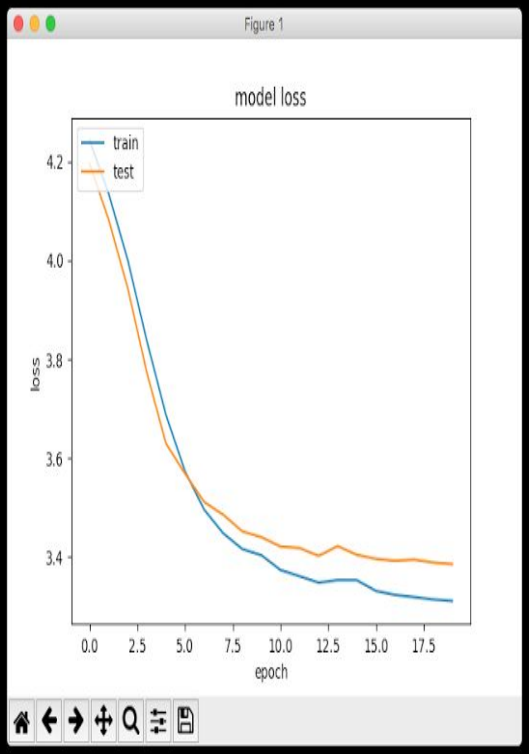
Making alterations to our output methodologies, our final model now returns the name of the person instead of the probabilities and arrays.

Activation Function used- Softmax is commonly used under log loss(or cross-entropy) regime, giving a non-linear variant of multinomial logistic regression. Thus it is ideal for our model as it can take inputs from categorical cross entropy and convert it into digits between 0 and 1, thus giving a probability distribution of the output classes.

Optimizer used- Adam uses moving averages of the parameters (momentum). This enables Adam to use a larger effective step size, and the algorithm will converge to this step size quickly, and the algorithm will converge to this step size without fine tuning.. With the initial models, and with different optimizers, the loss was decreasing at a very slow rate. Therefore we decided to use Adam make the loss function steeper, and to converge to the minima quicker.

We put our model to test for 20 epochs-

```
  File "/Users/mlproject/anaconda/lib/python3.6/site-packages/scipy/misc/pilutil.py", line 478, in imresize
    im = toimage(arr, mode=mode)
  File "/Users/mlproject/anaconda/lib/python3.6/site-packages/scipy/misc/pilutil.py", line 287, in toimage
    raise ValueError("'arr' does not have a suitable array shape for "
ValueError: 'arr' does not have a suitable array shape for any mode.
LABPC26s-iMac:desktop mlproject$ python face.py
Using Theano backend.
[ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
Train on 154 samples, validate on 66 samples
Epoch 1/20
154/154 [==============================] - 6s - loss: 2.3898 - acc: 0.1169 - val_loss: 2.3853 - val_acc: 0.1061
Epoch 2/20
154/154 [==============================] - 6s - loss: 2.3387 - acc: 0.1299 - val_loss: 2.3618 - val_acc: 0.0152
Epoch 3/20
 64/154 [===========>..................] - ETA: 3s - loss: 2.3381 - acc: 0.1719^Z
[1]+  Stopped                 python face.py
LABPC26s-iMac:desktop mlproject$ python face.py
Using Theano backend.
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
Train on 994 samples, validate on 426 samples
Epoch 1/20
994/994 [==============================] - 42s - loss: 4.2427 - acc: 0.0221 - val_loss: 4.1956 - val_acc: 0.0188
Epoch 2/20
994/994 [==============================] - 41s - loss: 4.1349 - acc: 0.0473 - val_loss: 4.0822 - val_acc: 0.0540
Epoch 3/20
994/994 [==============================] - 41s - loss: 4.0006 - acc: 0.0795 - val_loss: 3.9436 - val_acc: 0.0845
Epoch 4/20
994/994 [==============================] - 41s - loss: 3.8370 - acc: 0.1066 - val_loss: 3.7729 - val_acc: 0.1925
Epoch 5/20
994/994 [==============================] - 41s - loss: 3.6876 - acc: 0.3964 - val_loss: 3.6299 - val_acc: 0.5681
Epoch 6/20
994/994 [==============================] - 42s - loss: 3.5734 - acc: 0.4718 - val_loss: 3.5688 - val_acc: 0.6150
Epoch 7/20
994/994 [==============================] - 41s - loss: 3.4961 - acc: 0.6630 - val_loss: 3.5116 - val_acc: 0.7512
Epoch 8/20
994/994 [==============================] - 41s - loss: 3.4481 - acc: 0.7535 - val_loss: 3.4858 - val_acc: 0.7277
Epoch 9/20
994/994 [==============================] - 41s - loss: 3.4164 - acc: 0.8229 - val_loss: 3.4521 - val_acc: 0.8075
Epoch 10/20
994/994 [==============================] - 41s - loss: 3.4038 - acc: 0.7062 - val_loss: 3.4403 - val_acc: 0.8308
Epoch 11/20
994/994 [==============================] - 41s - loss: 3.3740 - acc: 0.8803 - val_loss: 3.4214 - val_acc: 0.8521
Epoch 12/20
994/994 [==============================] - 41s - loss: 3.3613 - acc: 0.8793 - val_loss: 3.4183 - val_acc: 0.8521
Epoch 13/20
994/994 [==============================] - 41s - loss: 3.3483 - acc: 0.9266 - val_loss: 3.4025 - val_acc: 0.8615
Epoch 14/20
994/994 [==============================] - 41s - loss: 3.3534 - acc: 0.7535 - val_loss: 3.4222 - val_acc: 0.8005
Epoch 15/20
994/994 [==============================] - 41s - loss: 3.3532 - acc: 0.7726 - val_loss: 3.4046 - val_acc: 0.8239
Epoch 16/20
994/994 [==============================] - 41s - loss: 3.3316 - acc: 0.9215 - val_loss: 3.3963 - val_acc: 0.8897
Epoch 17/20
994/994 [==============================] - 41s - loss: 3.3235 - acc: 0.9577 - val_loss: 3.3926 - val_acc: 0.8991
Epoch 18/20
994/994 [==============================] - 41s - loss: 3.3191 - acc: 0.9698 - val_loss: 3.3949 - val_acc: 0.8850
Epoch 19/20
994/994 [==============================] - 43s - loss: 3.3143 - acc: 0.9638 - val_loss: 3.3889 - val_acc: 0.8944
Epoch 20/20
994/994 [==============================] - 48s - loss: 3.3114 - acc: 0.9738 - val_loss: 3.3857 - val_acc: 0.9178
[[ 0.01340076  0.01339059  0.01339058  0.01340076  0.01339058  0.01339058
   0.01339058  0.01339058  0.01339058  0.01339058  0.01339458  0.01339058
   0.01339061  0.01339058  0.01339058  0.01339094  0.01341464  0.0134309
   0.01339058  0.01339058  0.01339059  0.01339058  0.01353027  0.01339058
   0.01339058  0.01339326  0.01372001  0.01339058  0.01339641  0.01339081
   0.01339059  0.01339059  0.01339059  0.01340964  0.01339059  0.01339058
   0.01339058  0.01339166  0.01339058  0.01339058  0.01614249  0.0133935
   0.0133906   0.0134032   0.01339058  0.01339059  0.01347298  0.01339058
   0.01341451  0.01339058  0.01343264  0.03639732  0.01339076  0.01339058
   0.01339058  0.03554422  0.01339059  0.01339058  0.01339096  0.01341544
   0.01360737  0.01339058  0.01347334  0.01339058  0.01348855  0.0133906
   0.01339058  0.01353278  0.01339064  0.0133906   0.01339059]]
```



As is visible, the loss function is remarkable, with accuracy as high as **97.38%**.

Here are a few sample successful predictions from our model:

```
Using Theano backend.
[ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[[ 0.01373457  0.01378289  0.01374364  0.0137326   0.0137326   0.0138994
   0.03732203  0.01373276  0.01373458  0.0137326   0.0137326   0.0137326
   0.0137326   0.01373262  0.0137326   0.0137326   0.0137326   0.0137326
   0.0137326   0.01373261  0.0137326   0.01373611  0.01373272  0.01373322
   0.01373639  0.01373476  0.0137326   0.01373272  0.01373268  0.01373667
   0.0137326   0.01373264  0.0137425   0.0137326   0.01373261  0.01373261
   0.01373908  0.01373268  0.01373283  0.01373264  0.01374137  0.01373269
   0.01373281  0.0137326   0.01373484  0.01382999  0.0137326   0.01373393
   0.0137326   0.01373928  0.0137326   0.0137326   0.01373334  0.01439457
   0.01381808  0.01374202  0.0137326   0.0137326   0.0137326   0.01373272
   0.01382696  0.0137326   0.0137326   0.0137326   0.01373269  0.01373265
   0.01389612  0.01373261  0.0137326   0.01373265  0.0137326 ]]
```

```
LABPC26s-iMac:desktop mlproject$ python face.py
Using Theano backend.
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[[ 0.01518053  0.01357142  0.01357199  0.01362939  0.01357142  0.01357143
   0.01357142  0.01357143  0.01357142  0.01357142  0.01394386  0.01357142
   0.01357142  0.01357142  0.01357142  0.01357145  0.01357143  0.01357232
   0.01357142  0.01357205  0.01357225  0.01357142  0.01357195  0.01357142
   0.01357146  0.01357166  0.01357365  0.01357142  0.01357755  0.01360738
   0.0135715   0.01463093  0.01358126  0.01357192  0.01357142  0.01357142
   0.01357142  0.01357167  0.01357153  0.01357142  0.0135746   0.01357225
   0.01357154  0.01358945  0.01357142  0.01357195  0.01370774  0.01357142
   0.01357153  0.01357182  0.01357142  0.01373388  0.01357142  0.01357142
   0.01357142  0.02312394  0.01357143  0.01357142  0.01357142  0.01357158
   0.03687887  0.01357143  0.01357151  0.01357142  0.01357171  0.01357142
   0.01357146  0.01357277  0.01357143  0.01357142  0.01365882]]
```

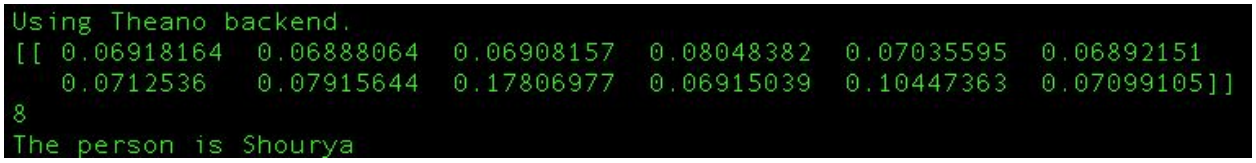
Towards the final stages of our project, when subject to test, our model bore exceptional outcomes. Taking a leap of faith, in a rather ambitious manner, we ran the algorithm for our newly formed dataset (Ashokan database) and fed to it test images which our application had not borne witness to before. The following images encapture our experience with our primary Machine Learning development:

An external new image of Mr. Koishore Roy was fed to the model. The dataset for Koishore's images was 4th in the overall dataset.

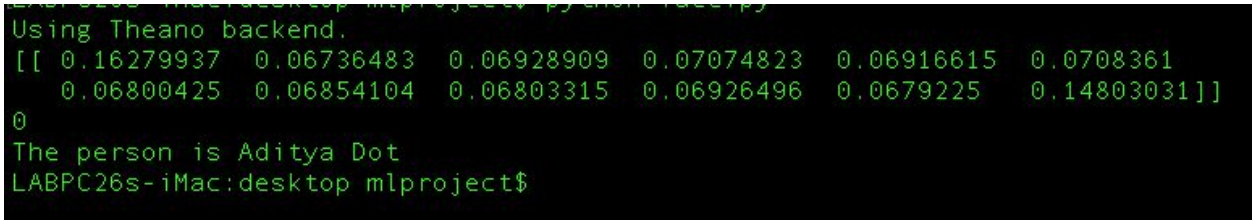
```
Using Theano backend.
Train on 66 samples, validate on 29 samples
Epoch 1/20
66/66 [==============================] - 3s - loss: 2.4971 - acc: 0.1364 - val_loss: 2.4993 - val_acc: 0.1034
Epoch 2/20
66/66 [==============================] - 3s - loss: 2.4794 - acc: 0.0909 - val_loss: 2.4864 - val_acc: 0.1034
Epoch 3/20
66/66 [==============================] - 3s - loss: 2.4542 - acc: 0.0909 - val_loss: 2.4741 - val_acc: 0.0345
Epoch 4/20
66/66 [==============================] - 3s - loss: 2.4494 - acc: 0.1212 - val_loss: 2.4634 - val_acc: 0.0345
Epoch 5/20
66/66 [==============================] - 3s - loss: 2.4392 - acc: 0.1061 - val_loss: 2.4674 - val_acc: 0.0345
Epoch 6/20
66/66 [==============================] - 3s - loss: 2.4011 - acc: 0.1061 - val_loss: 2.4759 - val_acc: 0.0690
Epoch 7/20
66/66 [==============================] - 3s - loss: 2.4229 - acc: 0.1667 - val_loss: 2.4812 - val_acc: 0.1379
Epoch 8/20
66/66 [==============================] - 3s - loss: 2.3734 - acc: 0.2879 - val_loss: 2.4706 - val_acc: 0.1379
Epoch 9/20
66/66 [==============================] - 3s - loss: 2.3722 - acc: 0.1818 - val_loss: 2.4610 - val_acc: 0.1034
Epoch 10/20
66/66 [==============================] - 3s - loss: 2.3469 - acc: 0.1818 - val_loss: 2.4646 - val_acc: 0.1034
Epoch 11/20
66/66 [==============================] - 3s - loss: 2.3447 - acc: 0.1818 - val_loss: 2.4585 - val_acc: 0.1034
Epoch 12/20
66/66 [==============================] - 3s - loss: 2.3270 - acc: 0.2121 - val_loss: 2.4313 - val_acc: 0.1034
Epoch 13/20
66/66 [==============================] - 3s - loss: 2.3061 - acc: 0.1667 - val_loss: 2.4036 - val_acc: 0.1034
Epoch 14/20
66/66 [==============================] - 3s - loss: 2.2947 - acc: 0.2273 - val_loss: 2.3680 - val_acc: 0.1034
Epoch 15/20
66/66 [==============================] - 3s - loss: 2.2588 - acc: 0.3333 - val_loss: 2.3274 - val_acc: 0.3103
Epoch 16/20
66/66 [==============================] - 3s - loss: 2.2335 - acc: 0.4545 - val_loss: 2.3023 - val_acc: 0.4483
Epoch 17/20
66/66 [==============================] - 3s - loss: 2.1986 - acc: 0.5455 - val_loss: 2.2600 - val_acc: 0.5172
Epoch 18/20
66/66 [==============================] - 3s - loss: 2.1693 - acc: 0.6061 - val_loss: 2.2279 - val_acc: 0.5517
Epoch 19/20
66/66 [==============================] - 3s - loss: 2.1629 - acc: 0.6515 - val_loss: 2.2166 - val_acc: 0.3793
Epoch 20/20
66/66 [==============================] - 3s - loss: 2.1539 - acc: 0.5909 - val_loss: 2.1949 - val_acc: 0.3103
[[ 0.10762095  0.07612409  0.06616759  0.14456487  0.08672705  0.06607223
   0.09379759  0.0781045   0.0871208   0.06312443  0.05988421  0.07069168]]
```

And therefore, as is visible, our model correctly predicted Koishore's face. (Assigned the highest probability to the 4th class).

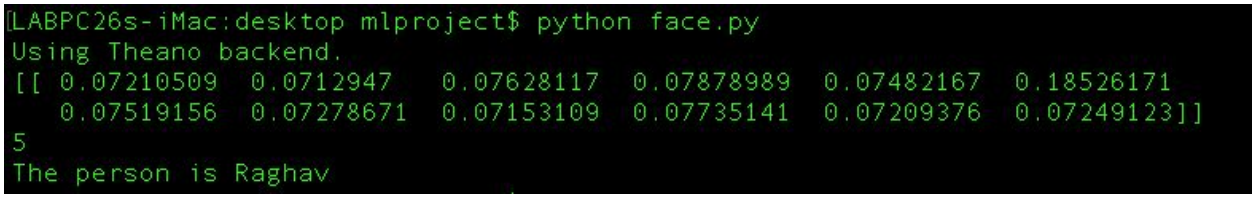
Post modifying the code, our model held the capacity to directly predict the name of the person as in the following situations:



Predicts ‘Shourya’ correctly



Predicts ‘Aditya Dot’ correctly



Predicts ‘Raghav’ correctly

The output predicted by these models **were accurate for all the given samples.**

Conclusion

Our purpose of building a Deep Face Recogniser was a success. In our journey we explored and critically analysed some of the most common and well known

Declaration

We hereby declare, that whilst the workload for this project may have been distributed equally and through mutual consent for sake of efficiency and time management, the two of us hold individual capacities to interpret and decode the assignment in its entirety. Further, we acknowledge that both of us adhered to the work and their corresponding deadlines assigned to us by our respective partners.

Shourya Shukla Raghav Mittal

A detailed workload distribution for this project, between the two participants, as per the demand of the prompt, is specified below:

Shourya Shukla:

1) Installed the interface for running the code, and various libraries without which this project could not have been a reality.
2) Helped choose an efficient dataset from a variety of databases.
3) Implemented models based on sample codes, related work, documentation and suggestions provided.
4) Made significant contributions that helped formulate detailed analysis of various models.
5) Played an exceptional role at maintaining decorum and good spirit in times where all hope was lost. Lead by example and was the driving force for the success of the project.

Raghav Mittal:

1) Duly prepared the project report as per the schedule and in accordance to the satisfactory levels from Shourya.
2) Provided sample models from github, stackoverflow and other similar websites with recommendations.
3) Gave relevant suggestions based on observations.

4) Took pictures of the order 10^2. Helped in categorization and arrangement of pictures. Further, developed augmentation models to add appropriate noise to the datasets.
5) Recorded data by taking screenshots and notes by hand to avoid mess and confusion.

Credits and Word of Thanks

This paper hold the following men and women in high regard and gratitude. It extends a word of appreciation and acknowledges the amount of faith and sincerity that the following bestowed upon us:

The following people were of great help when we were out of ideas on how to improve our model further-

1) Prof. Ravi Kothari, IBM
2) Vijay Lingam (Teaching Assistant for the Intro to Machine Learning course at Ashoka University)

The people mentioned below, along with others allowed us to capture pictures of them for our dataset-

3) Koishore Roy (current 3rd year undergraduate at Ashoka University)
4) Nishka Dasgupta (current 2nd year undergraduate at Ashoka University)
5) Aditya P (current 2nd undergraduate at Ashoka University) AKA Aditya Dot
6) Vinayak Aggarwal (current 2nd undergraduate at Ashoka University)
7) Mr. Neeraj (G4S security force personnel, deployed outside the Computer Lab)
8) On-spot Young India Fellows who decided to volunteer
9) Raghu (current 2nd undergraduate at Ashoka University)
10) Akbar Surani (current 2nd undergraduate at Ashoka University)
11) Sanchit Bansal (current 2nd undergraduate at Ashoka University)

Citations

1) Collection of facial images, Dr. Libor Spacek. University of Essex, 2007.