Unimodal Biometric Authentication system using face recognition with Convolutional Neural Network

Vutlhari Hlungwane : 1882766
Mpendulo Nxumalo : 2154519

June 18, 2022

Contents

1 Introduction

Convolutional Neural Networks are a special type of Deep learning Artificial Neural Networks that are used mostly for image classification in computer vision.The architecture is made of several layers,one of which is the convolution layer from which they derive their name and sets it apart from the basic fully connected layer neural networks.They also have Pooling layers such as MaxPooling layer or AveragePooling that perform down sampling of the input data thus reducing the dimensions of the data.In this project CNN are used to classify authorized user that form the training data/are enrolled into a database.
The system that we present in this project uses these convolutional neural networks to learn to classify the face images that are enrolled in the dataset.To authenticate we set a threshold for accuracy level.So when an image(we assume that the image presented is of a face and the model recognizes that is it a face) is presented to the system the CNN will use the trained parameters to try to match with the labels in the system so when an image is not in the system it will have low accuracy rates thus the user will be denied access and otherwise the user is authenticated.
In section 2 we give details of the data set that was used to train the model,in section 3 we provide details of the various methods and techniques used to process the data.section 4 gives the baseline model,section 5 presents the results from the experiments,section 6 presents the optimization techniques used to increase accuracy and section 7 concludes the report.

2 Data set details

The original data set that is used in this project was downloaded from kaggle. But due to computational resource constraints we downsized the data set. In our tailored data set there are 10 people enrolled and each with 50 pictures thus our total data set consists of 500 pictures of different celebrities of different genders,ethnicity and age. The original pictures are colour pictures of various size but for the purpose of this project we resized all the picture to 50x50 gray scale images, Using one hot encoding each person in the data set is given a label and each image is assigned that label.

```
[[array([[143, 108,  79, ...,  46,  36,  37],
         [ 99,  85,  84, ...,  44,  70,  56],
         [ 96,  70,  79, ...,  95,  69,  50],
         ...,
         [187, 185, 181, ...,  53,  45,  31],
         [183, 179, 194, ...,  53,  53,  53],
         [191, 191, 193, ...,  49,  57,  60]], dtype=uint8), array([0, 0, 0, 0, 0, 0, 1, 0, 0
```

The following are the labels/names of authorized users enrolled in the data set and also new users can been enrolled. 1.Brad Pitt

2.Alia Bhatt
3.Anushka Sharma
4.Andy Samberg
5.Alexandra Daddario
6.Ellen Degeneres
7.Hugh Jackman
8.Jessica Alba
9.Amitabh Bachchan
10.Akshay Kumnar

3 Data Processing and Exploration

In this section we explore the relationships of the data and other underlying patterns in the dataset.This section is divided into 2 parts, The first is the data exploration and the second is the Data Pre-processing that gives details of the techniques that were used to turn the pictures in the dataset into more CNN readable data.

3.1 Data Exploration

In this project about 50% of the data set was used for training the model,20% was used for validation and the last 20 was used for testing.The data set was split this way so that the models parameters can be tuned using the keras tuner libraries to ensure higher accuracy on the classification of the people enrolled in the data set.

3.2 Data Preprocessing

The original dataset contained pictures of coloured faces.To reduce the complexity of the dataset we used the OpenCV libraries to convert the pictures to grey scale images of size (50x50) pixels.The dataset was 3D dimensions but the CNN we use in this project take in inputs of 4D dimensions so we used tensorflow libraries to add an extra dimension without change the dataset.So our final dataset is 4D.After splliting the dataset as explained above our dataset has the following shapes:

```
print(X_train.shape)
print(X_test.shape)
print(X_validate.shape)
print(Y_train.shape)
print(Y_test.shape)
print(Y_validate.shape)
```

```
(300, 50, 50, 1)
(100, 50, 50, 1)
(100, 50, 50, 1)
(300, 10)
(100, 10)
(100, 10)
```

4 Baseline model

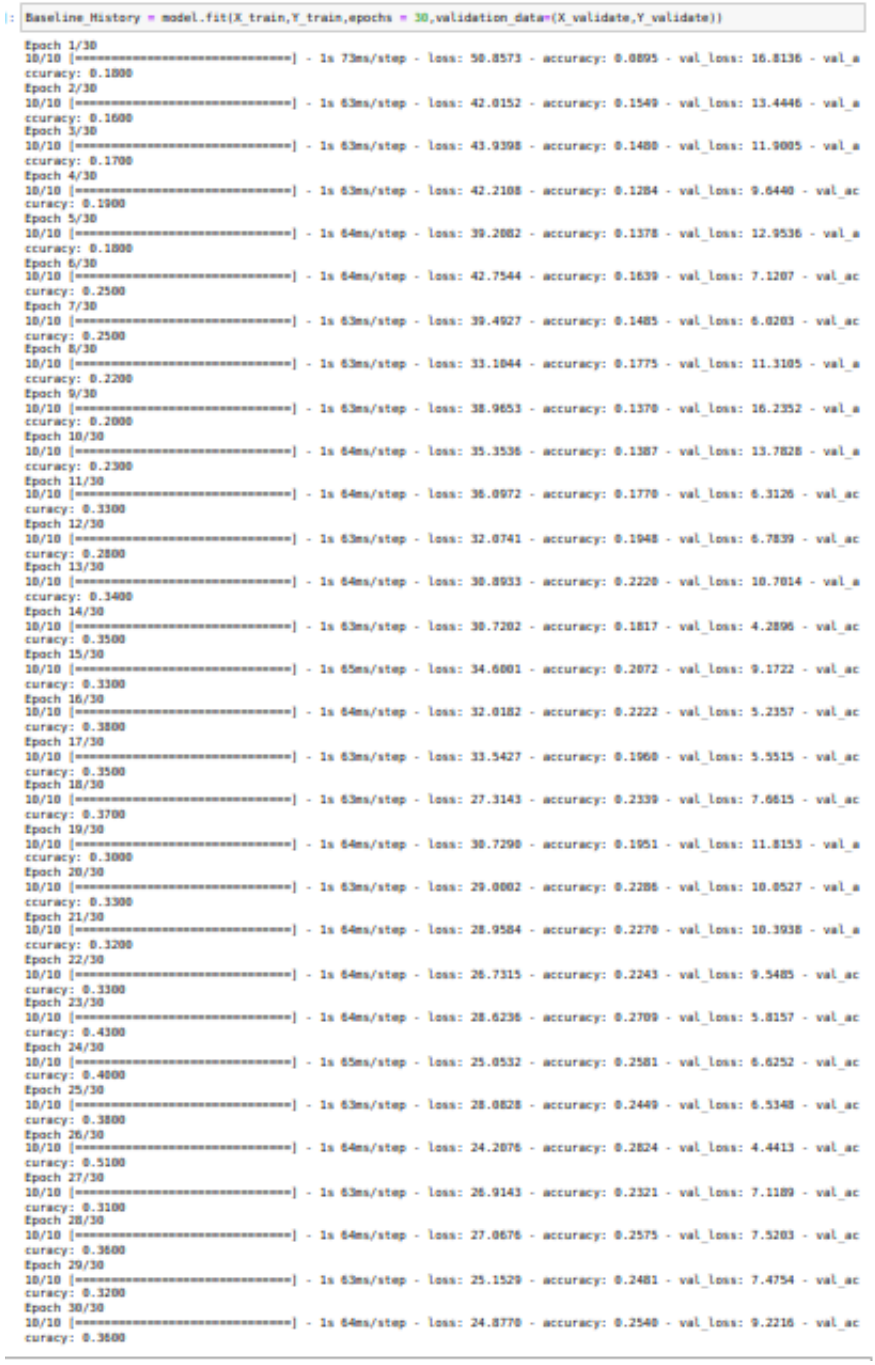
The implementation of the baseline CNN algorithm is presented in the Linear.ipynb jupyter notebook that goes with this report.The CNN algorithm was written in python using Tensorflow and Keras libraries.Since the input is a 2D images we use 2D convolutional layers from the kears libraries and the first of these layers accepts (50x50x1) dimensional data as input,The dimensions correspond to the height ,width and number of channels per pixels of the image but since we converted the image to grays scale the channel is 1.This layer is followed by Max-pooling layer that reduces the dimensionality of the data, which is follow by a flatten layer and after a couple of Dense layers are added.For regularization purposes a droupout layer is added to avoid over-fitting. The architecture of the baseline model is presented in the following figure

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 9, 9, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 2592)              0
_________________________________________________________________
dense (Dense)                (None, 2000)              5186000
_________________________________________________________________
dense_1 (Dense)              (None, 3000)              6003000
_________________________________________________________________
dense_2 (Dense)              (None, 40000)             120040000
_________________________________________________________________
dropout (Dropout)            (None, 40000)             0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                400010
_________________________________________________________________
activation (Activation)      (None, 10)                0
=================================================================
Total params: 131,629,330
Trainable params: 131,629,330
Non-trainable params: 0
```

The model has a total of 131 629 330 trainable parameters.

5 Results

The results of the baseline model is presented in the figure below



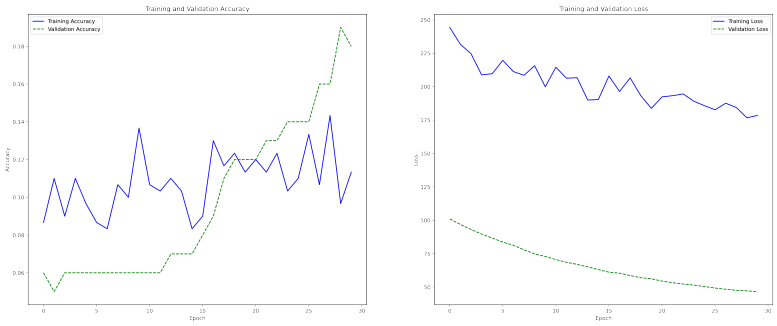
```
Baseline_History = model.fit(X_train,Y_train,epochs = 30,validation_data=(X_validate,Y_validate))

Epoch 1/30
10/10 [==============================] - 1s 73ms/step - loss: 50.8573 - accuracy: 0.0895 - val_loss: 16.8136 - val_a
ccuracy: 0.1800
Epoch 2/30
10/10 [==============================] - 1s 63ms/step - loss: 42.0152 - accuracy: 0.1549 - val_loss: 13.4446 - val_a
ccuracy: 0.1600
Epoch 3/30
10/10 [==============================] - 1s 63ms/step - loss: 43.9398 - accuracy: 0.1480 - val_loss: 11.9005 - val_a
ccuracy: 0.1700
Epoch 4/30
10/10 [==============================] - 1s 63ms/step - loss: 42.2108 - accuracy: 0.1284 - val_loss: 9.6440 - val_ac
curacy: 0.1900
Epoch 5/30
10/10 [==============================] - 1s 64ms/step - loss: 39.2082 - accuracy: 0.1378 - val_loss: 12.9536 - val_a
ccuracy: 0.1800
Epoch 6/30
10/10 [==============================] - 1s 64ms/step - loss: 42.7544 - accuracy: 0.1639 - val_loss: 7.1207 - val_ac
curacy: 0.2500
Epoch 7/30
10/10 [==============================] - 1s 63ms/step - loss: 39.4927 - accuracy: 0.1485 - val_loss: 6.0203 - val_ac
curacy: 0.2500
Epoch 8/30
10/10 [==============================] - 1s 63ms/step - loss: 33.1044 - accuracy: 0.1775 - val_loss: 11.3105 - val_a
ccuracy: 0.2200
Epoch 9/30
10/10 [==============================] - 1s 63ms/step - loss: 38.9653 - accuracy: 0.1370 - val_loss: 16.2352 - val_a
ccuracy: 0.2000
Epoch 10/30
10/10 [==============================] - 1s 64ms/step - loss: 35.3536 - accuracy: 0.1387 - val_loss: 13.7828 - val_a
ccuracy: 0.2300
Epoch 11/30
10/10 [==============================] - 1s 64ms/step - loss: 36.0972 - accuracy: 0.1770 - val_loss: 6.3126 - val_ac
curacy: 0.3300
Epoch 12/30
10/10 [==============================] - 1s 63ms/step - loss: 32.0741 - accuracy: 0.1948 - val_loss: 6.7839 - val_ac
curacy: 0.2800
Epoch 13/30
10/10 [==============================] - 1s 64ms/step - loss: 30.8933 - accuracy: 0.2220 - val_loss: 10.7014 - val_a
ccuracy: 0.3400
Epoch 14/30
10/10 [==============================] - 1s 63ms/step - loss: 30.7202 - accuracy: 0.1817 - val_loss: 4.2896 - val_ac
curacy: 0.3500
Epoch 15/30
10/10 [==============================] - 1s 65ms/step - loss: 34.6001 - accuracy: 0.2072 - val_loss: 9.1722 - val_ac
curacy: 0.3300
Epoch 16/30
10/10 [==============================] - 1s 64ms/step - loss: 32.0182 - accuracy: 0.2222 - val_loss: 5.2357 - val_ac
curacy: 0.3800
Epoch 17/30
10/10 [==============================] - 1s 63ms/step - loss: 33.5427 - accuracy: 0.1960 - val_loss: 5.5515 - val_ac
curacy: 0.3500
Epoch 18/30
10/10 [==============================] - 1s 63ms/step - loss: 27.3143 - accuracy: 0.2339 - val_loss: 7.6615 - val_ac
curacy: 0.3700
Epoch 19/30
10/10 [==============================] - 1s 64ms/step - loss: 30.7290 - accuracy: 0.1951 - val_loss: 11.8153 - val_a
ccuracy: 0.3000
Epoch 20/30
10/10 [==============================] - 1s 63ms/step - loss: 29.0002 - accuracy: 0.2286 - val_loss: 10.0527 - val_a
ccuracy: 0.3300
Epoch 21/30
10/10 [==============================] - 1s 64ms/step - loss: 28.9584 - accuracy: 0.2270 - val_loss: 10.3938 - val_a
ccuracy: 0.3200
Epoch 22/30
10/10 [==============================] - 1s 64ms/step - loss: 26.7315 - accuracy: 0.2243 - val_loss: 9.5485 - val_ac
curacy: 0.3300
Epoch 23/30
10/10 [==============================] - 1s 64ms/step - loss: 28.6236 - accuracy: 0.2709 - val_loss: 5.8157 - val_ac
curacy: 0.4300
Epoch 24/30
10/10 [==============================] - 1s 65ms/step - loss: 25.0532 - accuracy: 0.2581 - val_loss: 6.6252 - val_ac
curacy: 0.4000
Epoch 25/30
10/10 [==============================] - 1s 63ms/step - loss: 28.0828 - accuracy: 0.2449 - val_loss: 6.5348 - val_ac
curacy: 0.3800
Epoch 26/30
10/10 [==============================] - 1s 64ms/step - loss: 24.2876 - accuracy: 0.2824 - val_loss: 4.4413 - val_ac
curacy: 0.5100
Epoch 27/30
10/10 [==============================] - 1s 63ms/step - loss: 26.9143 - accuracy: 0.2321 - val_loss: 7.1189 - val_ac
curacy: 0.3100
Epoch 28/30
10/10 [==============================] - 1s 64ms/step - loss: 27.0676 - accuracy: 0.2575 - val_loss: 7.5203 - val_ac
curacy: 0.3600
Epoch 29/30
10/10 [==============================] - 1s 63ms/step - loss: 25.1529 - accuracy: 0.2481 - val_loss: 7.4754 - val_ac
curacy: 0.3200
Epoch 30/30
10/10 [==============================] - 1s 64ms/step - loss: 24.8770 - accuracy: 0.2540 - val_loss: 9.2216 - val_ac
curacy: 0.3600
```

A summary of the results when the model in tested on the test set of 100 gray scale pictures is given in the figure below

```
4/4 [==============================] - 0s 6ms/step - loss: 3.6226 - accu
racy: 0.5200
Testing Loss 3.6226487159729004
Testing Accuracy 52.00%
```

The following is a more graphical representation of the accuracy of the train and validation



The model has final classification of 52% classification accuracy which is quite impressive given that we have a very small dataset but the model can be improved by adding more data or employing normalization techniques which we found difficult to do on 4 dimensional data.But since this is an authentication system where we set a threshold on the accuracy result to either authenticate a user that is enrolled on the dataset or deny access to users that are not in the dataset who the model can't recognize.
To improve the learning of the model we use hyperparameters presented in the following section.

6 Optimization

In this project when we tried to hyper tune the models to get a better model we actually got a less accurate model than our baseline model thus we decided to stick to the baseline model for the purpose of this project
The following figure show the best model we could get after hyper tuning as it can be seen it gives accuracy of 46% which is less than the one for the baseline.

```
Trial summary
Hyperparameters:
Conv_Filters_1: 16
Conv_Filters_2: 16
Conv_Activation_2: sigmoid
Dropout_Rate_1: 0.25
Conv_Filters_3: 32
Conv_Activation_3: relu
Conv_Filters_4: 128
Conv_Activation_4: relu
Dropout_Rate_2: 0.5
Last_Dense_NumUnits: 112
Dropout_Rate_3: 0.35000000000000003
Adam_Optim_LR: 0.0005674460859990781
Score: 0.4599999934434891
Trial summary
```

7 Conclusion

This project is about using deep learning for authentication systems that biometric data specifically the face.By using convolutional neural networks we were able to learn certain parameter that would be used in the authentication stage(Which is not in the scope of the current project).When the model recognizes a face upon which it was trained it will give a certain matching accuracy level and if the accuracy level is above the set threshold(Different for different systems) the user would be authenticated.Due to the size of the datset the model only could accurately match 52% of the time.Thus with more computaional power and more data the model could improve.