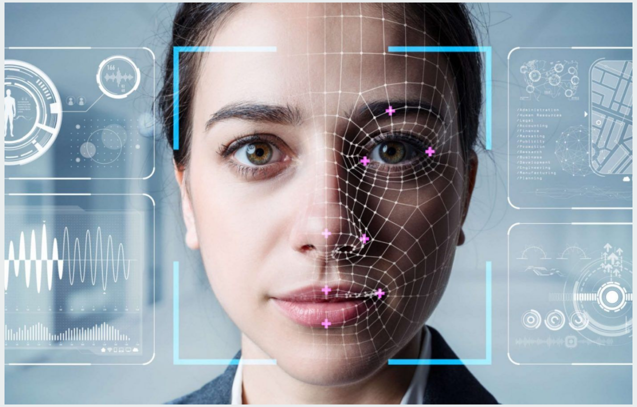
Facial Recognition

Nathan Langley, Greg Williams, Sam Luu

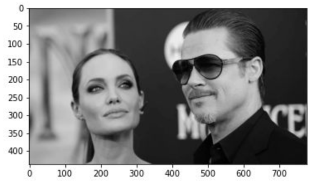


Motivation

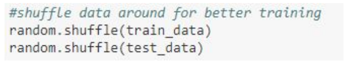
- Testing the different algorithms and concepts to create a simple facial recognition
- Test the accuracy of different concepts
    - CNN vs CRC
- Becoming more significant and common in society

Approach - Creating Dataset

- Using 5 different celebrities with 700 images for training and 100 images for testing
- Utilized OpenCV to read and add grayscale effect to images



- Data is shuffled before storing to increase training



```
#shuffle data around for better training
random.shuffle(train_data)
random.shuffle(test_data)
```

Approach - CRC

- Using the direct solution

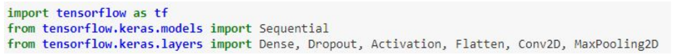
$$\boldsymbol{P} = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \cdot \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T$$

- Depending on accuracy results:
  - K-fold cross validation at 10 folds for training and validation sets
  - Each sub training and validation set will hold 70 images each
  - Found that k=10 is common as it produces low bias and a decent variance

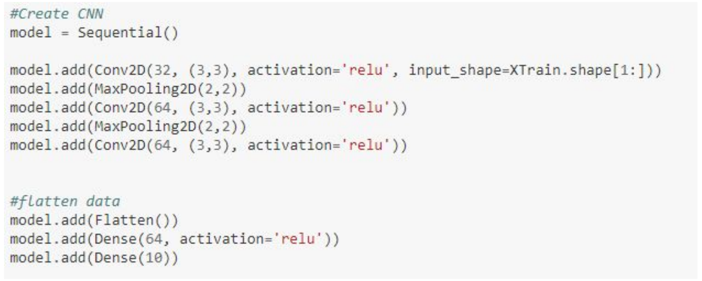
CRC Problems To Fix

- Because of the large dataset, it can not be stored in a normal array.
- Due to this we are currently attempting to swap it to a disk based data structure.

Approach - CNN

- Using TensorFlow Keras
- 

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
```



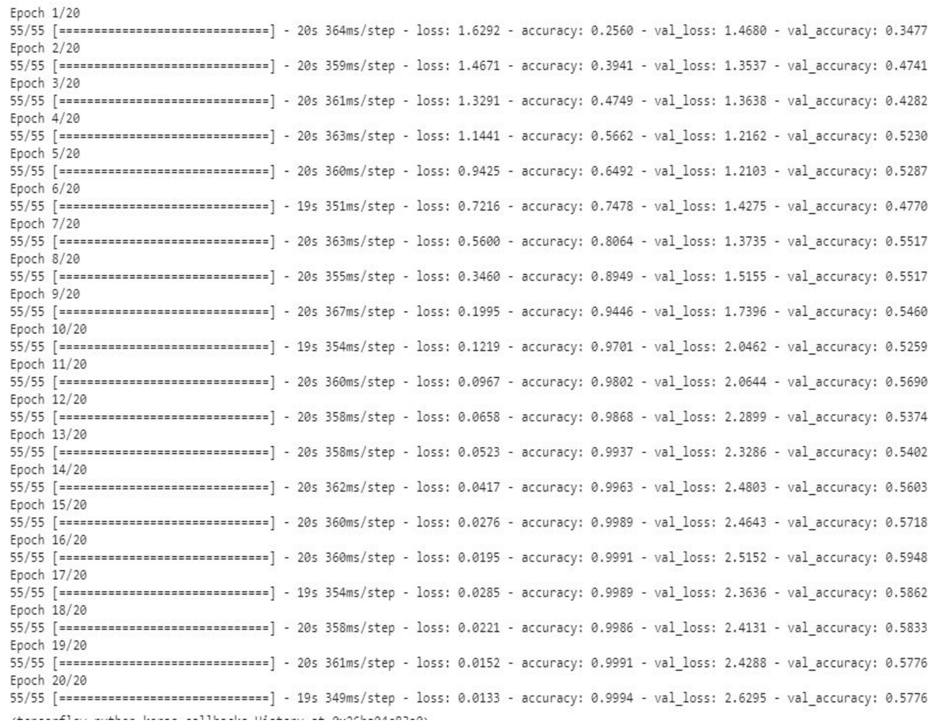
```
#Create CNN
model = Sequential()

model.add(Conv2D(32, (3,3), activation='relu', input_shape=XTrain.shape[1:]))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3), activation='relu'))


#flatten data
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(10))
```

Results

- 10 Epochs: gives a great accuracy
- 16 Epochs: gives the best accuracy
- 16+ Epochs: starts to diverge and re-converge



```
Epoch 1/20
55/55 [==============================] - 20s 364ms/step - loss: 1.6292 - accuracy: 0.2560 - val_loss: 1.4680 - val_accuracy: 0.3477
Epoch 2/20
55/55 [==============================] - 20s 359ms/step - loss: 1.4671 - accuracy: 0.3941 - val_loss: 1.3537 - val_accuracy: 0.4741
Epoch 3/20
55/55 [==============================] - 20s 361ms/step - loss: 1.3291 - accuracy: 0.4749 - val_loss: 1.3638 - val_accuracy: 0.4282
Epoch 4/20
55/55 [==============================] - 20s 363ms/step - loss: 1.1441 - accuracy: 0.5662 - val_loss: 1.2162 - val_accuracy: 0.5230
Epoch 5/20
55/55 [==============================] - 20s 360ms/step - loss: 0.9425 - accuracy: 0.6492 - val_loss: 1.2103 - val_accuracy: 0.5287
Epoch 6/20
55/55 [==============================] - 19s 351ms/step - loss: 0.7216 - accuracy: 0.7478 - val_loss: 1.4275 - val_accuracy: 0.4770
Epoch 7/20
55/55 [==============================] - 20s 363ms/step - loss: 0.5600 - accuracy: 0.8064 - val_loss: 1.3735 - val_accuracy: 0.5517
Epoch 8/20
55/55 [==============================] - 20s 355ms/step - loss: 0.3460 - accuracy: 0.8949 - val_loss: 1.5155 - val_accuracy: 0.5517
Epoch 9/20
55/55 [==============================] - 20s 367ms/step - loss: 0.1995 - accuracy: 0.9446 - val_loss: 1.7396 - val_accuracy: 0.5460
Epoch 10/20
55/55 [==============================] - 19s 354ms/step - loss: 0.1219 - accuracy: 0.9701 - val_loss: 2.0462 - val_accuracy: 0.5259
Epoch 11/20
55/55 [==============================] - 20s 360ms/step - loss: 0.0967 - accuracy: 0.9802 - val_loss: 2.0644 - val_accuracy: 0.5690
Epoch 12/20
55/55 [==============================] - 20s 358ms/step - loss: 0.0658 - accuracy: 0.9868 - val_loss: 2.2899 - val_accuracy: 0.5374
Epoch 13/20
55/55 [==============================] - 20s 358ms/step - loss: 0.0523 - accuracy: 0.9937 - val_loss: 2.3286 - val_accuracy: 0.5402
Epoch 14/20
55/55 [==============================] - 20s 362ms/step - loss: 0.0417 - accuracy: 0.9963 - val_loss: 2.4803 - val_accuracy: 0.5603
Epoch 15/20
55/55 [==============================] - 20s 360ms/step - loss: 0.0276 - accuracy: 0.9989 - val_loss: 2.4643 - val_accuracy: 0.5718
Epoch 16/20
55/55 [==============================] - 20s 360ms/step - loss: 0.0195 - accuracy: 0.9991 - val_loss: 2.5152 - val_accuracy: 0.5948
Epoch 17/20
55/55 [==============================] - 19s 354ms/step - loss: 0.0285 - accuracy: 0.9989 - val_loss: 2.3636 - val_accuracy: 0.5862
Epoch 18/20
55/55 [==============================] - 20s 358ms/step - loss: 0.0221 - accuracy: 0.9986 - val_loss: 2.4131 - val_accuracy: 0.5833
Epoch 19/20
55/55 [==============================] - 20s 361ms/step - loss: 0.0152 - accuracy: 0.9991 - val_loss: 2.4288 - val_accuracy: 0.5776
Epoch 20/20
55/55 [==============================] - 19s 349ms/step - loss: 0.0133 - accuracy: 0.9994 - val_loss: 2.6295 - val_accuracy: 0.5776
<tensorflow.python.keras.callbacks.History at 0x[illegible]>
```

Future Plans

- Finish/fine-tune CRC approach
- Fine-tune parameters for CNN to get better accuracy
- Compare CNN accuracies with CRC accuracies

References

- https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=k-Fold%20Cross-Validation%20Cross-validation%20is%20a%20resampling%20procedure%20used,such%2C%20the%20procedure%20is%20often%20called%20k-fold%20cross-validation.