EMPLOYEE ATTENDANCE USING FACIAL RECOGNITION

Abdulrahman Alrubaiya
Hatim Alshehri
Mohammed Alghamdi

Table of Content

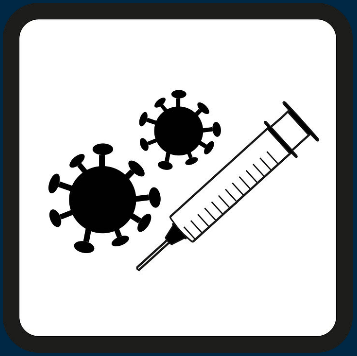
- → Introduction
- → Objective
- → Tools
- → Models
- → Lessons learned

Introduction

- Facial recognition is important in different domains
  - User experience
  - Public safety and national security
  - Control access
- Physiological biometric
  - Fingerprint
  - Iris etc..
- Can be used on existing hardware infrastructure
  - Cameras, image capturing devices

Objective

- Current pandemic
  - Contactless
- Employee attendance
- Liveness / spoofed
- Emotions detection
- Alerts
  - Unauthorized personnel
  - Motivational quotes
  - Mediums
    - SMS, Email etc..



TOOLS

OpenCV

NumPy

twilio

Keras

matplotlib

Models:

01

Model: Liveness

- Objective
- Data
- Baseline
- Network architecture
- Eval metrics
- Challenges

02

Model: Facial Recognition

- Objective
- Data
- Baseline
- Network architecture
- Eval metrics
- Challenges

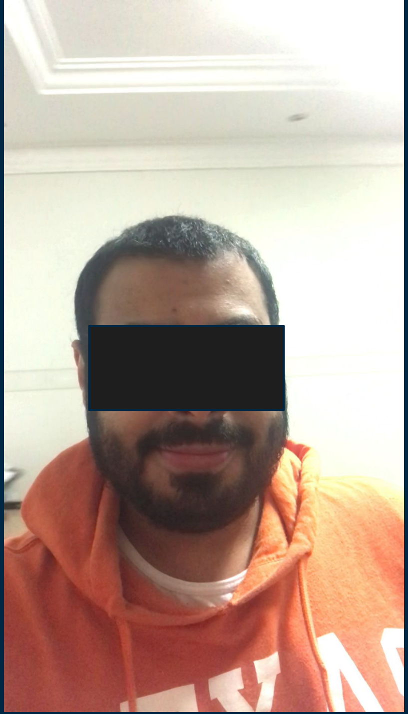
03

Model: Emotion detection

- Objective
- Data
- Baseline
- Network architecture
- Eval metrics
- Challenges

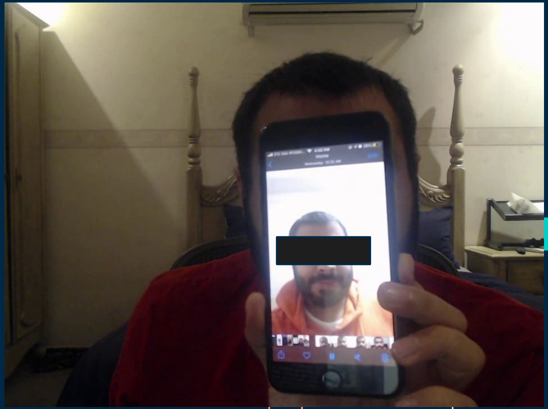
Liveness detection

- Objective: Create liveness detector capable of identifying real and fake/spoofed faces
- Data:
  - 1 minute video on iPhone for real, 1 minute video on Logitech Webcam for fake
  - 3600 images(1800 for each class), 300x300



Real (1080x1920)

Fake/Spoofed (960x720)



Liveness (Cont.)

- Preprocessing:
- Resize image to 32x32
- Intensify images to range of 0, 1
- Label encode (real, fake)
- One hot encode
- Split data train, validation 75-25

Data augmentation

- Rotation 20
- Zoom 0.15
- Width shift 0.2
- Height shift 0.2
- Horizontal flip

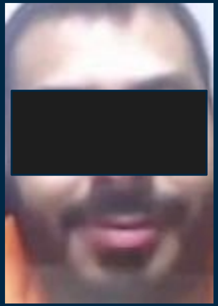
Liveness (Cont.)

- Baseline:
  - Simple CNN
    - 32, 16, D(24), 2 → Kernel=3, ReLU, Maxpooling 2x2, softmax

Set	Score
Training	0.86
Validation	0.84

Real

Fake



Liveness using LivenessNet (Cont.)

- LivenessNet
  - Hidden layers: 6
  - First:
    - Conv2D_1: Filters=16, kernel_size=3, padding=same
    - Activation=ReLU
    - BatchNormalization
    - MaxPooling=2x2
    - Dropout=0.25
  - Second:
    - Conv2D_2: Filters=32, kernel_size=3, padding=same
    - Activation=ReLU
    - BatchNormalization
    - MaxPooling=2x2
    - Dropout=0.25

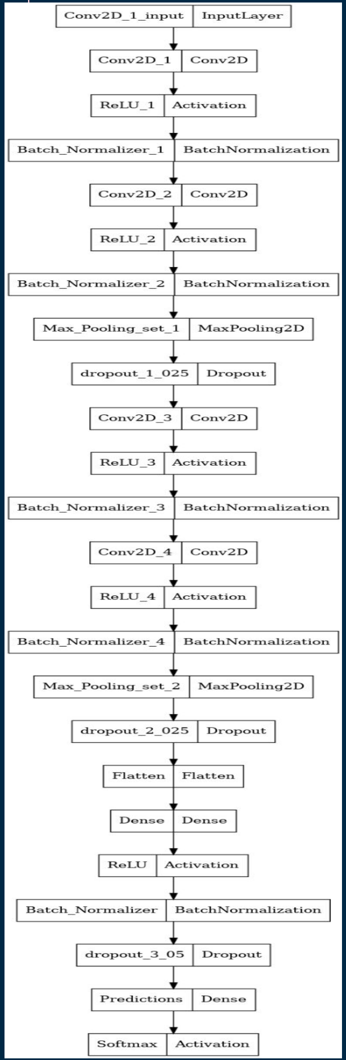
First: Conv => ReLU => Conv => ReLU=> Pool
Second: Conv => ReLU=> Conv => ReLU => Pool
Flatten set: Flatten() => Dense(64) => ReLU
Softmax classifier: Dense(2) => Softmax

- Loss=binary_crossentropy
- Optimizer=adam
- Metrics=accuracy
- Epochs=50, Batch_size=8

Liveness using LivenessNet (Cont.)

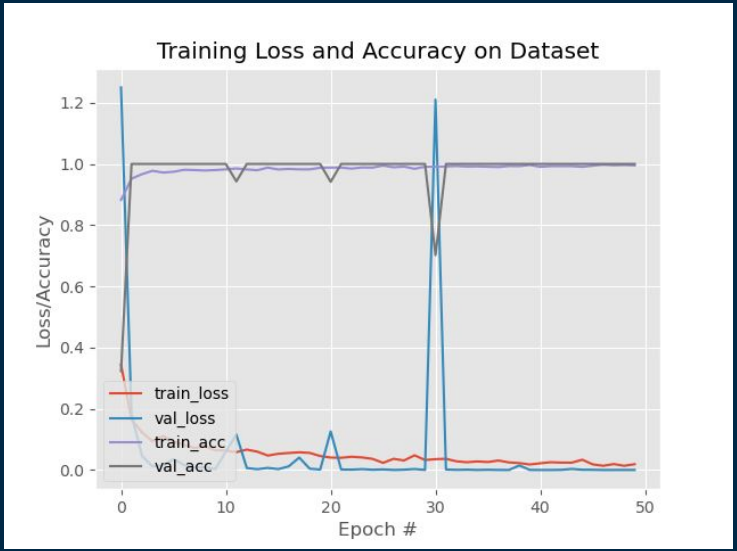
```
In [3]: # model summary
        model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv2D_1 (Conv2D)           (None, 32, 32, 16)        448

 ReLU_1 (Activation)         (None, 32, 32, 16)        0

 Batch_Normalizer_1 (BatchNo  (None, 32, 32, 16)       64
 rmalization)

 Conv2D_2 (Conv2D)           (None, 32, 32, 16)        2320

 ReLU_2 (Activation)         (None, 32, 32, 16)        0

 Batch_Normalizer_2 (BatchNo  (None, 32, 32, 16)       64
 rmalization)

 Max_Pooling_set_1 (MaxPooli  (None, 16, 16, 16)       0
 ng2D)

 dropout_1_025 (Dropout)     (None, 16, 16, 16)        0

 Conv2D_3 (Conv2D)           (None, 16, 16, 32)        4640

 ReLU_3 (Activation)         (None, 16, 16, 32)        0

 Batch_Normalizer_3 (BatchNo  (None, 16, 16, 32)       128
 rmalization)

 Conv2D_4 (Conv2D)           (None, 16, 16, 32)        9248

 ReLU_4 (Activation)         (None, 16, 16, 32)        0

 Batch_Normalizer_4 (BatchNo  (None, 16, 16, 32)       128
 rmalization)

 Max_Pooling_set_2 (MaxPooli  (None, 8, 8, 32)         0
 ng2D)

 dropout_2_025 (Dropout)     (None, 8, 8, 32)          0

 Flatten (Flatten)           (None, 2048)              0

 Dense (Dense)               (None, 64)                131136

 ReLU (Activation)           (None, 64)                0

 Batch_Normalizer (BatchNorm  (None, 64)               256
 alization)

 dropout_3_05 (Dropout)      (None, 64)                0

 Predictions (Dense)         (None, 2)                 130

 Softmax (Activation)        (None, 2)                 0

=================================================================
Total params: 148,562
Trainable params: 148,242
Non-trainable params: 320
_________________________________________________________________
```

Liveness using LivenessNet (Cont.)



Liveness using LivenessNet (Cont.)

Set	Score
Training	1
Validation	1

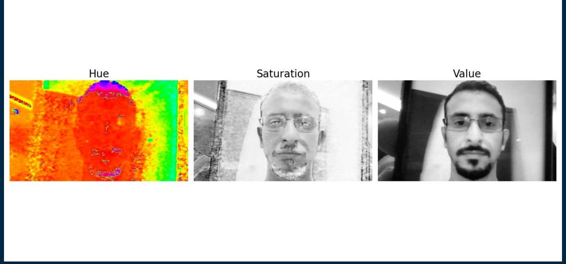


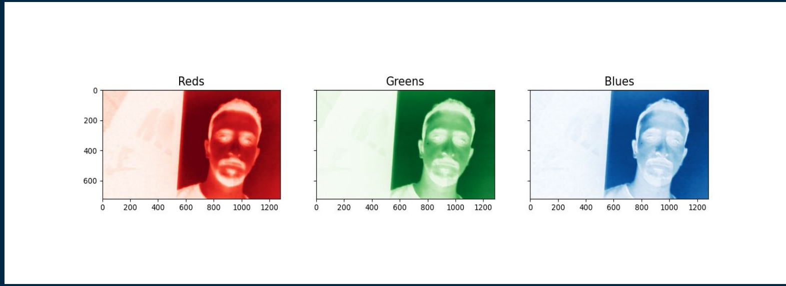
Liveness challenges

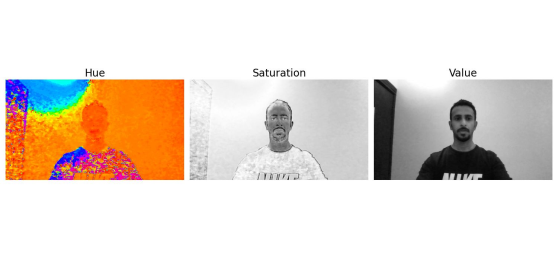
- Unable to generalize → Took frame by frame pictures instead of 1 frame per second
- White-lit room provides false predictions → Try different models

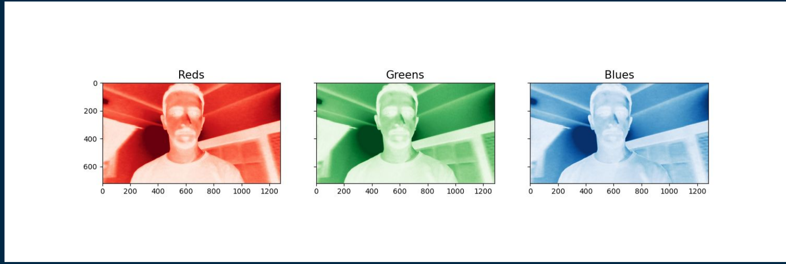
How problem solved

No good result with these images







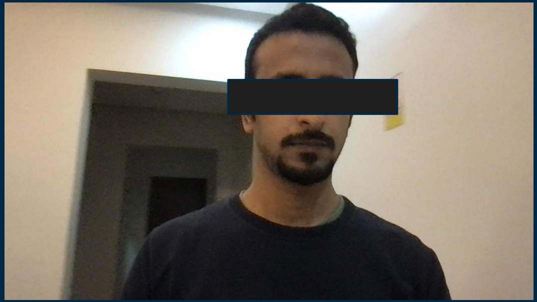


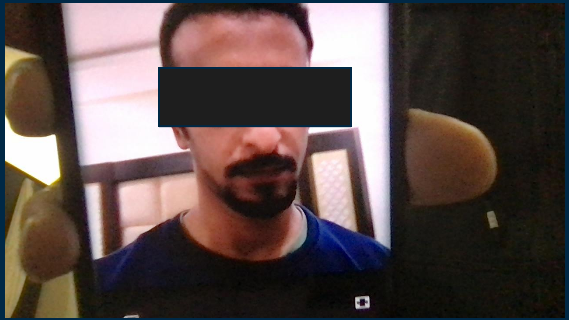
3400 Real

3400 with phone

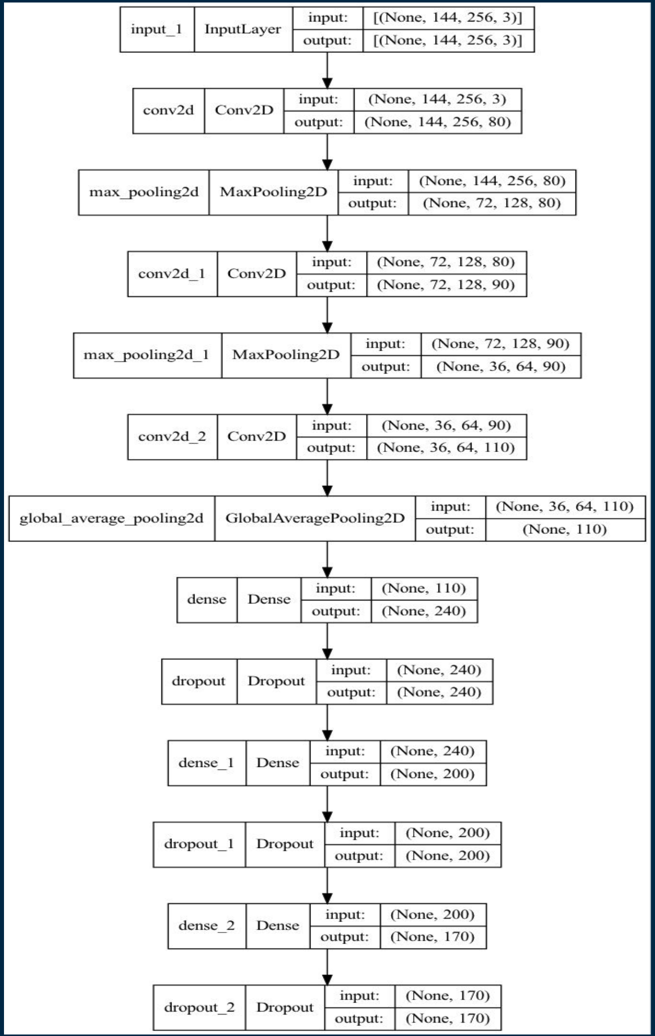
How problem solved

Using “Fast two-dimensional phase- unwrapping algorithm based on sorting by reliability following a noncontinuous path , 2002“ paper





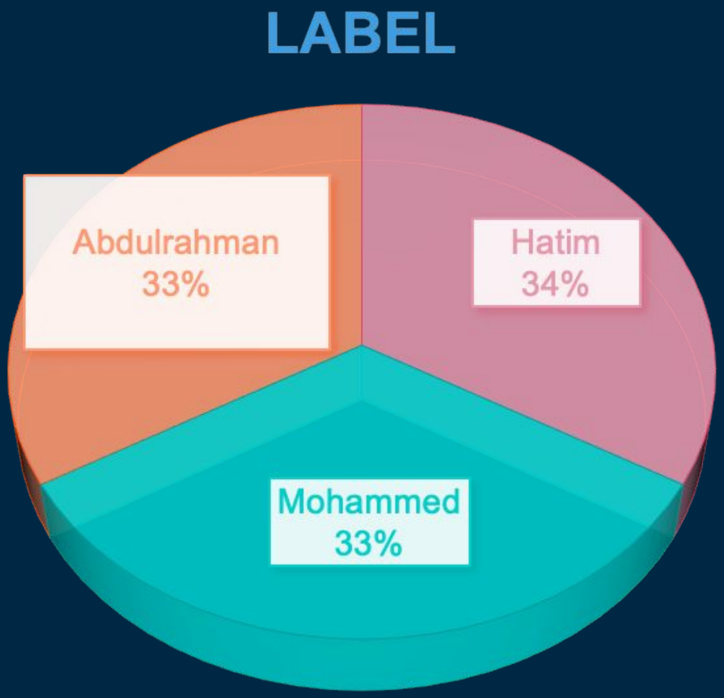
- accuracy: 0.9952
- val_accuracy: 0.9793
- Epoch 00057: early stopping
- Test accuracy: 0.9828



Facial Recognition Model:

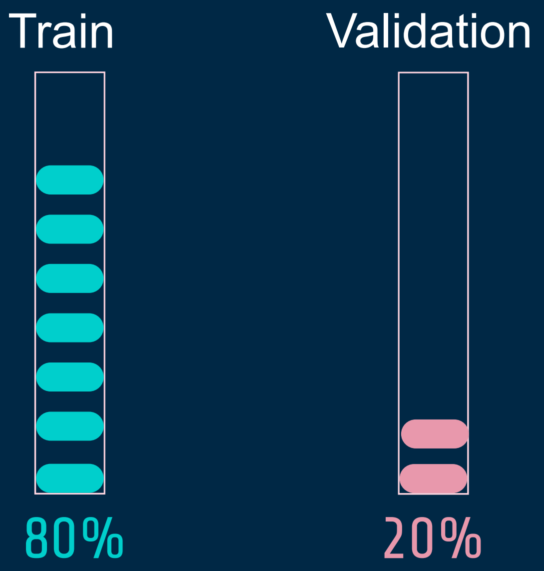
DATASET

- Facial Recognition.
- From OpenCv (ImageCapture).
- 9,000
- Multilabel:
    - Hatim
    - Abdulrahman
    - Mohammed

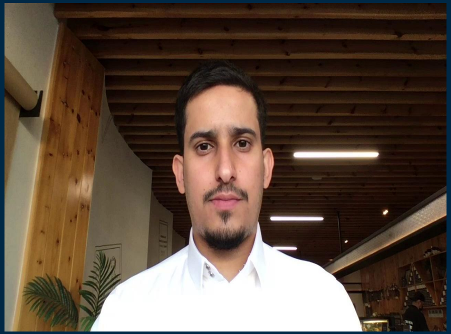


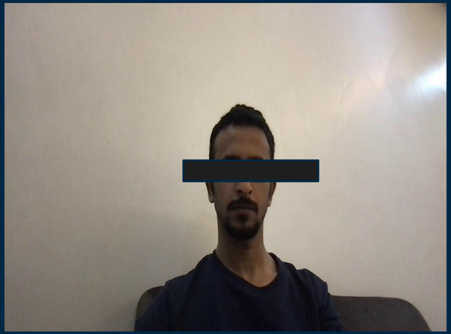
DATA AUGMENTATION

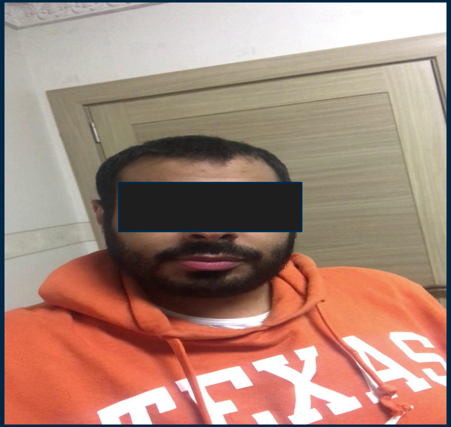
- Resize Images (220X220)
- Scaling/Normalization (1 to 255)
- Randomly Rotate Images (up to 20°)
- Randomly Flip Images (Horizontally)



DATA SAMPLE







Facial Recognition Models:

Simple NN

- Hidden layers: 1
- Dense: 16
- Activation: ReLU
- Optimizer: ADAM
- Loss function: Categorical Crossentropy
- Output Activation: SoftMax

CNN

- Layers added : 4
- Dense : 1000 – 850 – 1000 - 500
- BatchNormalization
- Activation : ReLU
- Optimizer : adam
- Weights : imagenet
- Include Top : False
- Loss function: Categorical Crossentropy
- Output Activation: SoftMax

Scores

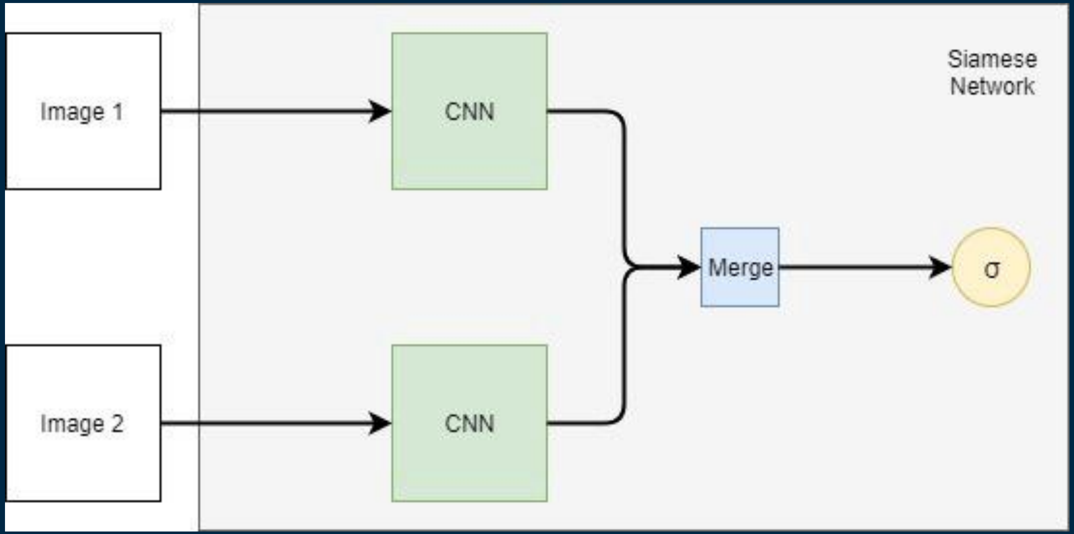
- Baseline Simple NN:
- Train Accuracy: 100%
- Validation Accuracy: 84.76%

- CNN
- Train Accuracy: 100%
- Validation Accuracy: 84.76%

Siamese Neural Networks Model:

After we tried the previous model in order to perform the facial recognition and got unsatisfactory results we decided to choose another approach to make this mission work well.

Siamese NN Architecture:



Data

- Bast Data : 17432 images
- 105 class
- Used :
- 23469 Images in total
- 4 classes



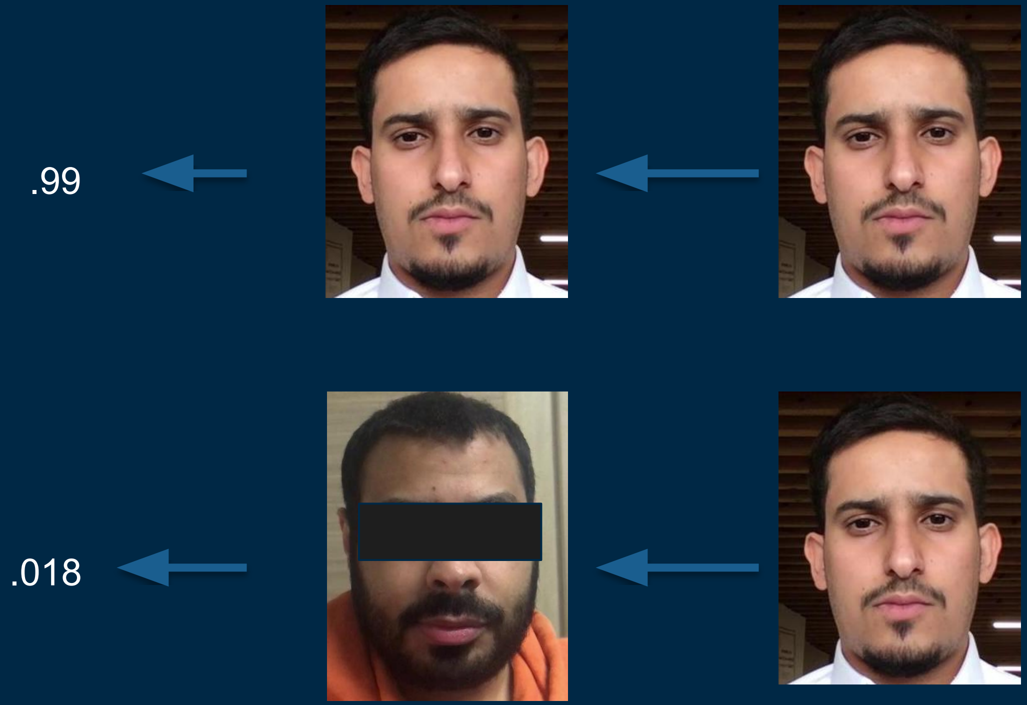

Model : Siamese Network

- Training accuracy: 0.9987
- val_accuracy: 0.99875
- Test accuracy: 0.9987

CNN

- Conv2D : 3
- Filter: 50 – 80 – 90
- Layers added : 3
- Dense : 150 – 190 – 120
- BatchNormalization
- Activation : ReLU
- Optimizer : adam

Similarity Example



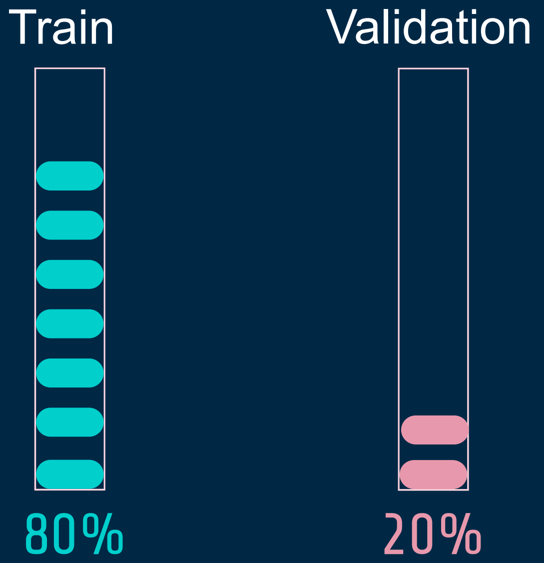
Emotion Detection Model

DATASET

- Facial Emotion Recognition
- From Kaggle
- ~ 37K
- Multilabel:
  - Neutral
  - Happy
  - Sad
  - Surprise
  - Fear
  - Disgust
  - Anger
- Greyscale

DATA PRE-PROCESSING

- Resize Images (142X256)
- Scaling/Normalization (0,1)
- Randomly Rotate Images (up to 20°)
- Randomly Flip Images (Horizontally)



DATA SAMPLE






Transfer Learning MODELS:

MobileNet

- Layers added : 4
- Dense : 1000 – 850 – 1000 - 500
- BatchNormalization
- Dropout : 0.3
- Activation : ReLU .
- Optimizer : adam

VGG-16

- Layers added : 3
- Dense : 1000 – 256 - 128
- BatchNormalization
- Dropout : 0.5
- Activation : ReLU.
- Optimizer : adam

- Loss function: Categorical Crossentropy
- Output Activation: SoftMax

MobileNet:

- Train Accuracy: 86.29%
- Validation Accuracy: 48.71%

VGG-16:

- Train Accuracy: 83.46%
- Validation Accuracy: 57.92%

Lessons learned

- Version control branch merging
- Github issues to track progress
- Academic papers
    - Stanford method
        - Three pass
            - 1: Title, abstract, introduction, headings, sub-headings, and conclusion
            - 2: Read and look at figures carefully
            - 3: Read paper carefully with all details

THANK YOU FOR LISTENING