VGGFace2 and FaceNet

Mask Detection|Emotion Recognition

Javae Elliott (jue203@nyu.edu) and
Petra Ivanović (pi2018@nyu.edu)

TABLE OF CONTENTS

01

MODELS USED

VGGFace2 and FaceNet512 summaries

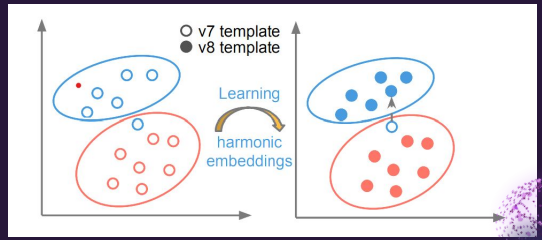
VGGFace2 Model

- Encodes a face into a representation of 2048 numbers
- Using Euclidean distance it calculates whether 2 images represent the same person
  - If the values are low (under the threshold) = same person
- Based on RESNET50 architecture
  - Convolutional network with 50 layers (48 convolutional layers, 1 MaxPool, 1 Average Pool layer)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FaceNet Model

- A deep neural network used for extracting features from an image of a person's face
- Uses Euclidean distance between face images' mapping for a measure of face similarity.
- Attempts a direct optimization of the embeddings themselves rather than "intermediate bottleneck approaches".



type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

02

ABSTRACT

Our problems and goals

GOALS AND OBJECTIVES

- Our goal is to use VGGFace and FaceNet to solve 2 problems, respectively
  - Recognise whether an individual is wearing a face mask in an image
  - Distinguish facial expressions in images

- Models that solve these problems could be used for
  - Disease spread tracking/risk analysis
  - Face recognition, verification

03

OUR DATA SETS

VGGFace Dataset,
Face Mask Detection Dataset,
Facial Expression Recognition Dataset

Kaggle Face Mask Detection dataset

We use the kaggle Face Mask Detection Dataset created by Ashish Jangra.

This is a dataset comprised of 11,792 images divided into training, testing, and validation.

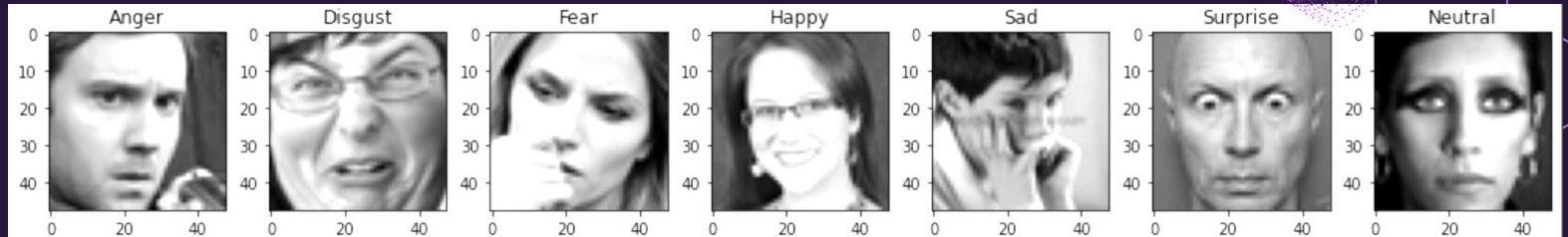
- Training set is comprised of 10,000 images
- Validation set is comprised of 800 images
- Testing set is comprised of 992 images

Kaggle Facial Expression Recognition dataset

We use the kaggle Facial Expression Recognition dataset created by Dumitru, Ian Goodfellow, Yoshua Bengio.

This is a dataset comprised of 48x48 grayscale images of faces divided into training and testing sets.

- Training set is comprised of 28,709 images
- Testing set is comprised of 3,589 images

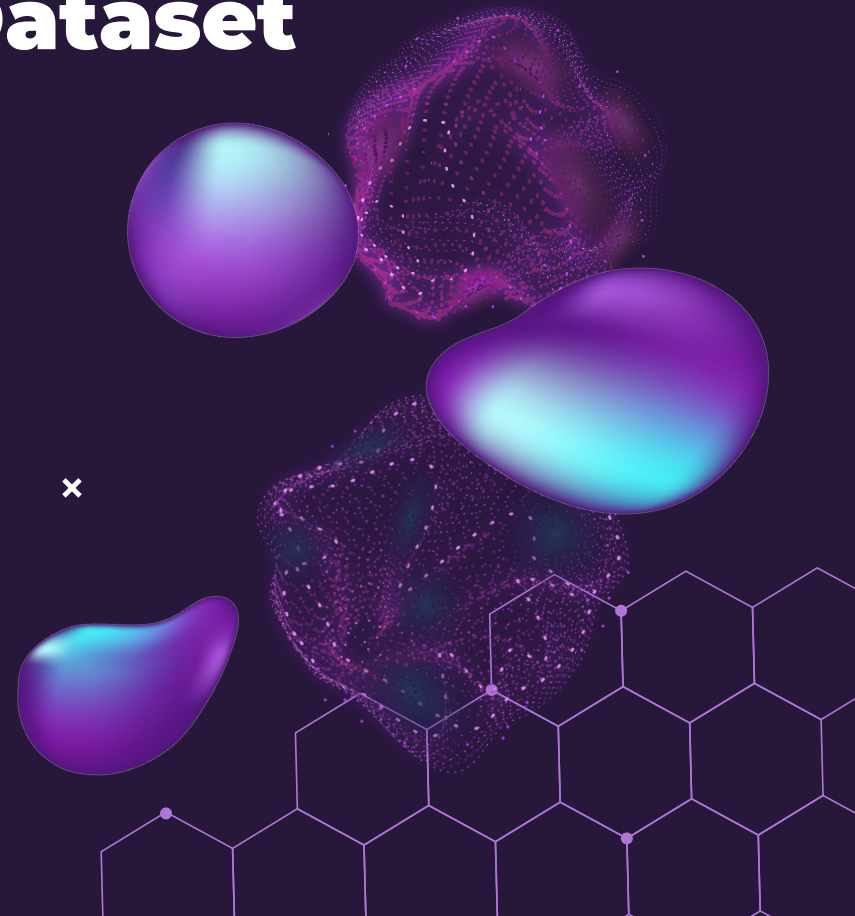


VGGFace Dataset

We use the models pretrained on the VGG Face Dataset.

VGG Face is a dataset of 2.6 million face images of 2,622 people, mostly public figures whose names were chosen based on popularity on the IMDB celebrity list.

Those names were fed into a Google Images search, and annotated by humans.



04

METHODOLOGY

Applying Transfer Learning to both our problems

Preprocessing

Rescaling, Normalization, Data Augmentation layers in constructed model

Feature Extraction

Freeze convolutional base, use as a feature extractor, train top-level classifier

Fine Tuning

Increase performance by training top-layer weights alongside classifier

PROJECT BREAKDOWN

STEP 1

Initial research on models and systems we could use to solve the problem

STEP 2

Data exploration and visualization

STEP 3

Creating child models from VGGFace2,FaceNet using train & valid set

STEP 4

Evaluate results of valid & test set

Loading and exploring the data

- We loaded our kaggle dataset, preprocessed the dataset, normalized the images and loaded the training, validation, and testing datasets.
- We are using a binary model with classes
  - 0 - image with a mask
  - 1 - image without a mask
- We are using the sparse categorical cross entropy loss in this dataset.

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json

!kaggle datasets download -d ashishjangra27/face-mask-12k-images-dataset
!unzip face-mask-12k-images-dataset.zip
```

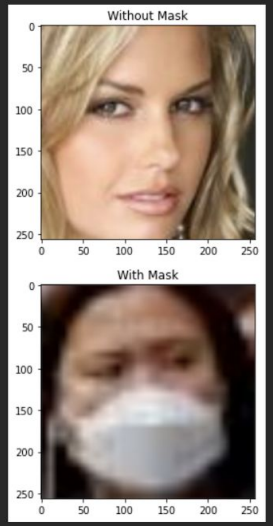
```
BATCH_SIZE =64
IMG_SIZE = (256, 256)
PATH = '/content/Face Mask Dataset/'

def loadSet(dir):
  dir = os.path.join(PATH, dir)
  ds= tf.keras.preprocessing.image_dataset_from_directory(
                      dir,
                      batch_size = BATCH_SIZE,
                      image_size=IMG_SIZE,
                      label_mode='binary',
                      #Sparse_categorical_crossentropy_loss to be used
                      class_names = ['WithMask','WithoutMask'],
                      seed=123,
                      shuffle=True)
  return ds

def normalize_image(image, label):
    return tf.cast(image, tf.float32) / 255., label

train_ds= loadSet('Train').map(normalize_image)
valid_ds= loadSet('Validation').map(normalize_image)
test_ds = loadSet('Test').map(normalize_image)
```

```
Found 10000 files belonging to 2 classes.
Found 800 files belonging to 2 classes.
Found 992 files belonging to 2 classes.
```

Data visualization

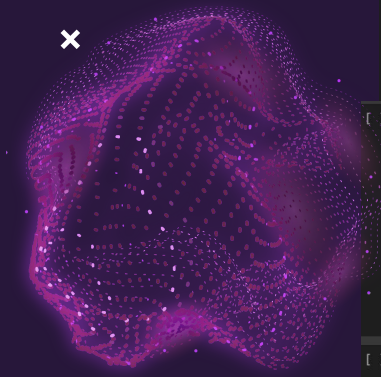
This is an example output for an image without and with a mask.

```
# data visualization
title_label = {1: "Without Mask", 0: "With Mask"}
for images, labels in train_ds.take(1):
    for i in np.arange(3):
        plt.title(title_label[int(labels[i])] )
        plt.imshow(images[i])
        plt.show()
    break
```



Loading the model

The untrained model has only 50% accuracy.



```
from keras_vggface.vggface import VGGFace

base_model = VGGFace(include_top=False,
                     input_shape=(256,256,3),
                     pooling='avg',weights='vggface')
base_model.trainable=False
```

```
[ ] #Add classification head

  data_rescale = tf.keras.layers.Rescaling(1./255.)

  data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.2)
  ])

  inputs = tf.keras.Input(shape=(256, 256, 3))
  x = data_rescale(inputs)
  x = data_augmentation(x)
  x = base_model(x, training=False)
  x = tf.keras.layers.Dropout(0.2)(x)
  outputs = tf.keras.layers.Dense(2,activation='sigmoid')(x)
  model = tf.keras.Model(inputs, outputs)
```

```
[ ] #Compile Model
  base_learning_rate = 0.001
  model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
                loss=tf.losses.SparseCategoricalCrossentropy(),
                metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3)]     0

 rescaling (Rescaling)       (None, 256, 256, 3)       0

 sequential (Sequential)     (None, 256, 256, 3)       0

 vggface_vgg16 (Functional)  (None, 512)               14714688

 dropout (Dropout)           (None, 512)               0

 dense (Dense)               (None, 2)                 1026

=================================================================
Total params: 14,715,714
Trainable params: 1,026
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
[ ] initial_epochs = 5
  loss0, accuracy0 = model.evaluate(valid_ds)
```

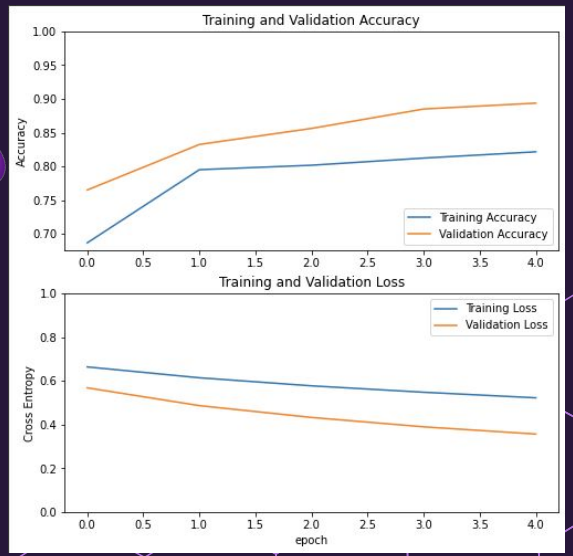
```
13/13 [==============================] - 21s 572ms/step - loss: 0.6974 - accuracy: 0.5000
```

Training the model

After training the model our accuracy is at 0.89.

```
[ ] history = model.fit(train_ds,
                        epochs=initial_epochs,
                        validation_data=valid_ds)
```

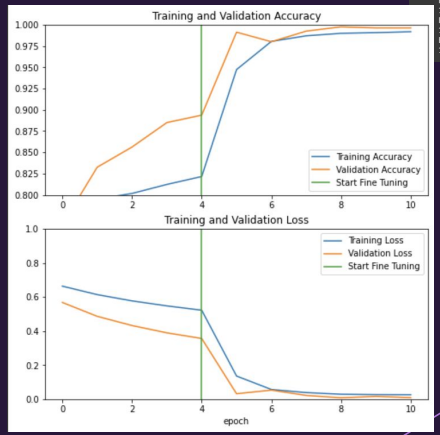
```
Epoch 1/5
157/157 [==============================] - 85s 529ms/step - loss: 0.6638 - accuracy: 0.6867 - val_loss: 0.5681 - val_accuracy: 0.7650
Epoch 2/5
157/157 [==============================] - 82s 521ms/step - loss: 0.6141 - accuracy: 0.7951 - val_loss: 0.4866 - val_accuracy: 0.8325
Epoch 3/5
157/157 [==============================] - 82s 522ms/step - loss: 0.5772 - accuracy: 0.8017 - val_loss: 0.4325 - val_accuracy: 0.8562
Epoch 4/5
157/157 [==============================] - 82s 523ms/step - loss: 0.5478 - accuracy: 0.8123 - val_loss: 0.3895 - val_accuracy: 0.8850
Epoch 5/5
157/157 [==============================] - 82s 522ms/step - loss: 0.5224 - accuracy: 0.8216 - val_loss: 0.3564 - val_accuracy: 0.8938
```



Fine tuning the model

After fine tuning the child model of VGGFace2 accuracy improves to 0.99.

```
[ ] base_model.trainable = True
    print("Number of layers in the base model: ", len(base_model.layers))

    Number of layers in the base model:  20

[ ] fine_tune_at = len(base_model.layers) - 5
    # Freeze all layers before `fine_tune_at` layer

    for layer in base_model.layers[:fine_tune_at]:
      layer.trainable = False

    base_learning_rate = 0.001
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.1*base_learning_rate),
                  loss=tf.losses.SparseCategoricalCrossentropy(),
                  metrics=['accuracy'])

    model.summary()

    Model: "model"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     input_2 (InputLayer)        [(None, 256, 256, 3)]     0

     rescaling (Rescaling)       (None, 256, 256, 3)       0

     sequential (Sequential)     (None, 256, 256, 3)       0

     vggface_vgg16 (Functional)  (None, 512)               14714688

     dropout (Dropout)           (None, 512)               0

     dense (Dense)               (None, 2)                 1026

    =================================================================
    Total params: 14,715,714
    Trainable params: 7,080,450
    Non-trainable params: 7,635,264
    _________________________________________________________________

    fine_tune_epochs = 5
    total_epochs =  initial_epochs + fine_tune_epochs

    history_fine = model.fit(train_ds,
                             epochs=total_epochs,
                             initial_epoch=history.epoch[-1],
                             validation_data=valid_ds)

    Epoch 5/10
    157/157 [==============================] - 91s 567ms/step - loss: 0.1357 - accuracy: 0.9473 - val_loss: 0.0316 - val_accuracy: 0.9912
    Epoch 6/10
    157/157 [==============================] - 89s 565ms/step - loss: 0.0565 - accuracy: 0.9804 - val_loss: 0.0529 - val_accuracy: 0.9800
    Epoch 7/10
    157/157 [==============================] - 88s 564ms/step - loss: 0.0384 - accuracy: 0.9870 - val_loss: 0.0216 - val_accuracy: 0.9925
    Epoch 8/10
    157/157 [==============================] - 89s 564ms/step - loss: 0.0286 - accuracy: 0.9899 - val_loss: 0.0079 - val_accuracy: 0.9975
    Epoch 9/10
    157/157 [==============================] - 89s 564ms/step - loss: 0.0265 - accuracy: 0.9907 - val_loss: 0.0159 - val_accuracy: 0.9962
    Epoch 10/10
    157/157 [==============================] - 89s 564ms/step - loss: 0.0258 - accuracy: 0.9918 - val_loss: 0.0087 - val_accuracy: 0.9962
```



Testing the model

Finally, we tested the model on our testing data set.
Our test accuracy is at 99.89%
Our test loss is 0.00878

```
loss1, accuracy1 = model.evaluate(test_ds)
```

```
16/16 [==============================] - 3s 175ms/step - loss: 0.0088 - accuracy: 0.9990
```

```
print("Test accuracy: ",accuracy1)
print("Test loss :",loss1)
```

```
Test accuracy:  0.9989919066429138
Test loss : 0.008786004967987537
```

05.1
RESULTS

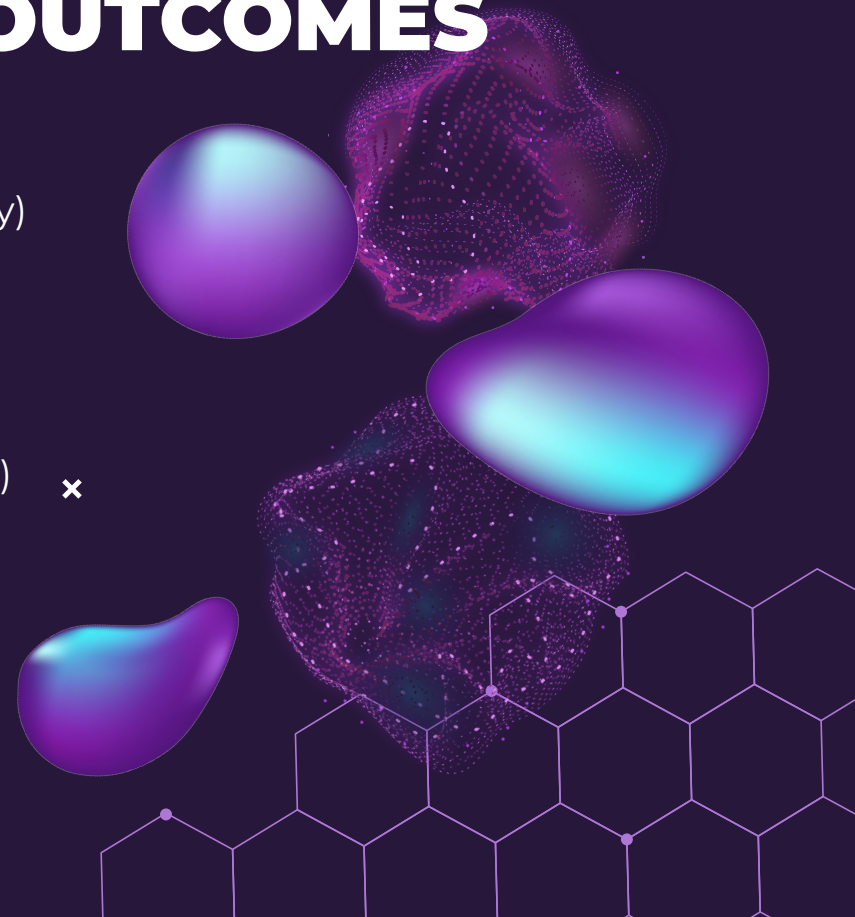
RESULTS AND OUTCOMES

Child model of VGGFace (n.b. validation accuracy)

- Pre-TransferLearn Accuracy was ~50%
- Post-FeatExtraction Accuracy was 89%
- Post-Finetuning Accuracy was 99.62%

Child model of FaceNet (n.b. validation accuracy)

- Pre-TransferLearn Accuracy was ~14%
- Post-FeatExtraction Accuracy was 32%
- Post-Finetuning Accuracy was 63%



05.2

POSSIBLE IMPROVEMENTS

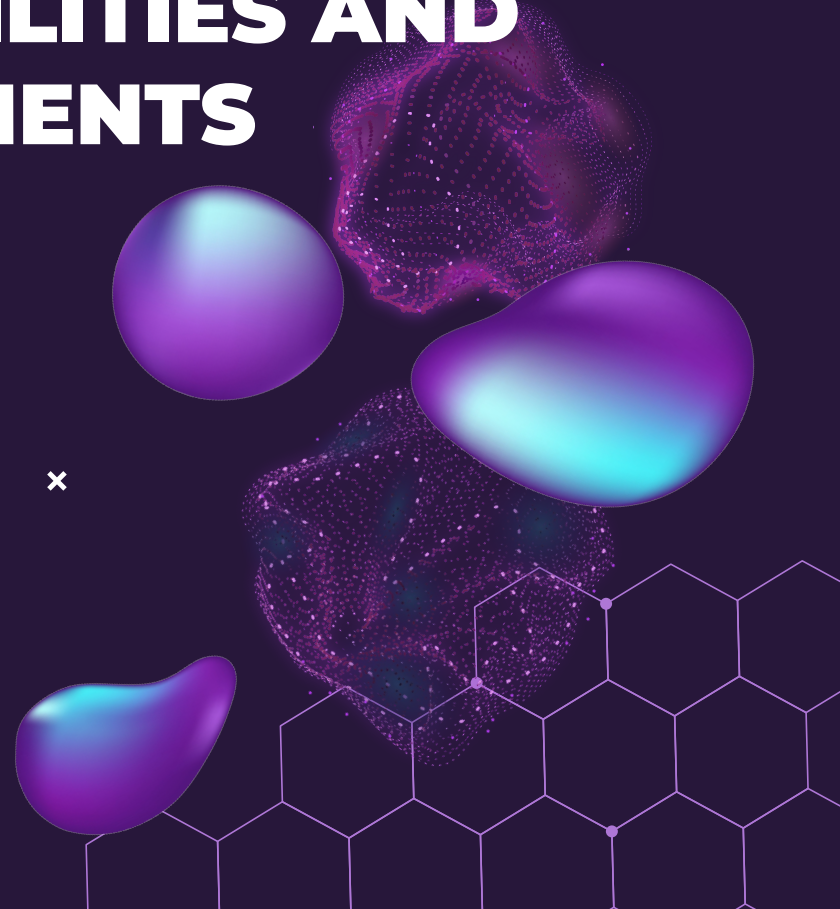
FUTURE POSSIBILITIES AND IMPROVEMENTS

Child model of VGG Face2

- Larger, more diverse, better image quality dataset
- Illumination and pose data augmentation
- Mask object localization

Child model of FaceNet

- Lost information because of grayscale?
- More diverse dataset
- Model Explainability using tf_explain
- Illumination and pose data augmentation



THANK YOU !

DO YOU HAVE ANY QUESTIONS?