

face

October 19, 2024

1 setup

```
[2]: !pip install opencv-python matplotlib
```

```
Requirement already satisfied: opencv-python in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (4.10.0.84)
Requirement already satisfied: matplotlib in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (3.9.2)
Requirement already satisfied: numpy>=1.17.0 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from opencv-python)
(1.26.4)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(1.3.0)
Requirement already satisfied: cyclor>=0.10 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(1.4.7)
Requirement already satisfied: packaging>=20.0 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(24.1)
Requirement already satisfied: pillow>=8 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: importlib-resources>=3.2.0 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from matplotlib)
(6.4.5)
```

Requirement already satisfied: zipp>=3.1.0 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from importlib-
resources>=3.2.0->matplotlib) (3.20.2)
Requirement already satisfied: six>=1.5 in
c:\users\maxco\pycharmprojects\face\venv\lib\site-packages (from python-
dateutil>=2.7->matplotlib) (1.16.0)

[notice] A new release of pip is available: 23.2.1 -> 24.2

[notice] To update, run: python.exe -m pip install --upgrade pip

```
[8]: import cv2
import os
import random
import numpy as np
from matplotlib import pyplot as plt
```

```
[10]: from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Layer, Dense, Flatten
import tensorflow as tf
```

```
[11]: gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

```
[12]: POSITIVE_PATH = os.path.join('data', 'positive')
NEGATIVE_PATH = os.path.join('data', 'negative')
ANCHOR_PATH = os.path.join('data', 'anchor')
```

```
[8]: os.makedirs(POSITIVE_PATH)
os.makedirs(NEGATIVE_PATH)
os.makedirs(ANCHOR_PATH)
```

```
[12]: !tar -xf lfw.tgz
```

```
[13]: for directory in os.listdir('lfw'):
    for file in os.listdir(os.path.join('lfw', directory)):
        Ex_Path = os.path.join('lfw', directory, file)
        NEW_PATH = os.path.join(NEGATIVE_PATH, file)
        os.replace(Ex_Path, NEW_PATH)
```

```
[5]: import uuid
```

```
[7]: cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
```

```

frame = frame[120:120+250,200:200+250,:]

if cv2.waitKey(1) & 0xFF == ord('p'):
    imgname = os.path.join(POSITIVE_PATH, '{}.jpg'.format(uuid.uuid1()))
    cv2.imwrite(imgname, frame)

if cv2.waitKey(1) & 0xFF == ord('a'):
    imgname = os.path.join(ANCHOR_PATH, '{}.jpg'.format(uuid.uuid1()))
    cv2.imwrite(imgname, frame)

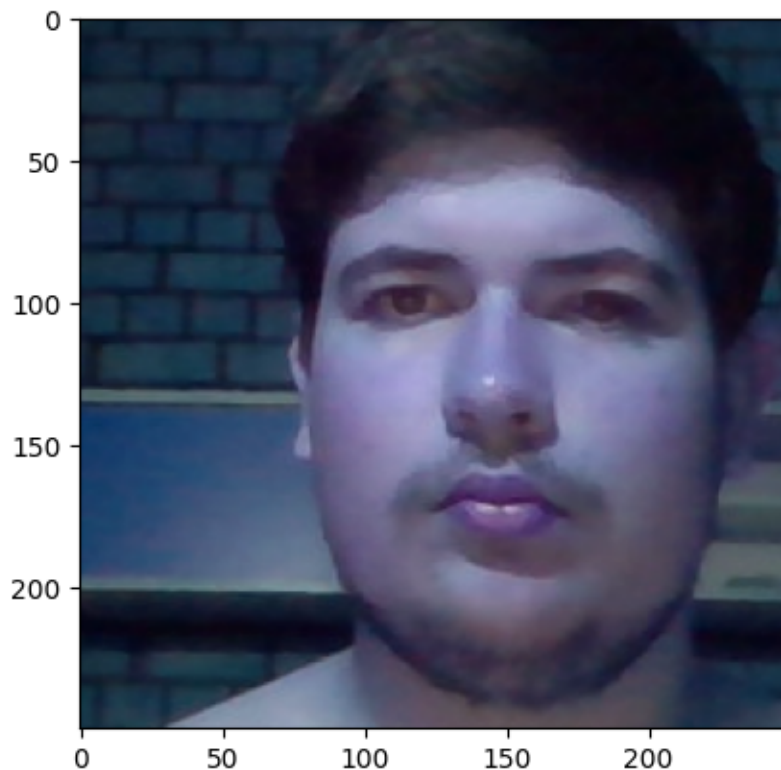
cv2.imshow('Image Collection', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()

```

```
[26]: plt.imshow(frame)
```

```
[26]: <matplotlib.image.AxesImage at 0x2b0bb2484f0>
```



```
[13]: anchor = tf.data.Dataset.list_files(os.path.join(ANCHOR_PATH + '*.jpg')).
      ↪take(300)
      positive = tf.data.Dataset.list_files(os.path.join(POSITIVE_PATH + '*.jpg')).
      ↪take(300)
      negative = tf.data.Dataset.list_files(os.path.join(NEGATIVE_PATH + '*.jpg')).
      ↪take(300)
```

```
[14]: dir_test = anchor.as_numpy_iterator()
```

```
[15]: print(dir_test.next())
```

```
b'data\\anchor\\ca20f9b5-8db4-11ef-8919-dcfb48a9057a.jpg'
```

```
[18]: def preprocess(file_path):
      byte_img = tf.io.read_file(file_path)
      img = tf.io.decode_jpeg(byte_img)
      img = tf.image.resize(img, (100,100))
      img = img / 255.0
      return img
```

```
[22]: img = preprocess('data\\anchor\\ca20f9b5-8db4-11ef-8919-dcfb48a9057a.jpg')
      img.numpy().max()
```

```
[22]: 0.9279412
```

```
[23]: tf.ones(len(anchor))
```

```
[23]: <tf.Tensor: shape=(62,), dtype=float32, numpy=
      array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
             1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
             1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
             1., 1., 1., 1., 1., 1., 1., 1., 1., 1.], dtype=float32)>
```

```
[26]: positives = tf.data.Dataset.zip((anchor, positive, tf.data.Dataset.
      ↪from_tensor_slices(tf.ones(len(anchor)))))

      negatives = tf.data.Dataset.zip((anchor, negative, tf.data.Dataset.
      ↪from_tensor_slices(tf.zeros(len(anchor)))))

      data = positives.concatenate(negatives)
```

```
[27]: samples = data.as_numpy_iterator()
```

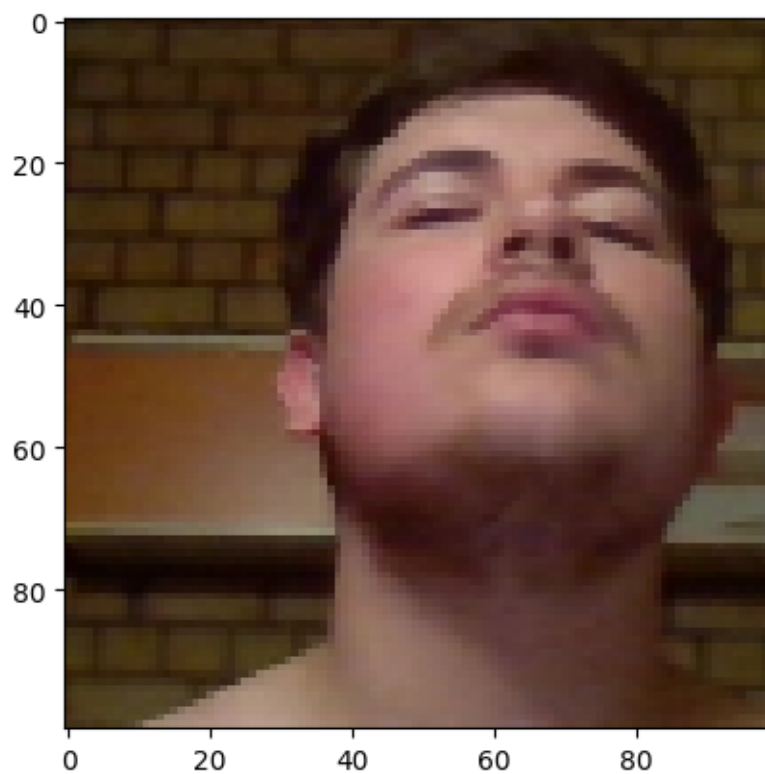
```
[65]: example = samples.next()
```

```
[126]: def preprocess_twin(input_img, valid_img, label):
      return(preprocess(input_img), preprocess(valid_img), label)
```

```
[68]: res = preprocess_twin(*example)
```

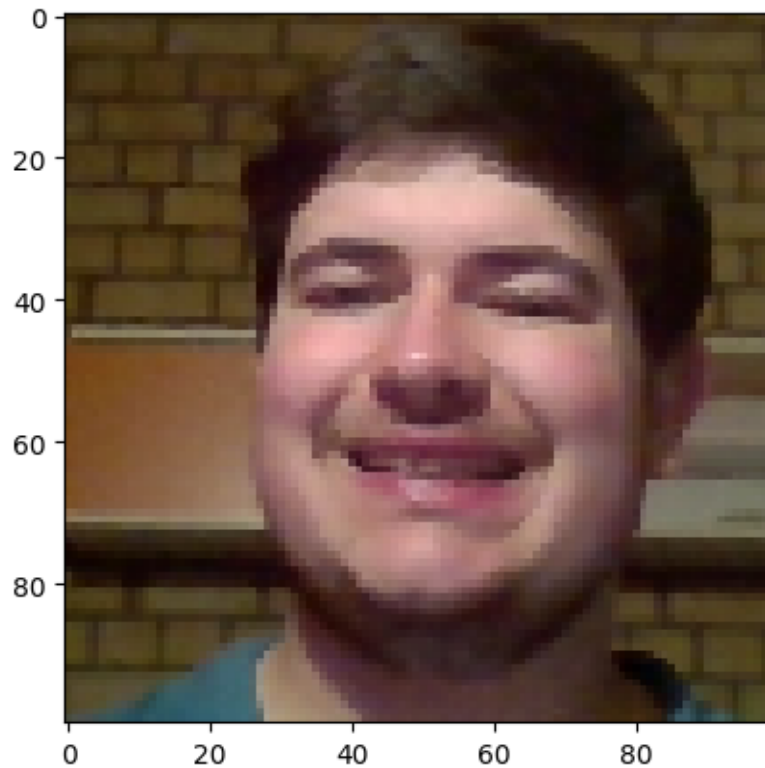
```
[76]: plt.imshow(res[0])
```

```
[76]: <matplotlib.image.AxesImage at 0x203c212f430>
```



```
[77]: plt.imshow(res[1])
```

```
[77]: <matplotlib.image.AxesImage at 0x203c21b82b0>
```



```
[78]: res[2]
```

```
[78]: 1.0
```

```
[80]: data = data.map(preprocess_twin)
      data = data.cache()
      data = data.shuffle(buffer_size=1024)
```

```
[81]: data
```

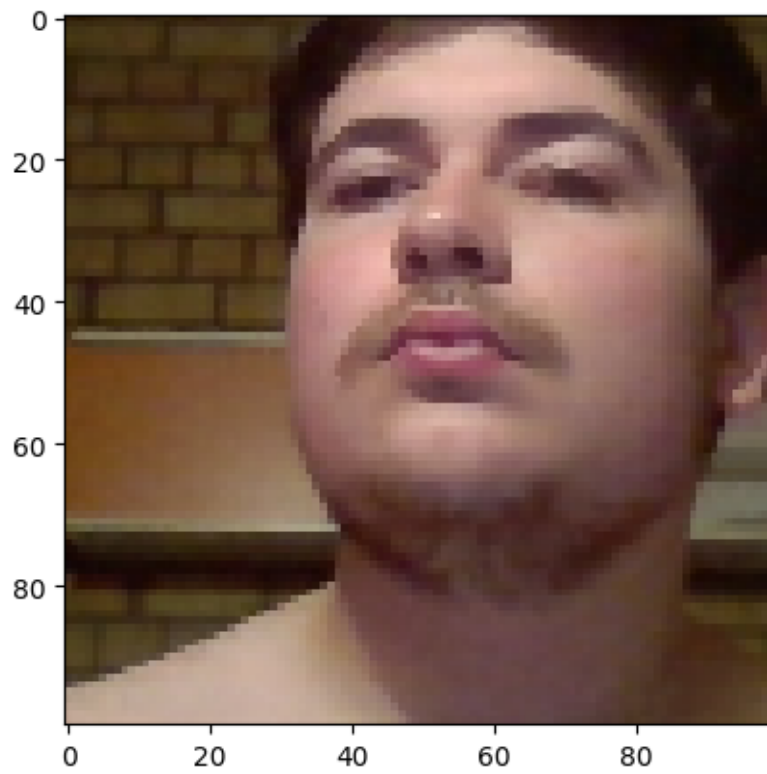
```
[81]: <_ShuffleDataset element_spec=(TensorSpec(shape=(100, 100, None),
dtype=tf.float32, name=None), TensorSpec(shape=(100, 100, None),
dtype=tf.float32, name=None), TensorSpec(shape=(), dtype=tf.float32,
name=None))>
```

```
[85]: samples = data.as_numpy_iterator()
```

```
[92]: samp = samples.next()
```

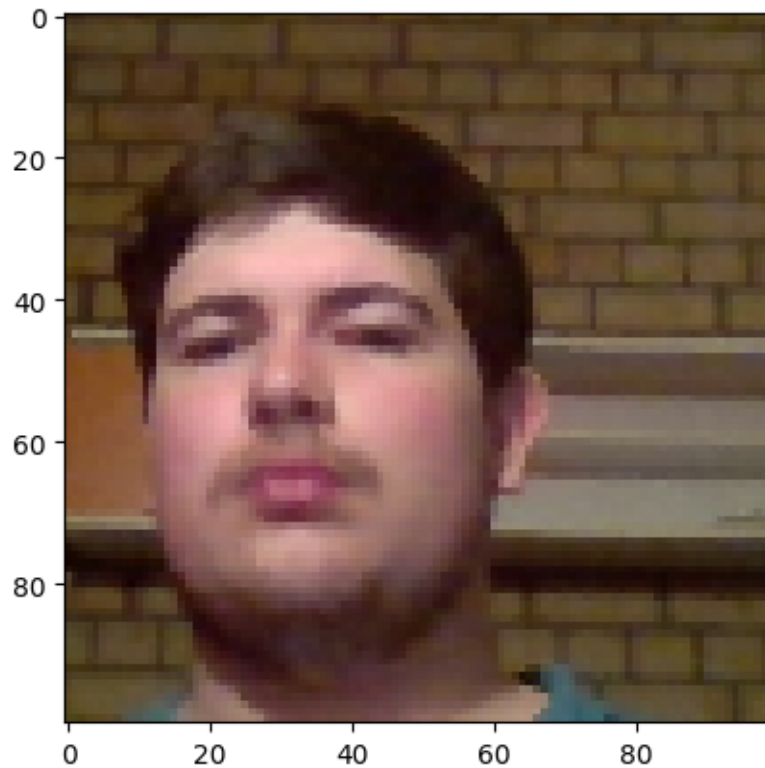
```
[93]: plt.imshow(samp[0])
```

```
[93]: <matplotlib.image.AxesImage at 0x203c25c70a0>
```



```
[94]: plt.imshow(samp[1])
```

```
[94]: <matplotlib.image.AxesImage at 0x203c2659e80>
```



```
[95]: samp[2]
```

```
[95]: 1.0
```

```
[125]: train_data = data.take(round(len(data)*.7))  
train_data = train_data.batch(16)  
train_data = train_data.prefetch(8)
```

```
[98]: train_samples = train_data.as_numpy_iterator()
```

```
[99]: train_samples= train_samples.next()
```

```
[124]: len(train_samples[0])
```

```
[124]: 16
```

```
[123]: test_data = data.skip(round(len(data)*.7))  
test_data = test_data.take(round(len(data)*.3))  
test_data = test_data.batch(16)  
test_data = test_data.prefetch(8)
```

```
[173]: def make_embedding():
inp = Input(shape=(100,100,3), name='input_image')
c1 = Conv2D(64, (10,10), activation='relu')(inp)
m1 = MaxPooling2D(64,(2,2),padding='same')(c1)
c2 = Conv2D(128, (7,7), activation='relu')(m1)
m2 = MaxPooling2D(64,(2,2),padding='same')(c2)
c3 = Conv2D(128, (4,4), activation='relu')(m2)
m3 = MaxPooling2D(64,(2,2),padding='same')(c3)
c4 = Conv2D(256, (4,4), activation='relu')(m3)
f1 = Flatten()(c4)
d1 = Dense(4096, activation='sigmoid')(f1)

return Model(inputs=inp, outputs=d1,name='embedding')
```

```
[174]: embedding = make_embedding()
```

```
[151]: embedding.summary()
```

Model: "embedding"

Layer (type)	Output Shape	Param #
input_image (InputLayer)	(None, 100, 100, 3)	0
conv2d_12 (Conv2D)	(None, 91, 91, 64)	19,264
max_pooling2d_9 (MaxPooling2D)	(None, 46, 46, 64)	0
conv2d_13 (Conv2D)	(None, 40, 40, 128)	401,536
max_pooling2d_10 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_14 (Conv2D)	(None, 17, 17, 128)	262,272
max_pooling2d_11 (MaxPooling2D)	(None, 9, 9, 128)	0
conv2d_15 (Conv2D)	(None, 6, 6, 256)	524,544
flatten_3 (Flatten)	(None, 9216)	0
dense_3 (Dense)	(None, 4096)	37,752,832

Total params: 38,960,448 (148.62 MB)

Trainable params: 38,960,448 (148.62 MB)

Non-trainable params: 0 (0.00 B)

```
[152]: class L1Dist(Layer):
        def __init__(self, **kwargs):
            super().__init__()

        def call(self, input_embedding, validation_embedding):
            return tf.math.abs(input_embedding - validation_embedding)
```

```
[153]: input_image = Input(shape=(100,100,3), name='input_img')
        validation_image = Input(shape=(100,100,3), name='validation_img')
```

```
[154]: inp_embedding = embedding(input_image)
        val_embedding = embedding(validation_image)
```

```
[155]: embedding(input_image)
```

```
[155]: <KerasTensor shape=(None, 4096), dtype=float32, sparse=False,
        name=keras_tensor_52>
```

```
[156]: siamese_layer = L1Dist()
```

```
[158]: distances = siamese_layer(inp_embedding, val_embedding)
```

```
[175]: def make_siamese_model():
        input_image = Input(shape=(100,100,3), name='input_img')
        validation_image = Input(shape=(100,100,3), name='validation_img')

        siamese_layer = L1Dist()
        siamese_layer._name = 'distance'
        distances = siamese_layer(embedding(input_image),
        ↪embedding(validation_image))

        classifier = Dense(1, activation='sigmoid')(distances)

        return Model(inputs=[input_image, validation_image], outputs=classifier,
        ↪name='siameseNetwork')
```

```
[159]: classifier = Dense(1, activation='sigmoid')(distances)
```

```
[160]: classifier
```

```
[160]: <KerasTensor shape=(None, 1), dtype=float32, sparse=False, name=keras_tensor_55>
```

```
[161]: siamese_network = Model(inputs=[input_image, validation_image],  
    ↪ outputs=classifier, name='siameseNetwork')
```

```
[162]: siamese_network.summary()
```

Model: "siameseNetwork"

Layer (type)	Output Shape	Param #	Connected to
input_img (InputLayer)	(None, 100, 100, 3)	0	-
validation_img (InputLayer)	(None, 100, 100, 3)	0	-
embedding (Functional)	(None, 4096)	38,960,448	input_img[0][0], validation_img[0]...
l1_dist_6 (L1Dist)	(None, 4096)	0	embedding[0][0], embedding[1][0]
dense_4 (Dense)	(None, 1)	4,097	l1_dist_6[1][0]

Total params: 38,964,545 (148.64 MB)

Trainable params: 38,964,545 (148.64 MB)

Non-trainable params: 0 (0.00 B)

```
[177]: make_siamese_model().summary()
```

Model: "siameseNetwork"

Layer (type)	Output Shape	Param #	Connected to
input_img (InputLayer)	(None, 100, 100, 3)	0	-
validation_img (InputLayer)	(None, 100, 100, 3)	0	-

embedding (Functional)	(None , 4096)	38,960,448	input_img[0][0], validation_img[0...
l1_dist_14 (L1Dist)	(None , 4096)	0	embedding[2][0], embedding[3][0]
dense_7 (Dense)	(None , 1)	4,097	l1_dist_14[0][0]

Total params: 38,964,545 (148.64 MB)

Trainable params: 38,964,545 (148.64 MB)

Non-trainable params: 0 (0.00 B)

[177]:

[]: