CIA(b)

Prashant[22122036]

Problem Statement : *The process of learning good features for machine learning applications can be very computationally expensive and may prove difficult in cases where little data is available. A prototypical example of this is the one-shot learning setting,*
in which we must correctly make predictions given only a single example of each new
class. In this paper, we explore a method for learning siamese neural networks which employ a unique structure to naturally rank similarity between inputs. Once a network has been tuned, we can then capitalize on powerful discriminative features to generalize the predictive power of the network not just to new data, but to entirely new classes from unknown distributions. Using a convolutional architecture, we are able to achieve strong results which exceed those of other deep learning models with near state-of-the-art performance on one-shot classification tasks.

1. Setup

1.1 Install Dependencies

In [90]:

```
!pip install tensorflow==2.8.0 tensorflow-gpu==2.8.0 opencv-python
matplotlib
Requirement already satisfied: tensorflow==2.8.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(2.8.0)
Requirement already satisfied: tensorflow-gpu==2.8.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(2.8.0)
Requirement already satisfied: opencv-python in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(4.5.5.64)
Requirement already satisfied: matplotlib in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(3.5.1)
Requirement already satisfied: gast>=0.2.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (0.5.3)
Requirement already satisfied: absl-py>=0.4.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.0.0)
Requirement already satisfied: astunparse>=1.6.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.6.3)
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (2.8.0)
```

```
Requirement already satisfied: tensorboard<2.9,>=2.8 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (2.8.0)
Requirement already satisfied: libclang>=9.0.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (13.0.0)
Requirement already satisfied: setuptools in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (57.4.0)
Requirement already satisfied: wrapt>=1.11.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.14.0)
Requirement already satisfied: h5py>=2.9.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (3.6.0)
Requirement already satisfied: termcolor>=1.1.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.1.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (0.24.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.1.2)
Requirement already satisfied: six>=1.12.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.16.0)
Requirement already satisfied: flatbuffers>=1.12 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (2.0)
Requirement already satisfied: numpy>=1.20 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.22.3)
Requirement already satisfied: protobuf>=3.9.2 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (3.20.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (1.44.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (4.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (3.3.0)
Requirement already satisfied: google-pasta>=0.1.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (0.2.0)
```

```
Requirement already satisfied: tf-estimator-nightly==2.8.0.dev2021122109 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorflow==2.8.0) (2.8.0.dev2021122109)
Requirement already satisfied: pyparsing>=2.2.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (3.0.8)
Requirement already satisfied: packaging>=20.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (1.4.2)
Requirement already satisfied: pillow>=6.2.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (9.1.0)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (4.32.0)
Requirement already satisfied: cycler>=0.10 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (0.11.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from astunparse>=1.6.0->tensorflow==2.8.0) (0.37.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorboard<2.9,>=2.8->tensorflow==2.8.0) (0.6.1)
Requirement already satisfied: werkzeug>=0.11.15 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorboard<2.9,>=2.8->tensorflow==2.8.0) (2.1.1)
Requirement already satisfied: requests<3,>=2.21.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorboard<2.9,>=2.8->tensorflow==2.8.0) (2.27.1)
Requirement already satisfied: markdown>=2.6.8 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorboard<2.9,>=2.8->tensorflow==2.8.0) (3.3.6)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorboard<2.9,>=2.8->tensorflow==2.8.0) (0.4.6)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorboard<2.9,>=2.8->tensorflow==2.8.0) (1.8.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from tensorboard<2.9,>=2.8->tensorflow==2.8.0) (2.6.3)
```

```
Requirement already satisfied: pyasn1-modules>=0.2.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
(0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
(4.8)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
(5.0.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8-
>tensorflow==2.8.0) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from markdown>=2.6.8->tensorboard<2.9,>=2.8->tensorflow==2.8.0) (4.11.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
(2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
(2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0)
(1.26.9)
Requirement already satisfied: zipp>=0.5 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.9,>=2.8-
>tensorflow==2.8.0) (3.8.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8-
>tensorflow==2.8.0) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in
c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
(from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1-
>tensorboard<2.9,>=2.8->tensorflow==2.8.0) (3.2.0)
```

1.2 Import Dependencies

In [91]:

```
# Import standard dependencies
```

```
import cv2
import os
import random
import numpy as np
from matplotlib import pyplot as plt
```

In [92]:

```
# Import tensorflow dependencies - Functional API
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Layer, Conv2D, Dense, MaxPooling2D,
Input, Flatten
import tensorflow as tf
```

1.3 Set GPU Growth

In [4]:

```
# Avoid OOM errors by setting GPU Memory Consumption Growth
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

1.4 Create Folder Structures

In [5]:

```
# Setup paths
POS_PATH = os.path.join('data', 'positive')
NEG_PATH = os.path.join('data', 'negative')
ANC_PATH = os.path.join('data', 'anchor')
```

In [6]:

```
# Make the directories
os.makedirs(POS_PATH)
os.makedirs(NEG_PATH)
os.makedirs(ANC_PATH)
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
Input In [6], in <cell line: 2>()
      1 # Make the directories
----> 2 os.makedirs(POS_PATH)
      3 os.makedirs(NEG_PATH)
      4 os.makedirs(ANC_PATH)

File ~\AppData\Local\Programs\Python\Python39\lib\os.py:225, in
makedirs(name, mode, exist_ok)
    223         return
    224 try:
--> 225     mkdir(name, mode)
    226 except OSError:
    227     # Cannot rely on checking for EEXIST, since the operating
system
    228     # could give priority to other errors like EACCES or EROFS
    229     if not exist_ok or not path.isdir(name):
```

```
FileExistsError: [WinError 183] Cannot create a file when that file already
exists: 'data\\positive'
```

1. Collect Positives and Anchors

In [7]:

```
# http://vis-www.cs.umass.edu/lfw/
```

In [ ]:

```
# Uncompress Tar GZ Labelled Faces in the Wild Dataset
!tar -xf lfw.tgz
```

In [ ]:

```
# Move LFW Images to the following repository data/negative
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(NEG_PATH, file)
        os.replace(EX_PATH, NEW_PATH)
```

2.2 Collect Positive and Anchor Classes

In [8]:

```
# Import uuid library to generate unique image names
import uuid
```

In [9]:

```
os.path.join(ANC_PATH, '{}.jpg'.format(uuid.uuid1()))
```

Out[9]:

```
'data\\anchor\\b1fc951c-bcb0-11ec-8860-c03eba3a1da6.jpg'
```

In [ ]:

```
# Establish a connection to the webcam
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    # Cut down frame to 250x250px
    frame = frame[120:120+250,200:200+250, :]

    # Collect anchors
    if cv2.waitKey(1) & 0XFF == ord('a'):
        # Create the unique file path
        imgname = os.path.join(ANC_PATH, '{}.jpg'.format(uuid.uuid1()))
        # Write out anchor image
        cv2.imwrite(imgname, frame)

    # Collect positives
    if cv2.waitKey(1) & 0XFF == ord('p'):
        # Create the unique file path
        imgname = os.path.join(POS_PATH, '{}.jpg'.format(uuid.uuid1()))
        # Write out positive image
        cv2.imwrite(imgname, frame)

    # Show image back to screen
    cv2.imshow('Image Collection', frame)
    # Breaking gracefully
    if cv2.waitKey(1) & 0XFF == ord('q'):
        break

# Release the webcam
cap.release()
# Close the image show frame
```

In [ ]:

```
plt.imshow(frame[120:120+250,200:200+250, :])
```

1. Load and Preprocess Images

3.1 Get Image Directories

In [10]:

```
anchor = tf.data.Dataset.list_files(ANC_PATH+'\*.jpg').take(300)
positive = tf.data.Dataset.list_files(POS_PATH+'\*.jpg').take(300)
negative = tf.data.Dataset.list_files(NEG_PATH+'\*.jpg').take(300)
```

In [68]:

```
dir_test = anchor.as_numpy_iterator()
```

In [12]:

```
print(dir_test.next())
```

```
b'data\\anchor\\d7862787-bc7d-11ec-979c-c03eba3a1da6.jpg'
```

One cool thing I noticed about training on pairs is that there are quadratically many possible pairs of images to train the model on, making it hard to overfit. Say we have C examples each of E classes. Since there are C·E images total, the total number of possible pairs is given by Npairs=(C·E2)=(C·E)!2!(C·E−2)! For omniglot with its 20 examples of 964 training classes, this leads to 185,849,560 possible pairs, which is huge! However, the siamese network needs examples of both same and different class pairs. There are E examples per class, so there will be (E2) pairs for every class, which means there are Nsame=(E2)·C possible pairs with the same class - 183,160 pairs for omniglot. Even though 183,160 example pairs is plenty, it's only a thousandth of the possible pairs, and the number of same-class pairs increases quadratically with E but only linearly with C. This is important because the siamese network should be given a 1:1 ratio of same-class and different-class pairs to train on - perhaps it implies that pairwise training is easier on datasets with lots of examples per class.

3.2 Preprocessing - Scale and Resize

In [2]:

```
def preprocess(file_path):

    # Read in image from file path
    byte_img = tf.io.read_file(file_path)
    # Load in the image
    img = tf.io.decode_jpeg(byte_img)

    # Preprocessing steps - resizing the image to be 100x100x3
    img = tf.image.resize(img, (100,100))
    # Scale image to be between 0 and 1
    img = img / 255.0

    # Return image
    return img
```

In [4]:

```
img = preprocess(b'data\\anchor\\ea60f2a8-bc7d-11ec-aadf-c03eba3a1da6.jpg')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [4], in <cell line: 1>()
----> 1 img = preprocess(b'data\\anchor\\ea60f2a8-bc7d-11ec-aadf-
c03eba3a1da6.jpg')

Input In [2], in preprocess(file_path)
      1 def preprocess(file_path):
      2 
      3     # Read in image from file path
----> 4     byte_img = tf.io.read_file(file_path)
      5     # Load in the image
      6     img = tf.io.decode_jpeg(byte_img)

NameError: name 'tf' is not defined
```

In [15]:

```
img.numpy().max()
```

Out[15]:

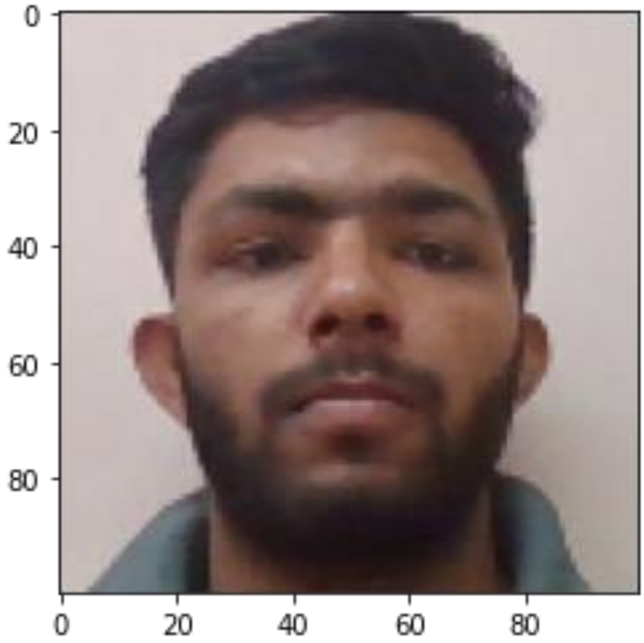
```
0.902451
```

In [16]:

```
plt.imshow(img)
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x1f6324722e0>
```



In [17]:

```
dataset.map(preprocess)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
Input In [17], in <cell line: 1>()
----> 1 dataset.map(preprocess)

NameError: name 'dataset' is not defined
```

3.3 Create Labelled Dataset

In [18]:

```
# (anchor, positive) => 1,1,1,1,1
# (anchor, negative) => 0,0,0,0,0
```

In [19]:

```
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)
```

In [20]:

```
samples = data.as_numpy_iterator()
```

In [21]:

```
exampple = samples.next()
```

In [22]:

```
exampple
```

Out[22]:

```
(b'data\\anchor\\05ea49ff-bc7e-11ec-aec7-c03eba3a1da6.jpg',
 b'data\\positive\\86594d73-bc7e-11ec-a952-c03eba3a1da6.jpg',
 1.0)
```

3.4 Build Train and Test Partition

In [23]:

```
def preprocess_twin(input_img, validation_img, label):
    return(preprocess(input_img), preprocess(validation_img), label)
```

In [24]:

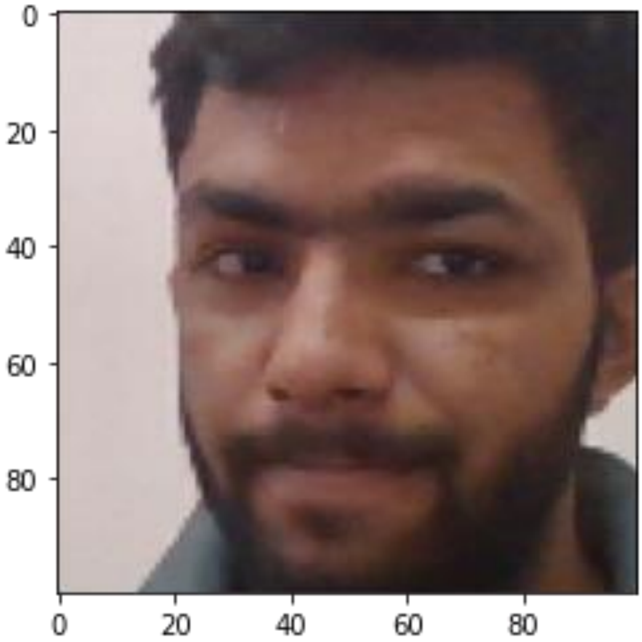
```
res = preprocess_twin(*exampple)
```

In [25]:

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

Out[25]:

```
<matplotlib.image.AxesImage at 0x1f634583c70>
```



In [26]:

```
res[2]
```

Out[26]:

```
1.0
```

In [27]:

```
# Build dataloader pipeline
data = data.map(preprocess_twin)
data = data.cache()
data = data.shuffle(buffer_size=1024)
```

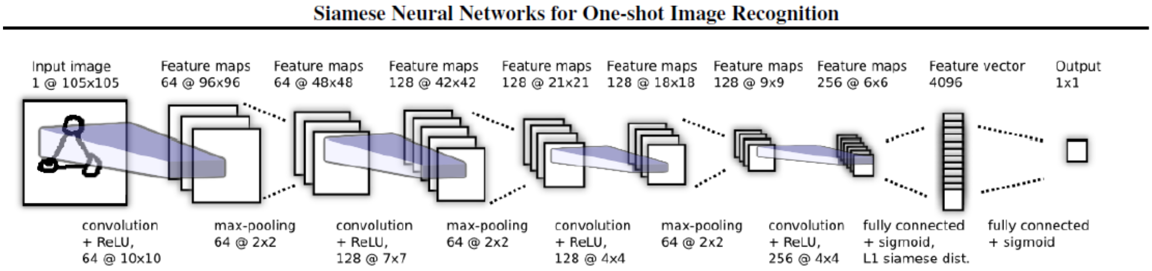
In [28]:

```
# Training partition
train_data = data.take(round(len(data)*.7))
train_data = train_data.batch(16)
train_data = train_data.prefetch(8)
```

In [29]:

```
# Testing partition
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)
```

ARCHITECTURE



1. Model Engineering

4.1 Build Embedding Layer

In [30]:

```
inp = Input(shape=(100,100,3), name='input_image')
```

In [31]:

```
c1 = Conv2D(64, (10,10), activation='relu')(inp)
```

In [32]:

```
m1 = MaxPooling2D(64, (2,2), padding='same')(c1)
```

In [33]:

```
c2 = Conv2D(128, (7,7), activation='relu')(m1)
m2 = MaxPooling2D(64, (2,2), padding='same')(c2)
```

In [34]:

```
c3 = Conv2D(128, (4,4), activation='relu')(m2)
m3 = MaxPooling2D(64, (2,2), padding='same')(c3)
```

In [35]:

```
c4 = Conv2D(256, (4,4), activation='relu')(m3)
f1 = Flatten()(c4)
d1 = Dense(4096, activation='sigmoid')(f1)
```

In [36]:

```
mod = Model(inputs=[inp], outputs=[d1], name='embedding')
```

In [37]:

```
mod.summary()
Model: "embedding"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_image (InputLayer)    [(None, 100, 100, 3)]     0

 conv2d (Conv2D)             (None, 91, 91, 64)        19264

 max_pooling2d (MaxPooling2D  (None, 46, 46, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 40, 40, 128)       401536
 max_pooling2d_1 (MaxPooling  (None, 20, 20, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 17, 17, 128)       262272

 max_pooling2d_2 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 conv2d_3 (Conv2D)           (None, 6, 6, 256)         524544

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 4096)              37752832

=================================================================
Total params: 38,960,448
Trainable params: 38,960,448
Non-trainable params: 0
_________________________________________________________________
```

In [38]:

```
def make_embedding():
    inp = Input(shape=(100,100,3), name='input_image')

    # First block
    c1 = Conv2D(64, (10,10), activation='relu')(inp)
    m1 = MaxPooling2D(64, (2,2), padding='same')(c1)

    # Second block
    c2 = Conv2D(128, (7,7), activation='relu')(m1)
    m2 = MaxPooling2D(64, (2,2), padding='same')(c2)

    # Third block
    c3 = Conv2D(128, (4,4), activation='relu')(m2)
    m3 = MaxPooling2D(64, (2,2), padding='same')(c3)

    # Final embedding block
    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')
```

In [39]:

```
embedding = make_embedding()
```

In [40]:

```
embedding.summary()
Model: "embedding"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 input_image (InputLayer)    [(None, 100, 100, 3)]     0

 conv2d_4 (Conv2D)           (None, 91, 91, 64)        19264

 max_pooling2d_3 (MaxPooling  (None, 46, 46, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 40, 40, 128)       401536

 max_pooling2d_4 (MaxPooling  (None, 20, 20, 128)      0
 2D)

 conv2d_6 (Conv2D)           (None, 17, 17, 128)       262272

 max_pooling2d_5 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 conv2d_7 (Conv2D)           (None, 6, 6, 256)         524544

 flatten_1 (Flatten)         (None, 9216)              0

 dense_1 (Dense)             (None, 4096)              37752832

=================================================================
Total params: 38,960,448
Trainable params: 38,960,448
Non-trainable params: 0
_________________________________________________________________
```

4.2 Build Distance Layer

In [41]:

```
# Siamese L1 Distance class
class L1Dist(Layer):

    # Init method - inheritance
    def __init__(self, **kwargs):
        super().__init__()

    # Magic happens here - similarity calculation
    def call(self, input_embedding, validation_embedding):
        return tf.math.abs(input_embedding - validation_embedding)
```

In [42]:

```
l1 = L1Dist()
```

In [43]:

```
l1(anchor_embedding, validation_embedding)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [43], in <cell line: 1>()
```

```
----> 1 l1(anchor_embedding, validation_embedding)

NameError: name 'anchor_embedding' is not defined
```

4.3 Make Siamese Model

In [44]:

```
input_image = Input(name='input_img', shape=(100,100,3))
validation_image = Input(name='validation_img', shape=(100,100,3))
```

In [45]:

```
inp_embedding = embedding(input_image)
val_embedding = embedding(validation_image)
```

In [46]:

```
siamese_layer = L1Dist()
```

In [47]:

```
distances = siamese_layer(inp_embedding, val_embedding)
```

In [48]:

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

In [49]:

```
classifier
```

Out[49]:

```
<KerasTensor: shape=(None, 1) dtype=float32 (created by layer 'dense_2')>
```

In [50]:

```
siamese_network = Model(inputs=[input_image, validation_image],
outputs=classifier, name='SiameseNetwork')
```

In [51]:

```
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)         38960448
['input_img[0][0]',

'validation_img[0][0]']

 l1_dist_1 (L1Dist)          (None, 4096)         0
['embedding[0][0]',

'embedding[1][0]']

 dense_2 (Dense)             (None, 1)            4097
['l1_dist_1[0][0]']
```

```
===========================================================================
=======================
Total params: 38,964,545
Trainable params: 38,964,545
Non-trainable params: 0
___________________________________________________________________________
_______________________
```

In [52]:

```
def make_siamese_model():

    # Anchor image input in the network
    input_image = Input(name='input_img', shape=(100,100,3))

    # Validation image in the network
    validation_image = Input(name='validation_img', shape=(100,100,3))

    # Combine siamese distance components
    siamese_layer = L1Dist()
    siamese_layer._name = 'distance'
    distances = siamese_layer(embedding(input_image),
embedding(validation_image))

    # Classification layer
    classifier = Dense(1, activation='sigmoid')(distances)

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

In [53]:

```
siamese_model = make_siamese_model()
```

In [54]:

```
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)         38960448
['input_img[0][0]',

'validation_img[0][0]']
```

```
 distance (L1Dist)              (None, 4096)         0
['embedding[2][0]',

'embedding[3][0]']

 dense_3 (Dense)                (None, 1)            4097
['distance[0][0]']

============================================================================
=======================
Total params: 38,964,545
Trainable params: 38,964,545
Non-trainable params: 0
____________________________________________________________________________
_______________________
```

1. Training

5.1 Setup Loss and Optimizer

In [55]:

```
binary_cross_loss = tf.losses.BinaryCrossentropy()
```

In [56]:

```
opt = tf.keras.optimizers.Adam(1e-4) # 0.0001
```

5.2 Establish Checkpoints

In [57]:

```
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, 'ckpt')
checkpoint = tf.train.Checkpoint(opt=opt, siamese_model=siamese_model)
```

5.3 Build Train Step Function

In [58]:

```
test_batch = train_data.as_numpy_iterator()
```

In [59]:

```
batch_1 = test_batch.next()
```

In [60]:

```
X = batch_1[:2]
```

In [61]:

```
y = batch_1[2]
```

In [62]:

```
y
```

Out[62]:

```
array([0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0.],
      dtype=float32)
```

In [63]:

```
tf.losses.BinaryCrossentropy??
```

In [64]:

```
@tf.function
def train_step(batch):
    # Record all of our operations
    with tf.GradientTape() as tape:
        # Get anchor and positive/negative image
        X = batch[:2]
        # Get label
        y = batch[2]

        # Forward pass
        yhat = siamese_model(X, training=True)
        # Calculate loss
        loss = binary_cross_loss(y, yhat)
    print(loss)

    # Calculate gradients
    grad = tape.gradient(loss, siamese_model.trainable_variables)

    # Calculate updated weights and apply to siamese model
    opt.apply_gradients(zip(grad, siamese_model.trainable_variables))

    # Return loss
    return loss
```

5.4 Build Training Loop

In [65]:

```
def train(data, EPOCHS):
    # Loop through epochs
    for epoch in range(1, EPOCHS+1):
        print('\n Epoch {}/{}'.format(epoch, EPOCHS))
        progbar = tf.keras.utils.Progbar(len(data))

        # Loop through each batch
        for idx, batch in enumerate(data):
            # Run train step here
            train_step(batch)
            progbar.update(idx+1)

        # Save checkpoints
        if epoch % 10 == 0:
            checkpoint.save(file_prefix=checkpoint_prefix)
```

5.5 Train the model

In [66]:

```
EPOCHS = 25
```

In [67]:

```
train(train_data, EPOCHS)
```

```
 Epoch 1/25
```

```
Tensor("binary_crossentropy/weighted_loss/value:0", shape=(),
dtype=float32)
Tensor("binary_crossentropy/weighted_loss/value:0", shape=(),
dtype=float32)
26/27 [===========================>..] - ETA: 31s
Tensor("binary_crossentropy/weighted_loss/value:0", shape=(),
dtype=float32)
27/27 [==============================] - 841s 31s/step

 Epoch 2/25
27/27 [==============================] - 839s 31s/step

 Epoch 3/25
27/27 [==============================] - 848s 31s/step

 Epoch 4/25
27/27 [==============================] - 850s 31s/step

 Epoch 5/25
27/27 [==============================] - 849s 31s/step

 Epoch 6/25
27/27 [==============================] - 853s 32s/step

 Epoch 7/25
27/27 [==============================] - 853s 32s/step

 Epoch 8/25
27/27 [==============================] - 847s 31s/step

 Epoch 9/25
27/27 [==============================] - 856s 32s/step

 Epoch 10/25
27/27 [==============================] - 1070s 40s/step

 Epoch 11/25
27/27 [==============================] - 1294s 48s/step

 Epoch 12/25
27/27 [==============================] - 1452s 54s/step

 Epoch 13/25
27/27 [==============================] - 1243s 46s/step

 Epoch 14/25
27/27 [==============================] - 1380s 51s/step

 Epoch 15/25
```

```
27/27 [==============================] - 1280s 47s/step

 Epoch 16/25
27/27 [==============================] - 1404s 52s/step

 Epoch 17/25
27/27 [==============================] - 1392s 51s/step

 Epoch 18/25
27/27 [==============================] - 1207s 45s/step

 Epoch 19/25
27/27 [==============================] - 825s 30s/step

 Epoch 20/25
27/27 [==============================] - 813s 30s/step

 Epoch 21/25
27/27 [==============================] - 808s 30s/step

 Epoch 22/25
27/27 [==============================] - 819s 30s/step

 Epoch 23/25
27/27 [==============================] - 811s 30s/step

 Epoch 24/25
27/27 [==============================] - 850s 32s/step

 Epoch 25/25
27/27 [==============================] - 821s 30s/step
```

1. Evaluate Model

6.1 Import Metrics

In [71]:

```
# Import metric calculations
from tensorflow.keras.metrics import Precision, Recall
```

6.2 Make Predictions

In [72]:

```
# Get a batch of test data
test_input, test_val, y_true = test_data.as_numpy_iterator().next()
```

In [73]:

```
# Make predictions
y_hat = siamese_model.predict([test_input, test_val])
y_hat
```

Out[73]:

```
array([[6.9442613e-08],
       [1.0000000e+00],
       [3.7634537e-10],
       [1.0000000e+00],
       [9.9995184e-01],
       [1.0000000e+00],
       [1.0000000e+00],
       [1.0000000e+00],
       [4.9038396e-09],
       [1.0000000e+00],
       [9.9809766e-01],
       [1.0000000e+00],
       [1.0000000e+00],
       [2.0272344e-08],
       [1.0000000e+00],
       [1.0053657e-09]], dtype=float32)
```

In [74]:

Out[74]:

```
[0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0]
```

In [75]:

```
y_true
```

Out[75]:

```
array([0., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0.],
      dtype=float32)
```

6.3 Calculate Metrics

In [76]:

```
# Creating a metric object
m = Recall()

# Calculating the recall value
m.update_state(y_true, y_hat)

# Return Recall Result
m.result().numpy()
```

Out[76]:

```
1.0
```

In [77]:

```
# Creating a metric object
m = Precision()

# Calculating the recall value
m.update_state(y_true, y_hat)

# Return Recall Result
m.result().numpy()
```

Out[77]:

```
1.0
```

6.4 Viz Results

In [78]:

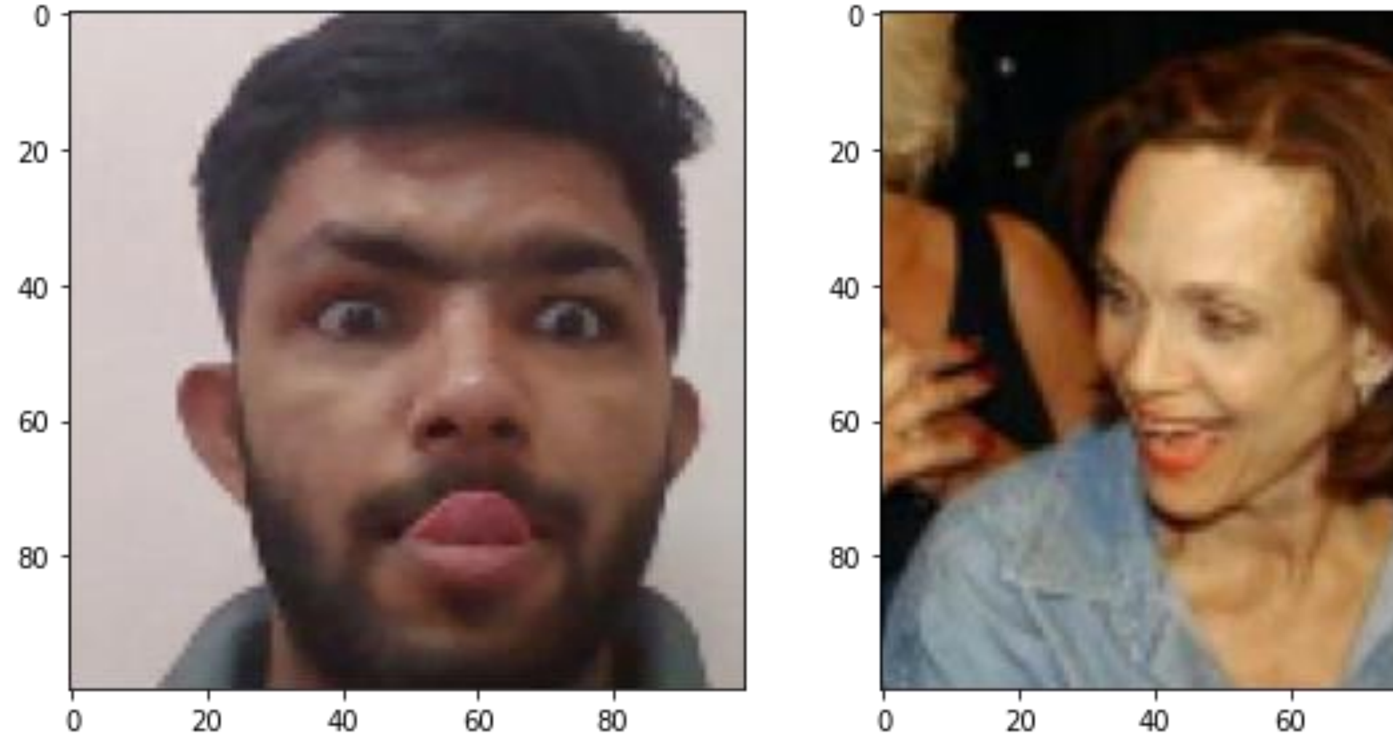
```
# Set plot size
```

```
plt.figure(figsize=(10,8))

# Set first subplot
plt.subplot(1,2,1)
plt.imshow(test_input[0])

# Set second subplot
plt.subplot(1,2,2)
plt.imshow(test_val[0])

# Renders cleanly
plt.show()
```



1. Save Model

In [79]:

```
# Save weights
siamese_model.save('siamesemodel.h5')
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
```

In [80]:

```
L1Dist
```

Out[80]:

```
__main__.L1Dist
```

In [81]:

```
# Reload model
model = tf.keras.models.load_model('siamesemodel.h5',
                                   custom_objects={'L1Dist':L1Dist,
'BinaryCrossentropy':tf.losses.BinaryCrossentropy})
WARNING:tensorflow:No training configuration found in the save file, so the
model was *not* compiled. Compile it manually.
```

In [82]:

```
# Make predictions with reloaded model
model.predict([test_input, test_val])
```

Out[82]:

```
array([[6.9442613e-08],
       [1.0000000e+00],
       [3.7634537e-10],
       [1.0000000e+00],
       [9.9995184e-01],
       [1.0000000e+00],
       [1.0000000e+00],
       [1.0000000e+00],
       [4.9038396e-09],
       [1.0000000e+00],
       [9.9809766e-01],
       [1.0000000e+00],
       [1.0000000e+00],
       [2.0272344e-08],
       [1.0000000e+00],
       [1.0053657e-09]], dtype=float32)
```

View model summary

model.summary()

1. Real Time Test

In [83]:

```
application_data\verification_images
```

```
  Input In [83]
    application_data\verification_images
                                       ^
SyntaxError: unexpected character after line continuation character
```

In [84]:

```
os.listdir(os.path.join('application_data', 'verification_images'))
```

Out[84]:

```
['4945cd02-bc7e-11ec-870b-c03eba3a1da6.jpg',
 '540ac8aa-bc7e-11ec-9656-c03eba3a1da6.jpg',
 '594a62d0-bc7e-11ec-ab5e-c03eba3a1da6.jpg',
 '639b258b-bc7e-11ec-946f-c03eba3a1da6.jpg',
 '6d0cfc61-bc7e-11ec-b2b5-c03eba3a1da6.jpg',
 '6d1b4ca6-bc7e-11ec-96fd-c03eba3a1da6.jpg',
 '6e4a8383-bc7e-11ec-86a6-c03eba3a1da6.jpg',
 '6e5fc8d2-bc7e-11ec-99ad-c03eba3a1da6.jpg',
 '6f77c40d-bc7e-11ec-b548-c03eba3a1da6.jpg',
 '6f8d3da2-bc7e-11ec-b870-c03eba3a1da6.jpg',
 '74f2276f-bc7e-11ec-b816-c03eba3a1da6.jpg',
 '761a6154-bc7e-11ec-8f85-c03eba3a1da6.jpg',
 '78b58c20-bc7e-11ec-9691-c03eba3a1da6.jpg',
 '798b8e7e-bc7e-11ec-9fc4-c03eba3a1da6.jpg',
 '7a565d9d-bc7e-11ec-9040-c03eba3a1da6.jpg',
 '7b1fc851-bc7e-11ec-b613-c03eba3a1da6.jpg',
 '7c5fc92f-bc7e-11ec-ab19-c03eba3a1da6.jpg',
 '7c6e480e-bc7e-11ec-b405-c03eba3a1da6.jpg',
 '7e757c77-bc7e-11ec-990f-c03eba3a1da6.jpg',
 '7f8b84fa-bc7e-11ec-b308-c03eba3a1da6.jpg',
 '82ec664b-bc7e-11ec-aaf2-c03eba3a1da6.jpg',
 '86594d73-bc7e-11ec-a952-c03eba3a1da6.jpg',
 '8a3292ad-bc7e-11ec-80ed-c03eba3a1da6.jpg',
 '8ad0c5d0-bc7e-11ec-a33a-c03eba3a1da6.jpg',
 '8d13178d-bc7e-11ec-891c-c03eba3a1da6.jpg']
```

In [85]:

```
os.path.join('application_data', 'input_image', 'input_image.jpg')
```

Out[85]:

```
'application_data\\input_image\\input_image.jpg'
```

In [86]:

```
for image in os.listdir(os.path.join('application_data', 
'verification_images')):
    validation_img = os.path.join('application_data', 
'verification_images', image)
    print(validation_img)
application_data\verification_images\4945cd02-bc7e-11ec-870b-
c03eba3a1da6.jpg
application_data\verification_images\540ac8aa-bc7e-11ec-9656-
c03eba3a1da6.jpg
application_data\verification_images\594a62d0-bc7e-11ec-ab5e-
c03eba3a1da6.jpg
application_data\verification_images\639b258b-bc7e-11ec-946f-
c03eba3a1da6.jpg
application_data\verification_images\6d0cfc61-bc7e-11ec-b2b5-
c03eba3a1da6.jpg
application_data\verification_images\6d1b4ca6-bc7e-11ec-96fd-
c03eba3a1da6.jpg
application_data\verification_images\6e4a8383-bc7e-11ec-86a6-
c03eba3a1da6.jpg
application_data\verification_images\6e5fc8d2-bc7e-11ec-99ad-
c03eba3a1da6.jpg
application_data\verification_images\6f77c40d-bc7e-11ec-b548-
c03eba3a1da6.jpg
application_data\verification_images\6f8d3da2-bc7e-11ec-b870-
c03eba3a1da6.jpg
application_data\verification_images\74f2276f-bc7e-11ec-b816-
c03eba3a1da6.jpg
```

```
application_data\verification_images\761a6154-bc7e-11ec-8f85-
c03eba3a1da6.jpg
application_data\verification_images\78b58c20-bc7e-11ec-9691-
c03eba3a1da6.jpg
application_data\verification_images\798b8e7e-bc7e-11ec-9fc4-
c03eba3a1da6.jpg
application_data\verification_images\7a565d9d-bc7e-11ec-9040-
c03eba3a1da6.jpg
application_data\verification_images\7b1fc851-bc7e-11ec-b613-
c03eba3a1da6.jpg
application_data\verification_images\7c5fc92f-bc7e-11ec-ab19-
c03eba3a1da6.jpg
application_data\verification_images\7c6e480e-bc7e-11ec-b405-
c03eba3a1da6.jpg
application_data\verification_images\7e757c77-bc7e-11ec-990f-
c03eba3a1da6.jpg
application_data\verification_images\7f8b84fa-bc7e-11ec-b308-
c03eba3a1da6.jpg
application_data\verification_images\82ec664b-bc7e-11ec-aaf2-
c03eba3a1da6.jpg
application_data\verification_images\86594d73-bc7e-11ec-a952-
c03eba3a1da6.jpg
application_data\verification_images\8a3292ad-bc7e-11ec-80ed-
c03eba3a1da6.jpg
application_data\verification_images\8ad0c5d0-bc7e-11ec-a33a-
c03eba3a1da6.jpg
application_data\verification_images\8d13178d-bc7e-11ec-891c-
c03eba3a1da6.jpg
```

In [87]:

```
def verify(model, detection_threshold, verification_threshold):
    # Build results array
    results = []
    for image in os.listdir(os.path.join('application_data',
'verification_images')):
        input_img = preprocess(os.path.join('application_data',
'input_image', 'input_image.jpg'))
        validation_img = preprocess(os.path.join('application_data',
'verification_images', image))

        # Make Predictions
        result = model.predict(list(np.expand_dims([input_img,
validation_img], axis=1)))
        results.append(result)

    # Detection Threshold: Metric above which a prediciton is considered
positive
    detection = np.sum(np.array(results) > detection_threshold)
    # Verification Threshold: Proportion of positive predictions / total
positive samples
    verification = detection /
len(os.listdir(os.path.join('application_data', 'verification_images')))
    verified = verification > verification_threshold

    return results, verified
```

8.2 OpenCV Real Time Verification

In [88]:

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    frame = frame[120:120+250,200:200+250, :]

    cv2.imshow('Verification', frame)

    # Verification trigger
    if cv2.waitKey(10) & 0xFF == ord('v'):
        # Save input image to application_data/input_image folder
        cv2.imwrite(os.path.join('application_data', 'input_image',
'input_image.jpg'), frame)
        # Run verification
        results, verified = verify(model, 0.9, 0.7)
        print(verified)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

```
True
True
True
True
False
True
True
True
```

In [89]:

```
np.sum(np.squeeze(results) > 0.9)
```

Out[89]:

```
21
```

In [ ]:

References

[1] Fe-Fei, Li, Fergus, Robert, and Perona, Pietro. A bayesian approach to unsupervised one-shot learning of object categories. In Computer Vision, 2003. Proceedings.

Ninth IEEE International Conference on, pp. 1134–
1141. IEEE, 2003.
[2]Fei-Fei, Li, Fergus, Robert, and Perona, Pietro. One-shot
learning of object categories. Pattern Analysis and Machine
Intelligence, IEEE Transactions on, 28(4):594–
611, 2006.
[3]Hinton, Geoffrey, Osindero, Simon, and Teh, Yee-Whye.
A fast learning algorithm for deep belief nets. Neural
computation, 18(7):1527–1554, 2006.
[4]Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E.
Imagenet classification with deep convolutional neural
networks. In Advances in neural information processing
systems, pp. 1097–1105, 2012.
[5] Lake, Brenden M, Salakhutdinov, Ruslan, Gross, Jason,
and Tenenbaum, Joshua B. One shot learning of simple
visual concepts. In Proceedings of the 33rd Annual Conference
of the Cognitive Science Society, volume 172,
2011.
[6] Lake, Brenden M, Salakhutdinov, Ruslan, and Tenenbaum,
Joshua B. Concept learning as motor program induction:
A large-scale empirical study. In Proceedings of the 34th
Annual Conference of the Cognitive Science Society, pp.
659–664, 2012.
[7] Lake, Brenden M, Salakhutdinov, Ruslan R, and Tenenbaum,
Josh. One-shot learning by inverting a compositional
causal process. In Advances in neural information
processing systems, pp. 2526–2534, 2013.
[8] Lake, Brenden M, Lee, Chia-ying, Glass, James R, and
Tenenbaum, Joshua B. One-shot learning of generative
speech concepts. Cognitive Science Society, 2014.
[9] Lim, Joseph Jaewhan. Transfer learning by borrowing examples
for multiclass object detection. Master's thesis,
Massachusetts Institute of Technology, 2012.
[10] Maas, Andrew and Kemp, Charles. One-shot learning with
bayesian networks. Cognitive Science Society, 2009.

Problem faced

1. *Data preprocessing, I find very difficult and I have taken help of stack overflow*
2. *Because of gpu and low computation power, its take lot of time to train the model. And checking the result of model I have to change the hyperparameter and again train the model. It's very difficult*
3. *I try to build the application also for the same for real time detection but because of some constrain I am not able to do but surely I Build the application for the same*