Facial Recognition using Deep Learning

Report Prepared in Complete Fulfillment of the

Project Course: CS F377

Design Project

By

REUBEN MENEZES

2015A7PS0291U

Under The Supervision of

Dr VADIVEL

Assistant Professor, Computer Science

At

BITS Pilani, Dubai Campus

Dubai International Academic City, Dubai

U.A.E

First Semester, 2018-19

<u>Facial Recognition using Deep Learning</u>

Report Prepared in Complete Fulfillment of the

Project Course: CS F377

Design Project

By

REUBEN MENEZES

2015A7PS0291U

Under The Supervision of

Dr VADIVEL

Assistant Professor, Computer Science

At

BITS Pilani, Dubai Campus

Dubai International Academic City, Dubai

U.A.E

First Semester, 2018-19

ABSTRACT

Course name: Design Project

Course number: CS F377

Duration: 23.08.2018 to 27.12.2018

Date of start: 23.08.2018

Date of submission: NIL

Title of Report: Facial Recognition using Deep Learning

Name / ID Number: REUBEN MENEZES / 2015A7PS0291U

Discipline of Student: B.E (Hons) Computer Science Engineering

Name of Project Supervisor: Dr. Vadivel

Keywords: Deep Learning, Convolutional Neural Networks, Facial Recognition

Project Area: Machine Learning , Facial Recognition

Abstract: Applying Deep learning techniques to create a model that will not only be able to detect faces but also recognise whose face it is. The main steps I am going to use is using a neural network to identify unique data points in faces and also return face encodings of each face.Storing the face encodings in a list along with whose face is is.The Test data returns face encodings which is then matched with the already known face encodings.Thus if there is a match the model can determine whose face it is.

Signature of the Student **Signature of Supervisor**

Date: **Date:**

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to Prof. Dr. R.N. Saha, Director - BITS Pilani, Dubai Campus who has given me an opportunity to understand and apply engineering concepts in real-life problems through the Project.

I am very grateful to Prof. Dr. Vadivel, Assistant Professor Computer Science Department, BITS PILANI Dubai Campus, my mentor and Project in-charge, for assisting me enthusiastically in the entire project. It is only with his guidance that the objectives of the project have been accomplished.

I am grateful to examiners Dr Sujala Shetty and CS Department Faculty Members for their valuable suggestions.

Above all, I thank the Lord for giving me the strength to carry out this work to the best of my abilities.

Name: Reuben Menezes

ID: 2015A7PS0291U

Certificate

To Whomsoever it May Concern

This is to certify that the Semester Project Report titled, Facial Recognition using Deep Learning ,submitted by Reuben Menezes, ID: 2015A7PS0291U ,in complete fulfillment of the requirement of CSF377 ,Design Project embodies the work done by him under my supervision.

Signature of the Supervisor

Name: Dr Vadivel

Designation: Assistant Professor

Date : 27/12/18

Table of contents

LITERATURE SURVEY

Facial Recognition Using Neural Networks

In recent years, deep convolution networks have achieved great success in the area of computer vision. Face recognition has been one of its extensively researched and most interesting applications. The significance of face recognition is due to its technical challenges and wide potential application in video surveillance, identity authentication, multimedia applications, home and office security, law enforcement and different human-computer interaction activities. The appearance of a person in an image is a result of the mutual effects of the illumination, pose, 3-D structure of face, occlusion and background which are uncontrolled constraints that are encountered in real world applications [1]. Different approaches have been proposed to deal with these technical hitches, like in [1] illumination normalization techniques have been proposed and in [2] ANN based classifier is used to resolve this non- linearity.

In recent years, deep learning methods, specifically CNN has attained remarkable results on face recognition in unconstrained environment. CNN learning based features are more resilient to complex intra-personal variations. CNN methods [3], [4], [5] have attained the best three face recognition rates on the FRUE benchmark database LFW (Labeled Faces in the Wild) [6].

Another major problem is the generalization capability of the face recognition systems. The training of CNN models shows exceptional performance with large training datasets, but they are not suitable for learning from few samples. In such cases the use of deep learning is put off as the network will overfit severely with smaller training data [7].

To deal with this problem, generation of additional synthetic training data has been investigated by a number of researchers [8], [9], [10]. The main advantages of synthetic training data is that there is an uninhibited control over the nuisance factor of samples and desired number of training samples can be generated [7]. Data synthesis has also been applied to other recognition problems like, object detection [10], text recognition [8] and scene understanding [9]. Data augmentation is one other easy method which has been used to tackle the problem of dearth of training data [11], [12]. It is very much like data synthesis, but easier to implement. It is more limited in, that available training images are distorted without influencing the semantic class labels. These include rotation, cropping, scaling, applying noise, etc [7], [13].

Deep Learning :

Deep learning techniques are a part of machine learning methods based on learning multiple levels of representation
and abstraction that helps to make sense of data such as images, sound, and text. Deep learning replaces handcrafted
feature extraction with efficient algorithms for unsupervised
or semi-supervised feature learning and hierarchical feature
extraction [15].
The recent upsurge of deep learning techniques was set off by the groundbreaking work of [16] in 2006, but deep learning has been here for quite a long time [17]. In previous
few years, deep learning has shown exceptional performance
in natural languages, speech recognition and computer vision [18]. Deep learning nowadays is generally centered on multilayered neural networks [17].
The deep neural architectures are: feedforward neural networks, recurrent neural networks and convolutional neural networks. Feedforward networks sends unstructured
information from one end called input to the other end called
output; so they are called feedforward. The feedforward networks approximates some function f by defining a mapping of y=f*(x; θ) and then learning the value of the parameters θ that best approximates f, so they are seen as

universal function approximators [19]. Recurrent neural

networks model dynamics over time (and space) using self-
replicated components. RNNs are specialized for processing

sequential data. They preserve some amount of memory, and
can save long-term dependencies. RNNs are powerful computational machines – they can approximate any program [20]. A convolutional neural network has trainable
filters and local neighborhood pooling operations applied alternatingly on the input images which results in a hierarchy of increasingly complex features. The pooling operations down-sample the input representation and enlarges the input patterns [22]. CNNs take advantage of the
repetitive local input patterns across time and space, so they
are translation-invariant – the capability found in visual cortex of a human [21]. Local input patterns are small data slices, of distinct size, e.g., a group of pixels in an image.

Face Recognition :

In previous years, the performance of face recognition algorithms has increased a great deal. The significance of face recognition is due to its technical challenges and wide potential application. The first popular face recognition

technique is Eigenface (Principal Component Analysis) [23]. It can be pictured as a single layer linear model. Fisherface (Linear Discriminant Analysis) [10] is also a single layer linear model. Laplacianface (Locality Preserving Projection) [24] also used linear features. Then,
many handcrafted local nonlinear feature based methods emerged, such as Local Phase Quantization (LPQ) [9], Local
Binary Patterns (LBP) [25], and Fisher vectors. These hand
crafted features achieved excellent face recognition performance, however it decreased considerably in

unconstrained environments where the face images cover
intra-personal variations like, pose, illumination,
expression
and occlusion as shown in Labeled Faces in the Wild
(LFW)
benchmark [6],[26].
In last few years, deep learning methods, especially CNN
has achieved very impressive results on face recognition
in
unconstrained environment. The main benefit of CNNs is
that all the processing layers, even the pixel level input
have
configurable parameters that can be learned from data.
This
averts the necessity for hand crafted feature design, and
replaces it with supervised data driven learning of
features.
CNN learning based features are more resilient to complex
intra-personal variations [26]. CNN methods [3], [4], [5]
have attained the best three face recognition rates on the
FRUE benchmark database LFW (Labeled Faces in the
Wild) [6].

Planned Approach :

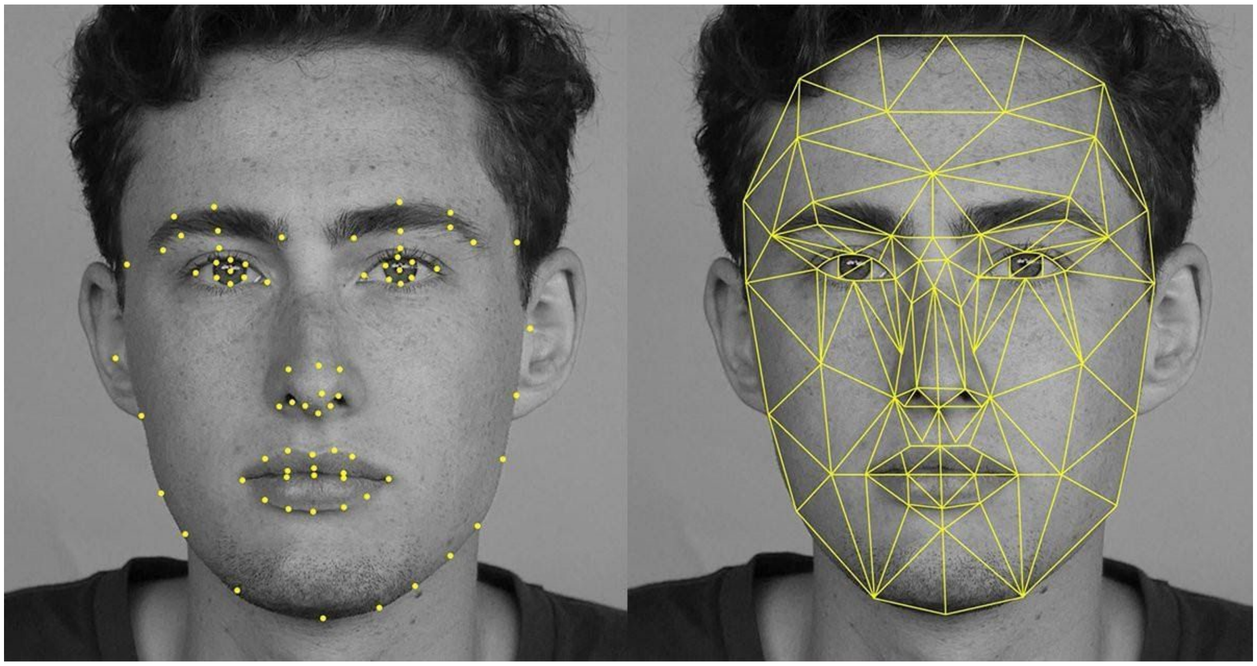
The details of the approach are presented in the following
steps:
1) Select a subset of image samples from an initially
labelled dataset and add it to the training set. Create a
testing set from the remaining samples.
2) Generate synthetic images by applying noise to each of
the training samples in the training set. The noise
applied could be any one of:
a. Poisson Noise,
b. Gaussian Noise.
3) Add these synthetically generated noisy image samples
to the training set to create the augmented training set
with double the number of training samples.
4) The augmented training set is then provided to the CNN
model for training.
5) The testing set is then recognized with the above
trained
network.
6) The percentage of recognition rate is calculated based
on the number of correct matches with respect to total

number of test images.

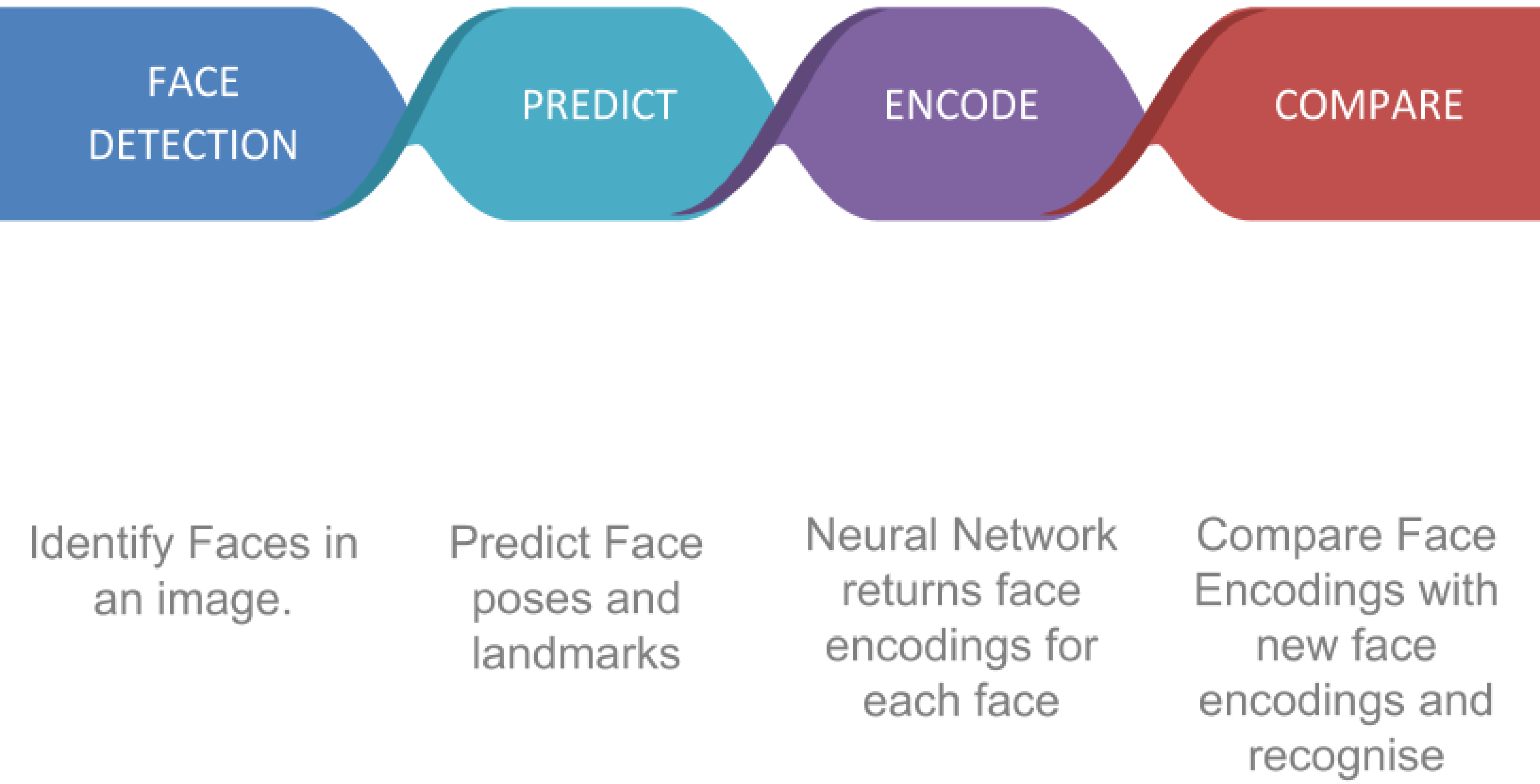
Deep convolutional neural networks have shown significant performance in the area of computer vision including the difficult face recognition problem. While the training of CNN models shows exceptional performance with large dataset but they are not suitable for learning from
datasets with few samples. To counter this problem of learning face representation from a smaller dataset, we propose a new approach where the training dataset is augmented with synthetically generated samples by adding
Gaussian or Poisson noise to the training set. This actually improves the generalization power of CNNs and adds robustness against overfitting to the model. Using the proposed approach for limited training data, 95.21% recognition rate is achieved on the standard AT&T face database for 4 training samples and 99.92% for 5 training samples. This technique can be applied to other face analysis problems in future. The limitation of this approach
is that it uses supervised learning with human-annotated data. The deep learning models are often big, so they demand large memory and more computational power which can be fulfilled with the use of GPUs. We need to tune many parameters and lack the understanding or interpretation of the model as it becomes more big and complicated.

OBJECTIVES

My Objective is to build a machine learning model to not only be able to detect faces but also to enable the machine to recognize faces .The Model should be able to recognize faces based on past training data when presented with new image data.



BLUEPRINT


FACE DETECTION
PREDICT
ENCODE
COMPARE
Identify Faces in an image.
Predict Face poses and landmarks
Neural Network returns face encodings for each face
Compare Face Encodings with new face encodings and recognise

STEPS :

TRAINING DATA:

- **Import the libraries required.**
- **Face Detection**
- **Tolerance level set as 0.5/0.6**
- **Send the images to the neural network**
- **Neural network sends back face encodings for the images**
- **Store Face Encodings in a array**

TESTING DATA:

- **Send Images to Neural network**

- **Neural network sends back face encodings for the images**
- **Compare new Face Encodings with the Encodings in the array and therefore recognize whose face it is.**

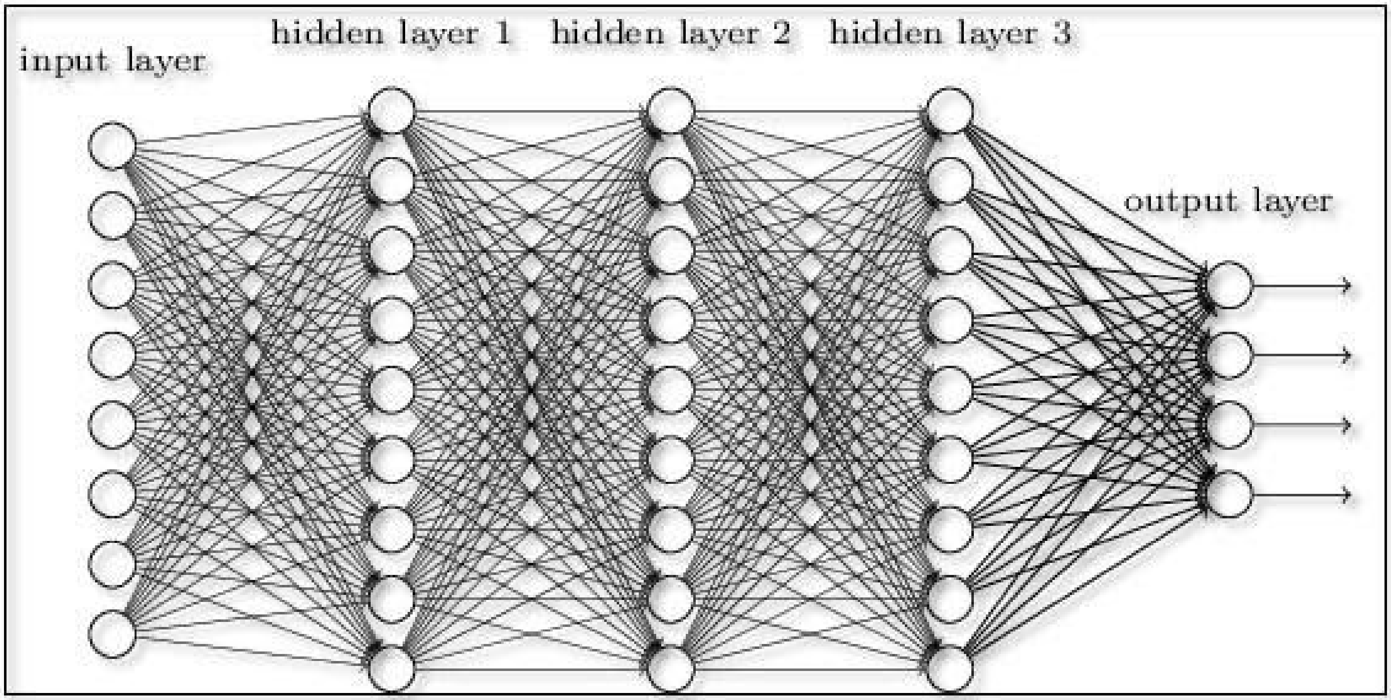
Neural Network

The Neural Network does the following and consists of an input layer , some hidden layers and an output layer.

- ***Regression***
- ***Clustering***
- ***Classification***
- ***Data Transformation***

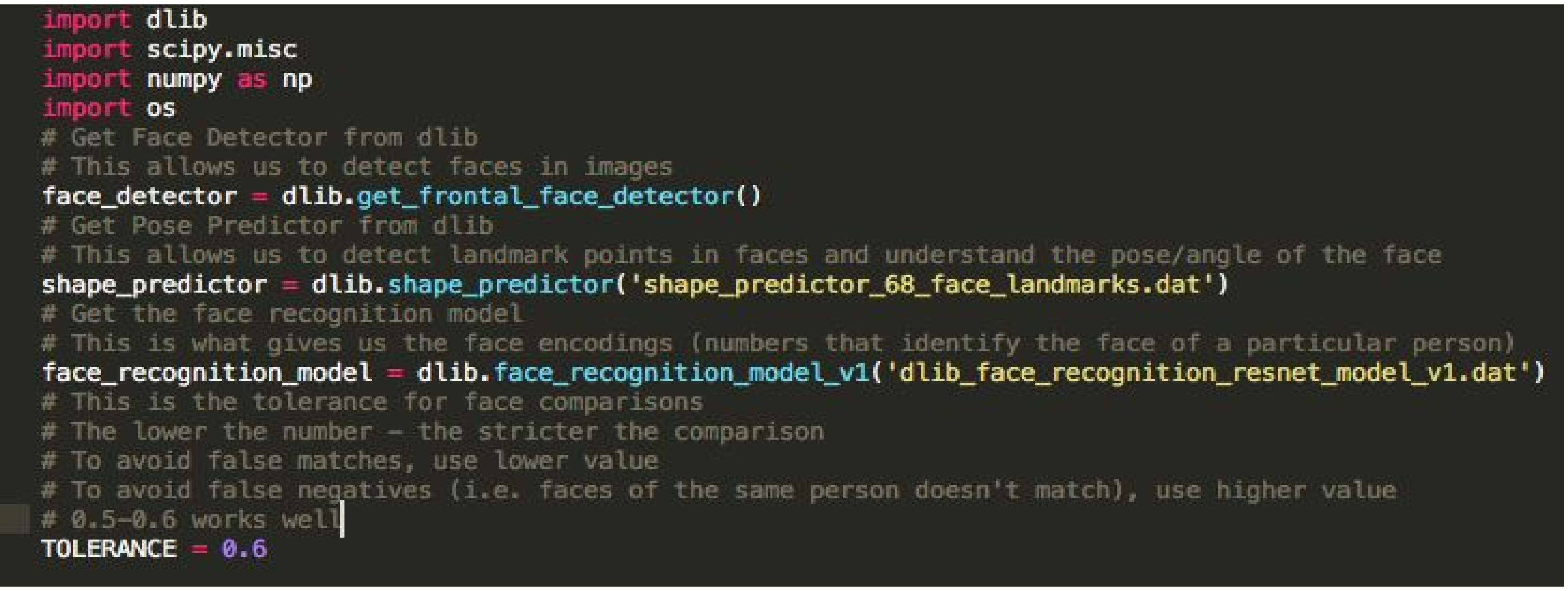
And finally

- ***Structured Prediction based on Markov Random Fields.***



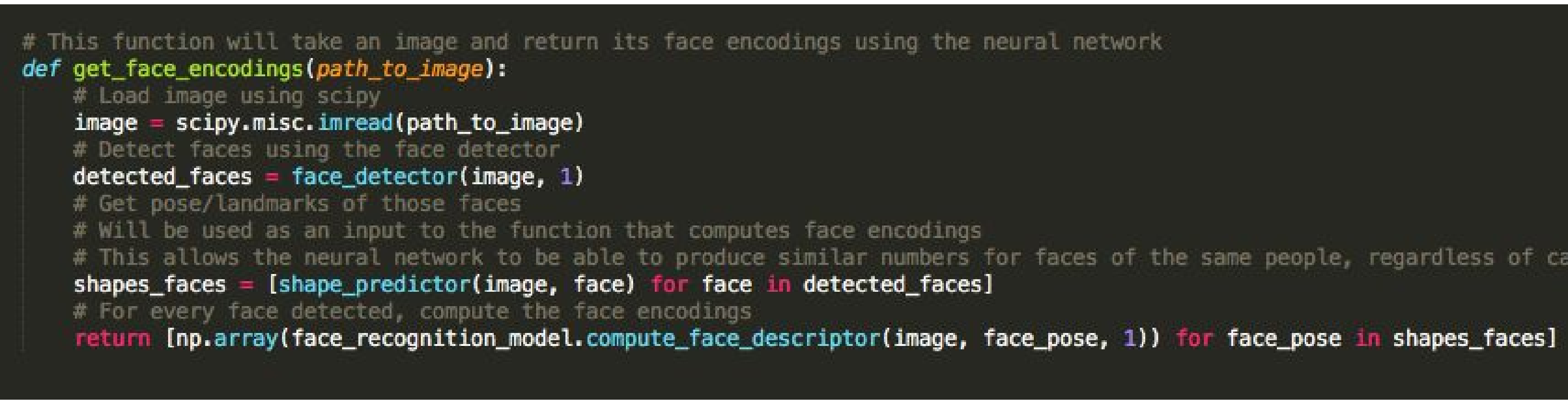
Code :

Imported Libraries required and used Dlib's Face Detector Function:



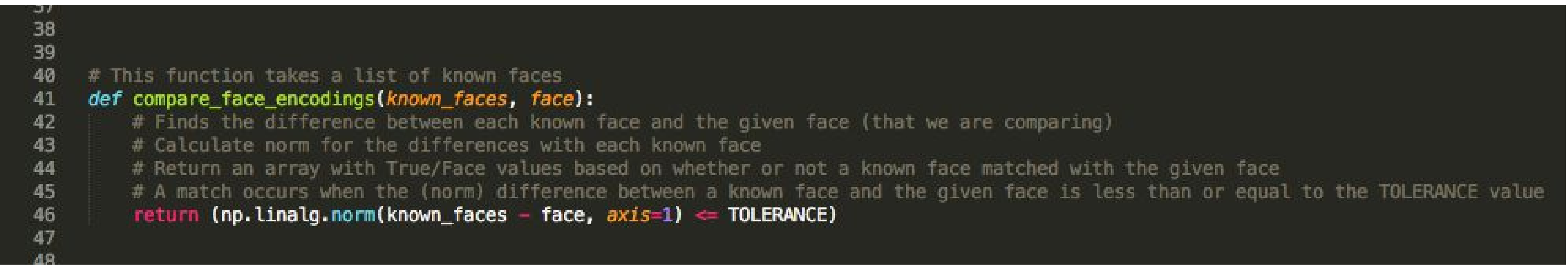
```
import dlib
import scipy.misc
import numpy as np
import os
# Get Face Detector from dlib
# This allows us to detect faces in images
face_detector = dlib.get_frontal_face_detector()
# Get Pose Predictor from dlib
# This allows us to detect landmark points in faces and understand the pose/angle of the face
shape_predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
# Get the face recognition model
# This is what gives us the face encodings (numbers that identify the face of a particular person)
face_recognition_model = dlib.face_recognition_model_v1('dlib_face_recognition_resnet_model_v1.dat')
# This is the tolerance for face comparisons
# The lower the number - the stricter the comparison
# To avoid false matches, use lower value
# To avoid false negatives (i.e. faces of the same person doesn't match), use higher value
# 0.5-0.6 works well
TOLERANCE = 0.6
```

Send Training Data to Neural Network and get Face Encoding from image.



```
# This function will take an image and return its face encodings using the neural network
def get_face_encodings(path_to_image):
    # Load image using scipy
    image = scipy.misc.imread(path_to_image)
    # Detect faces using the face detector
    detected_faces = face_detector(image, 1)
    # Get pose/landmarks of those faces
    # Will be used as an input to the function that computes face encodings
    # This allows the neural network to be able to produce similar numbers for faces of the same people, regardless of ca
    shapes_faces = [shape_predictor(image, face) for face in detected_faces]
    # For every face detected, compute the face encodings
    return [np.array(face_recognition_model.compute_face_descriptor(image, face_pose, 1)) for face_pose in shapes_faces]
```

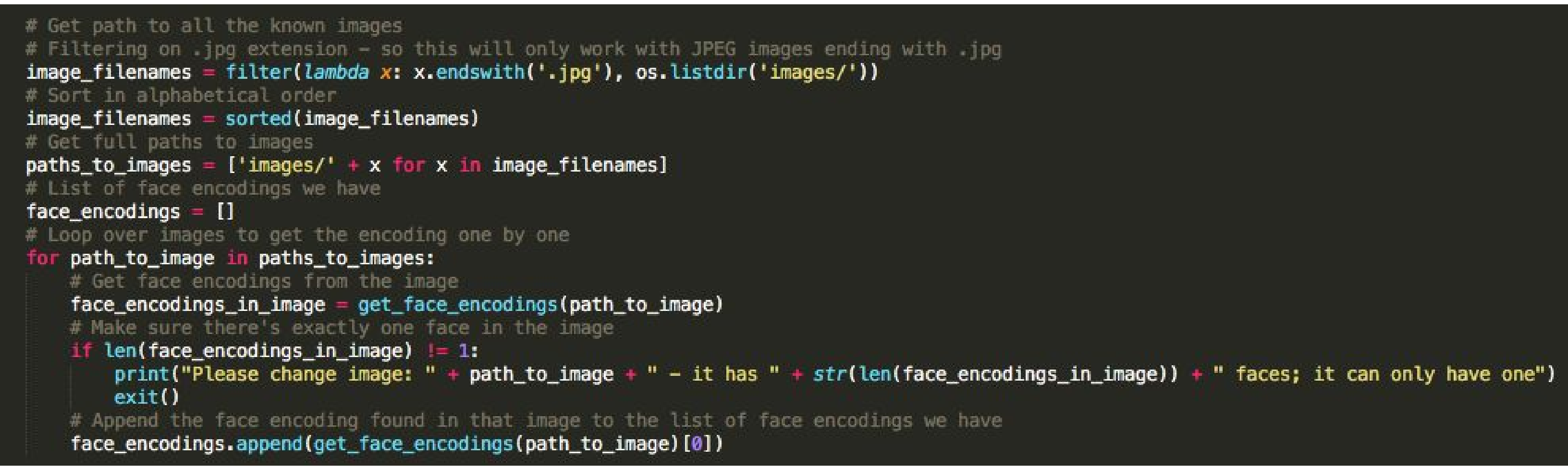
Store the Encodings received from the neural network in a list



```
# This function takes a list of known faces
def compare_face_encodings(known_faces, face):
    # Finds the difference between each known face and the given face (that we are comparing)
    # Calculate norm for the differences with each known face
    # Return an array with True/Face values based on whether or not a known face matched with the given face
    # A match occurs when the (norm) difference between a known face and the given face is less than or equal to the TOLERANCE value
    return (np.linalg.norm(known_faces - face, axis=1) <= TOLERANCE)
```

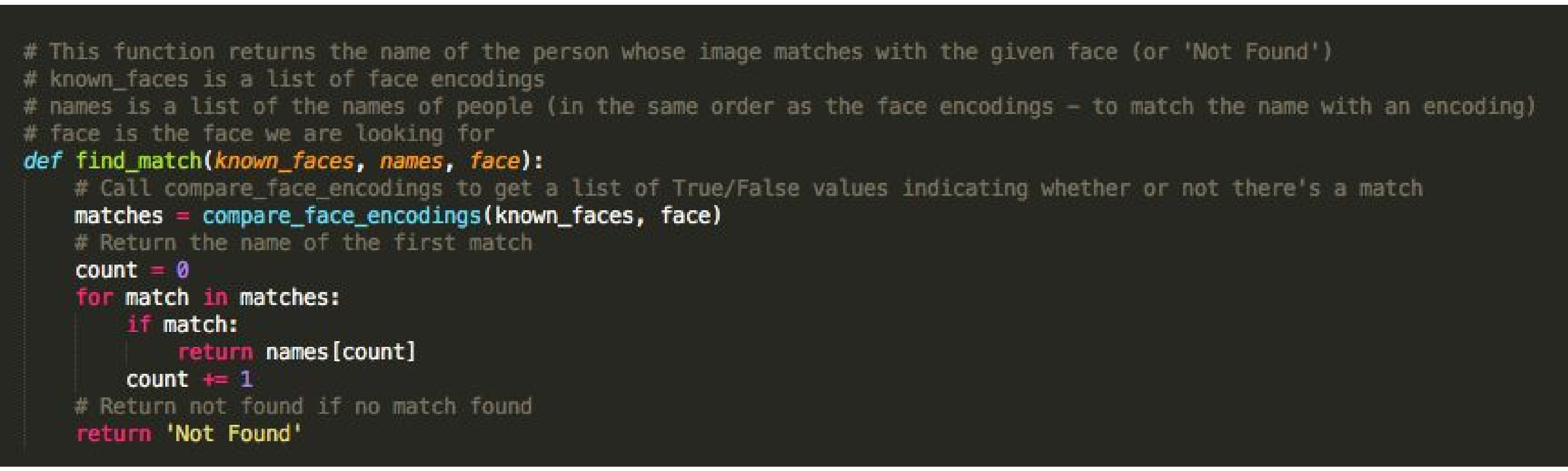
Define the Path for the training images.

Filter so the neural network selects only .jpg files



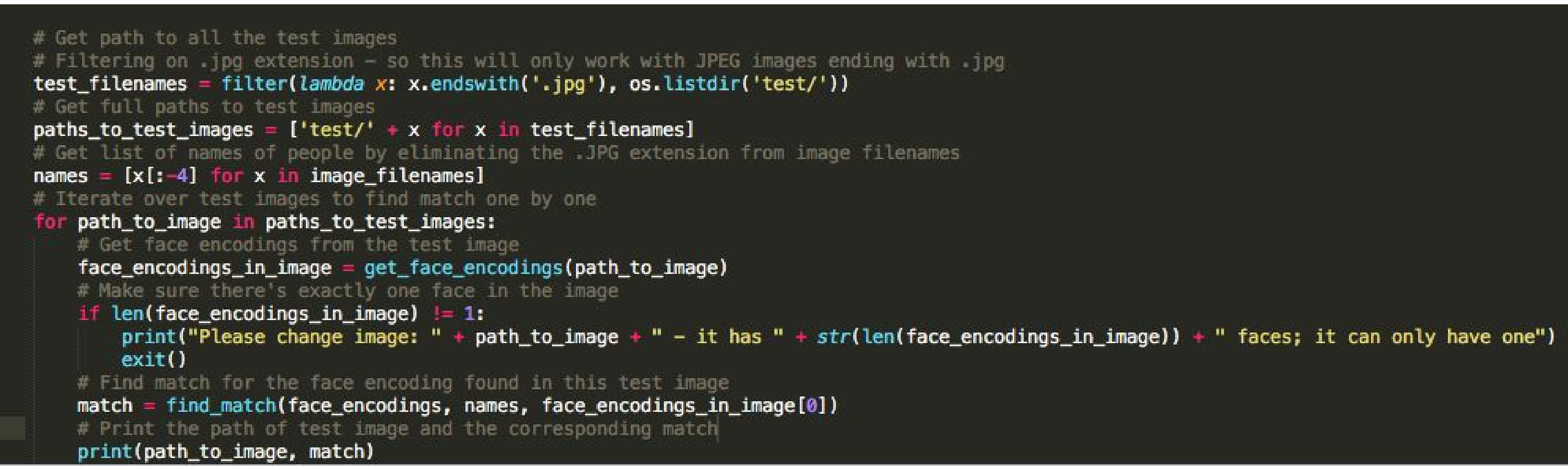
```
# Get path to all the known images
# Filtering on .jpg extension - so this will only work with JPEG images ending with .jpg
image_filenames = filter(lambda x: x.endswith('.jpg'), os.listdir('images/'))
# Sort in alphabetical order
image_filenames = sorted(image_filenames)
# Get full paths to images
paths_to_images = ['images/' + x for x in image_filenames]
# List of face encodings we have
face_encodings = []
# Loop over images to get the encoding one by one
for path_to_image in paths_to_images:
    # Get face encodings from the image
    face_encodings_in_image = get_face_encodings(path_to_image)
    # Make sure there's exactly one face in the image
    if len(face_encodings_in_image) != 1:
        print("Please change image: " + path_to_image + " - it has " + str(len(face_encodings_in_image)) + " faces; it can only have one")
        exit()
    # Append the face encoding found in that image to the list of face encodings we have
    face_encodings.append(get_face_encodings(path_to_image)[0])
```

Define the function find_match which compares encoding in list to the testing image encodings



```python
# This function returns the name of the person whose image matches with the given face (or 'Not Found')
# known_faces is a list of face encodings
# names is a list of the names of people (in the same order as the face encodings - to match the name with an encoding)
# face is the face we are looking for
def find_match(known_faces, names, face):
    # Call compare_face_encodings to get a list of True/False values indicating whether or not there's a match
    matches = compare_face_encodings(known_faces, face)
    # Return the name of the first match
    count = 0
    for match in matches:
        if match:
            return names[count]
        count += 1
    # Return not found if no match found
    return 'Not Found'
```

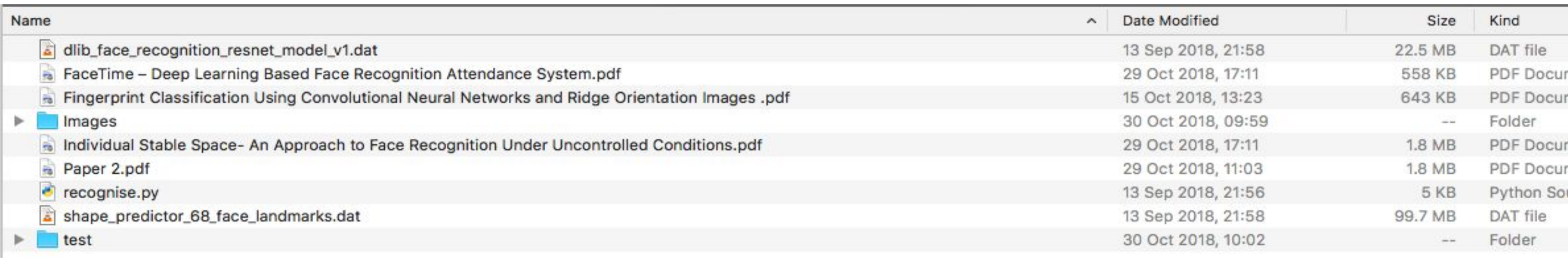
Define the path for the training images



```python
# Get path to all the test images
# Filtering on .jpg extension - so this will only work with JPEG images ending with .jpg
test_filenames = filter(lambda x: x.endswith('.jpg'), os.listdir('test/'))
# Get full paths to test images
paths_to_test_images = ['test/' + x for x in test_filenames]
# Get list of names of people by eliminating the .JPG extension from image filenames
names = [x[:-4] for x in image_filenames]
# Iterate over test images to find match one by one
for path_to_image in paths_to_test_images:
    # Get face encodings from the test image
    face_encodings_in_image = get_face_encodings(path_to_image)
    # Make sure there's exactly one face in the image
    if len(face_encodings_in_image) != 1:
        print("Please change image: " + path_to_image + " - it has " + str(len(face_encodings_in_image)) + " faces; it can only have one")
        exit()
    # Find match for the face encoding found in this test image
    match = find_match(face_encodings, names, face_encodings_in_image[0])
    # Print the path of test image and the corresponding match
    print(path_to_image, match)
```

Preparations

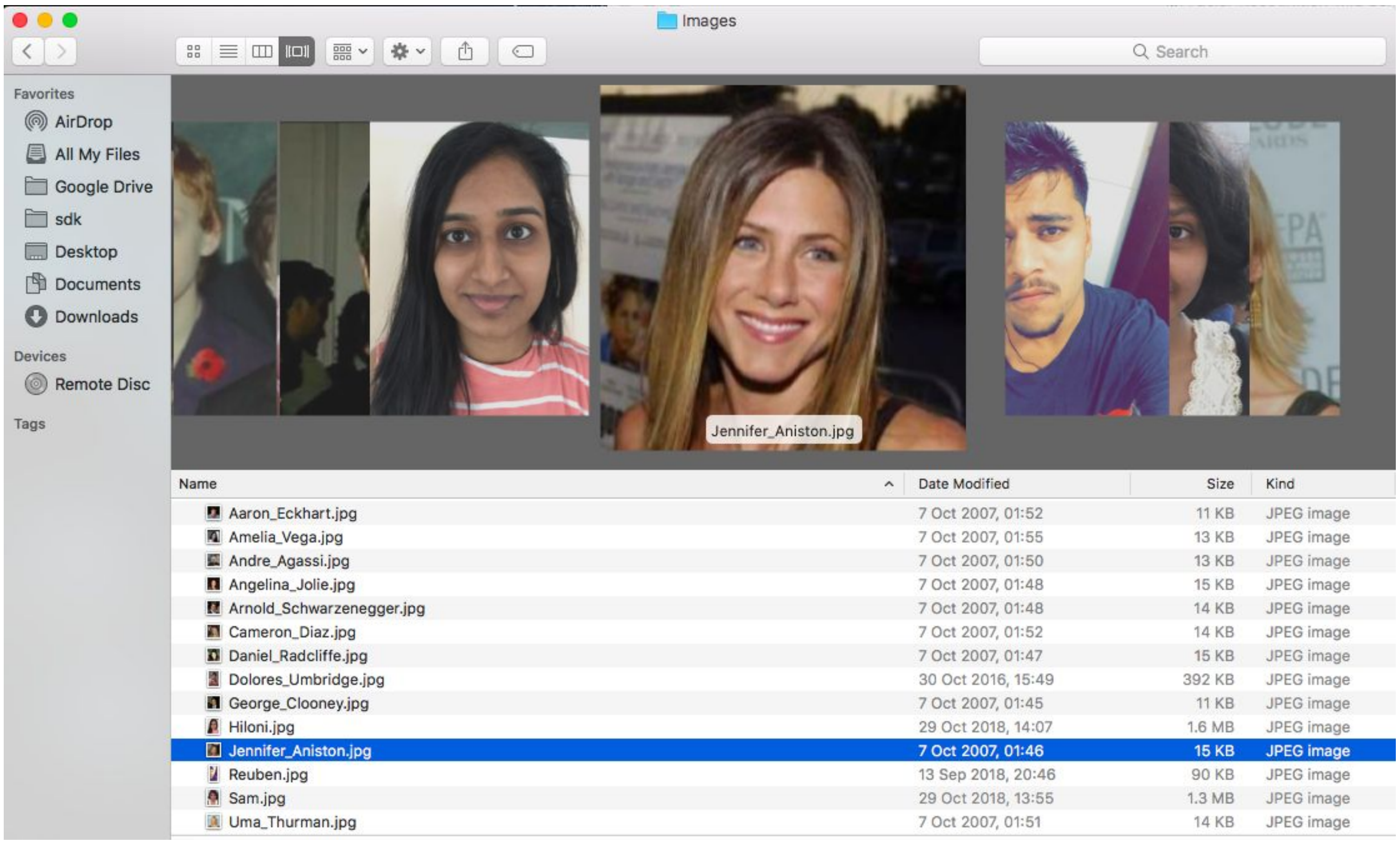
Created a folder called Images which stores .jpg training images



Name	Date Modified	Size	Kind
dlib_face_recognition_resnet_model_v1.dat	13 Sep 2018, 21:58	22.5 MB	DAT file
FaceTime – Deep Learning Based Face Recognition Attendance System.pdf	29 Oct 2018, 17:11	558 KB	PDF Docur
Fingerprint Classification Using Convolutional Neural Networks and Ridge Orientation Images .pdf	15 Oct 2018, 13:23	643 KB	PDF Docur
Images	30 Oct 2018, 09:59	--	Folder
Individual Stable Space- An Approach to Face Recognition Under Uncontrolled Conditions.pdf	29 Oct 2018, 17:11	1.8 MB	PDF Docur
Paper 2.pdf	29 Oct 2018, 11:03	1.8 MB	PDF Docur
recognise.py	13 Sep 2018, 21:56	5 KB	Python Sou
shape_predictor_68_face_landmarks.dat	13 Sep 2018, 21:58	99.7 MB	DAT file
test	30 Oct 2018, 10:02	--	Folder

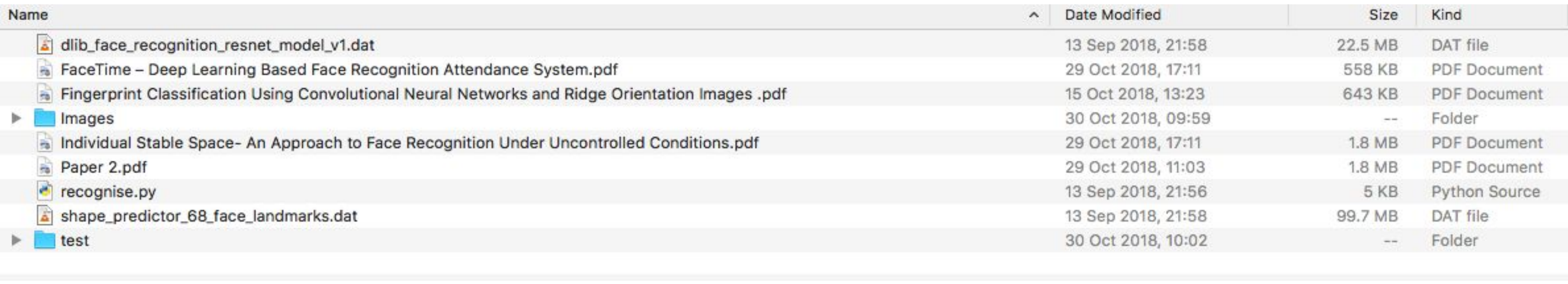
Store Training Data with the names corresponding to each face

And store each file with .jpg extension



Name	Date Modified	Size	Kind
Aaron_Eckhart.jpg	7 Oct 2007, 01:52	11 KB	JPEG image
Amelia_Vega.jpg	7 Oct 2007, 01:55	13 KB	JPEG image
Andre_Agassi.jpg	7 Oct 2007, 01:50	13 KB	JPEG image
Angelina_Jolie.jpg	7 Oct 2007, 01:48	15 KB	JPEG image
Arnold_Schwarzenegger.jpg	7 Oct 2007, 01:48	14 KB	JPEG image
Cameron_Diaz.jpg	7 Oct 2007, 01:52	14 KB	JPEG image
Daniel_Radcliffe.jpg	7 Oct 2007, 01:47	15 KB	JPEG image
Dolores_Umbridge.jpg	30 Oct 2016, 15:49	392 KB	JPEG image
George_Clooney.jpg	7 Oct 2007, 01:45	11 KB	JPEG image
Hiloni.jpg	29 Oct 2018, 14:07	1.6 MB	JPEG image
Jennifer_Aniston.jpg	7 Oct 2007, 01:46	15 KB	JPEG image
Reuben.jpg	13 Sep 2018, 20:46	90 KB	JPEG image
Sam.jpg	29 Oct 2018, 13:55	1.3 MB	JPEG image
Uma_Thurman.jpg	7 Oct 2007, 01:51	14 KB	JPEG image

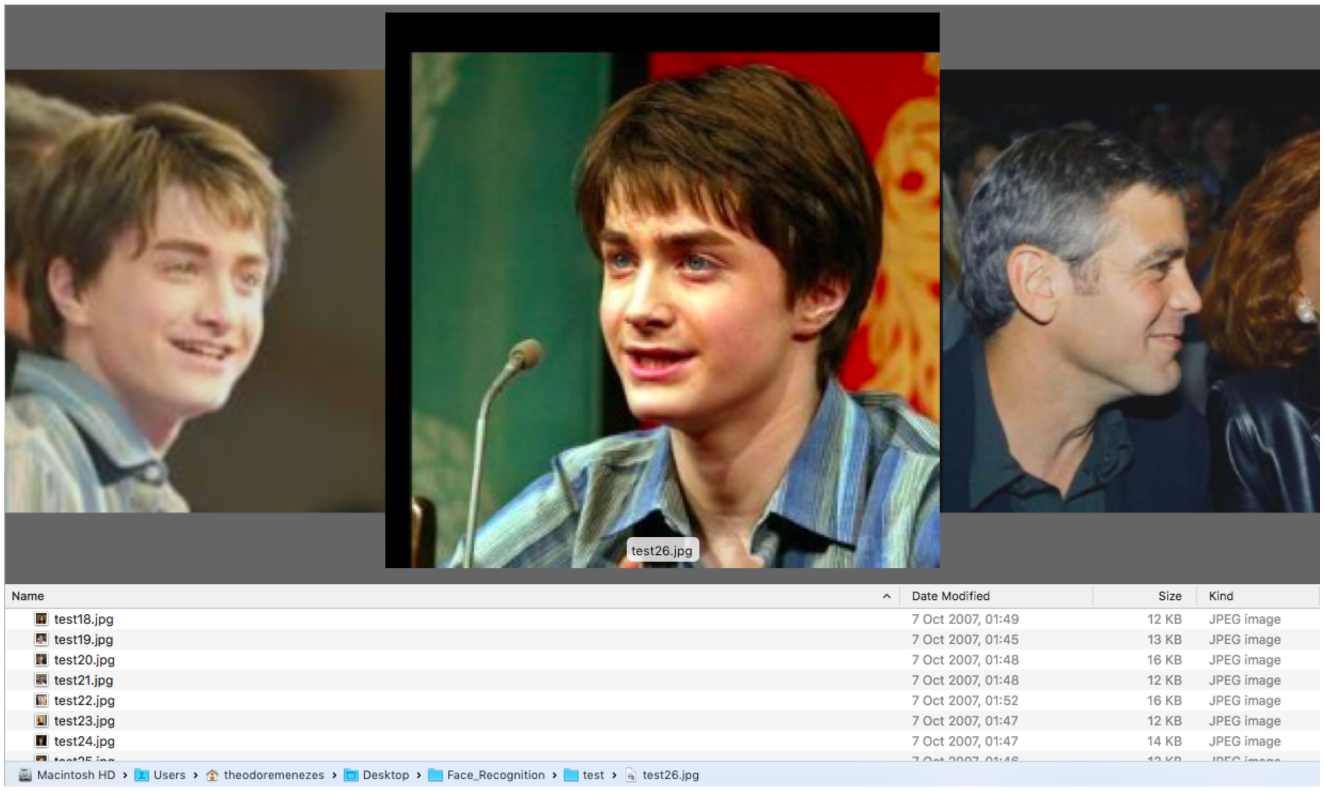
Store Test Images in a folder called test



Name	Date Modified	Size	Kind
dlib_face_recognition_resnet_model_v1.dat	13 Sep 2018, 21:58	22.5 MB	DAT file
FaceTime – Deep Learning Based Face Recognition Attendance System.pdf	29 Oct 2018, 17:11	558 KB	PDF Document
Fingerprint Classification Using Convolutional Neural Networks and Ridge Orientation Images .pdf	15 Oct 2018, 13:23	643 KB	PDF Document
Images	30 Oct 2018, 09:59	--	Folder
Individual Stable Space- An Approach to Face Recognition Under Uncontrolled Conditions.pdf	29 Oct 2018, 17:11	1.8 MB	PDF Document
Paper 2.pdf	29 Oct 2018, 11:03	1.8 MB	PDF Document
recognise.py	13 Sep 2018, 21:56	5 KB	Python Source
shape_predictor_68_face_landmarks.dat	13 Sep 2018, 21:58	99.7 MB	DAT file
test	30 Oct 2018, 10:02	--	Folder

Store Test images in the test folder

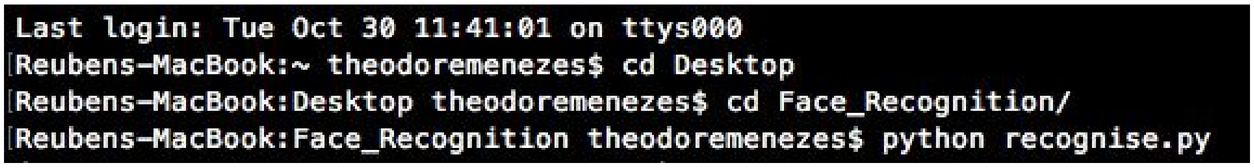
In the format shown in the picture below



Name	Date Modified	Size	Kind
test18.jpg	7 Oct 2007, 01:49	12 KB	JPEG image
test19.jpg	7 Oct 2007, 01:45	13 KB	JPEG image
test20.jpg	7 Oct 2007, 01:48	16 KB	JPEG image
test21.jpg	7 Oct 2007, 01:48	12 KB	JPEG image
test22.jpg	7 Oct 2007, 01:52	16 KB	JPEG image
test23.jpg	7 Oct 2007, 01:47	12 KB	JPEG image
test24.jpg	7 Oct 2007, 01:47	14 KB	JPEG image

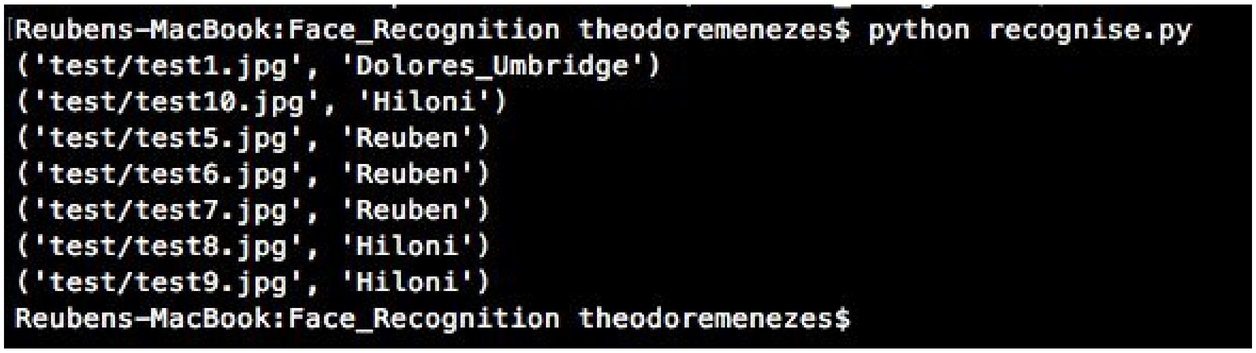
Macintosh HD › Users › theodoremenezes › Desktop › Face_Recognition › test › test26.jpg

Used terminal to run the code


```
Last login: Tue Oct 30 11:41:01 on ttys000
[Reubens-MacBook:~ theodoremenezes$ cd Desktop
[Reubens-MacBook:Desktop theodoremenezes$ cd Face_Recognition/
[Reubens-MacBook:Face_Recognition theodoremenezes$ python recognise.py
```

Output

We Receive the name recognized by the neural network alongside each test image

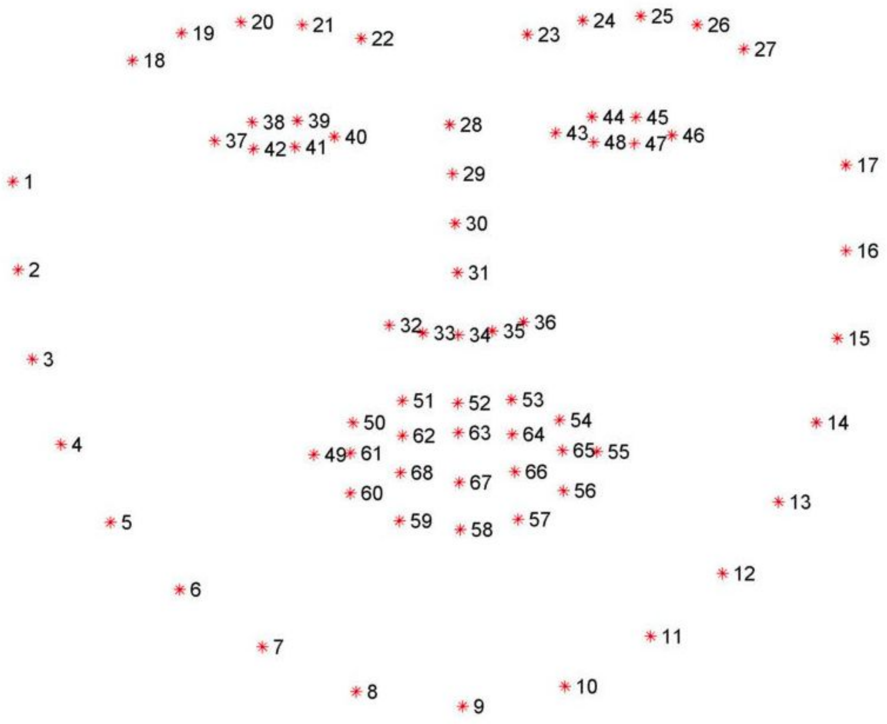

```
[Reubens-MacBook:Face_Recognition theodoremenezes$ python recognise.py
('test/test1.jpg', 'Dolores_Umbridge')
('test/test10.jpg', 'Hiloni')
('test/test5.jpg', 'Reuben')
('test/test6.jpg', 'Reuben')
('test/test7.jpg', 'Reuben')
('test/test8.jpg', 'Hiloni')
('test/test9.jpg', 'Hiloni')
Reubens-MacBook:Face_Recognition theodoremenezes$
```

Repetitive Training with Large Dataset

I trained the model with a dataset found on the internet which had pictures of 5749 people and each person has 1 picture or more.[Github Link].

Each picture is pre processed and brought down to a square of 250 * 250 px. This size helps the neural network to determine the faces clearly.

We use dlib as API which helps in face detection and uses the 68 point system where it tries to help find the 68 points which determine if a face is present in the image



Capabilities of the Neural Network after being trained on a large Dataset

1. The neural network having been trained on a large enough dataset could easily determine who the person in the image was even if the person’s face wasn’t present in the picture in it’s entirety.

Example:

The Neural Network is able to recognize both these pictures as George Clooney even though his face is not completely visible based on the probability that the face encodings returned from these images has a match of more than 0.6 of George Clooney’s original encodings.

2.Different Settings like facial hair and spectacles just train the model better to recognize the person easily.

The Neural Network identifies both these pictures as Hiloni Mehta.

3.Different Angles and Different Haircuts do not hinder the accurate prediction of the neural network

Example:

Neural Network recognizes these pictures as Jennifer Aniston irrespective of the fact that she has different haircuts and looks older across the images.

Conclusion

In conclusion my project has achieved

- *Recognition of faces even though the complete face is not in the image.*
- *Recognition of faces over the years.[Age does not factor]*
- *Recognition of faces with and without external influencers such as spectacles.*
- *Recognition of a person through different hairstyles.*

I used a Dataset of images with 5749 distinct faces and atleast one picture per face which helped my model to distinguish and learn and therefore make more accurate predictions based on probability.

The Accuracy Calculated is at 94.06% using the dataset I found[link is attached in references]

References

Papers I referenced :

1. https://ieeexplore.ieee.org/document/7231817

2. https://ieeexplore.ieee.org/document/840634

3. https://ieeexplore.ieee.org/document/7351562

Dataset I used : Can be found on my Github Page

https://github.com/reuben154/Face_Classification-using-Deep-Learnin
g

Appendix:

Source Code :

```
import dlib
import scipy.misc
import numpy as np
import os
# Get Face Detector from dlib
# This allows us to detect faces in images
face_detector = dlib.get_frontal_face_detector()
# Get Pose Predictor from dlib
# This allows us to detect landmark points in faces and understand the
pose/angle of the face
shape_predictor =
dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
# Get the face recognition model
# This is what gives us the face encodings (numbers that identify the face
of a particular person)
face_recognition_model =
dlib.face_recognition_model_v1('dlib_face_recognition_resnet_model_v1.da
t')
```

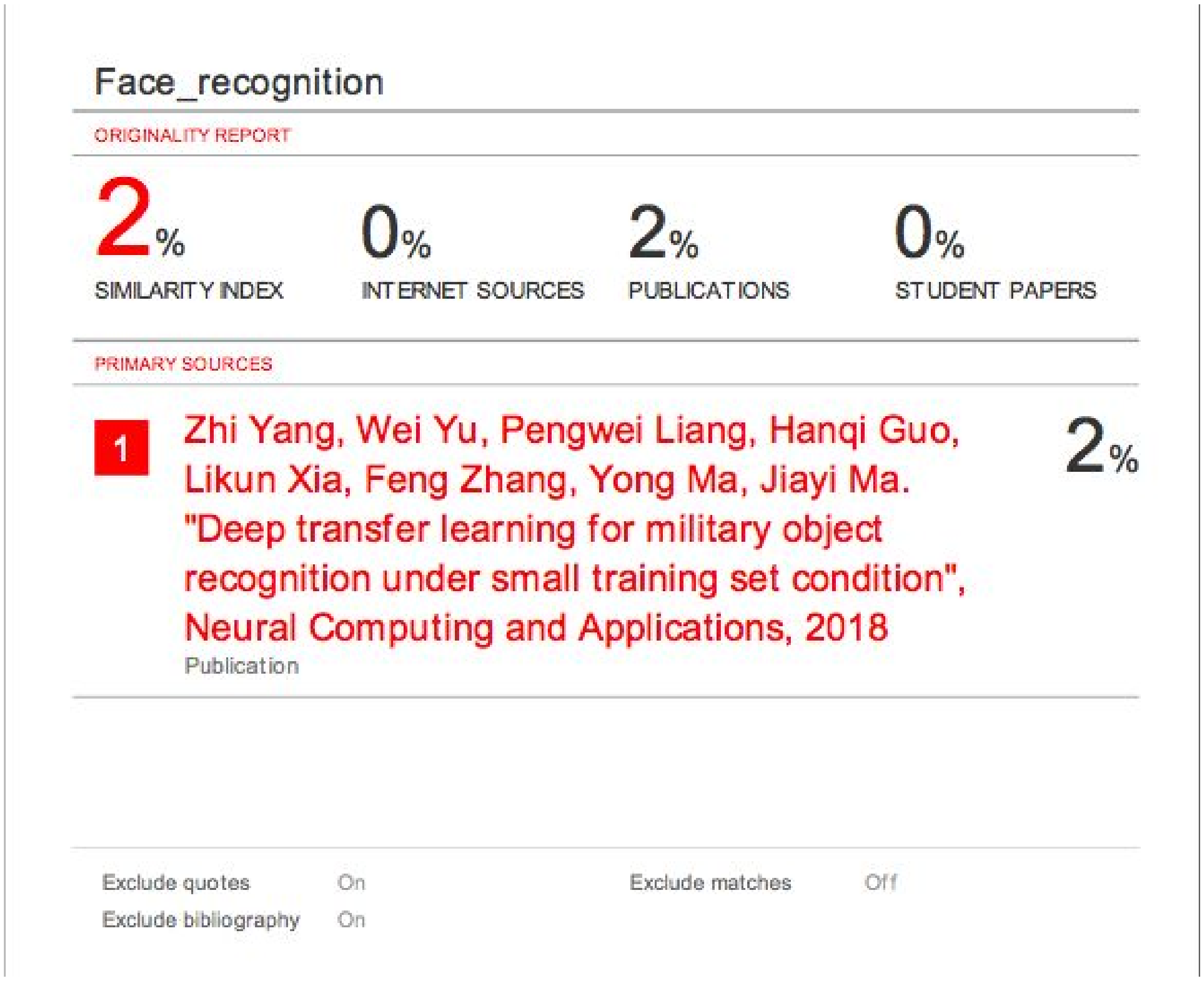
```
# This is the tolerance for face comparisons
# The lower the number - the stricter the comparison
# To avoid false matches, use lower value
# To avoid false negatives (i.e. faces of the same person doesn't match), use higher value
# 0.5-0.6 works well
TOLERANCE = 0.6


# This function will take an image and return its face encodings using the neural network
def get_face_encodings(path_to_image):
    # Load image using scipy
    image = scipy.misc.imread(path_to_image)
    # Detect faces using the face detector
    detected_faces = face_detector(image, 1)
    # Get pose/landmarks of those faces
    # Will be used as an input to the function that computes face encodings
    # This allows the neural network to be able to produce similar numbers for faces of the same people, regardless of camera angle and/or face positioning in the image
    shapes_faces = [shape_predictor(image, face) for face in detected_faces]
    # For every face detected, compute the face encodings
    return
[np.array(face_recognition_model.compute_face_descriptor(image,
face_pose, 1)) for face_pose in shapes_faces]




# This function takes a list of known faces

def compare_face_encodings(known_faces, face):

    # Finds the difference between each known face and the given face (that
we are comparing)

    # Calculate norm for the differences with each known face

    # Return an array with True/Face values based on whether or not a
known face matched with the given face

    # A match occurs when the (norm) difference between a known face and
the given face is less than or equal to the TOLERANCE value

    return (np.linalg.norm(known_faces - face, axis=1) <= TOLERANCE)



# This function returns the name of the person whose image matches with
the given face (or 'Not Found')

# known_faces is a list of face encodings

# names is a list of the names of people (in the same order as the face
encodings - to match the name with an encoding)
```

```
# face is the face we are looking for
def find_match(known_faces, names, face):
    # Call compare_face_encodings to get a list of True/False values
indicating whether or not there's a match
    matches = compare_face_encodings(known_faces, face)
    # Return the name of the first match
    count = 0
    for match in matches:
        if match:
            return names[count]
        count += 1
    # Return not found if no match found
    return 'Not Found'
# Get path to all the known images
# Filtering on .jpg extension - so this will only work with JPEG images
ending with .jpg
image_filenames = filter(lambda x: x.endswith('.jpg'), os.listdir('images/'))
# Sort in alphabetical order
image_filenames = sorted(image_filenames)
# Get full paths to images
paths_to_images = ['images/' + x for x in image_filenames]
# List of face encodings we have
```

```
face_encodings = []

# Loop over images to get the encoding one by one

for path_to_image in paths_to_images:

    # Get face encodings from the image

    face_encodings_in_image = get_face_encodings(path_to_image)

    # Make sure there's exactly one face in the image

    if len(face_encodings_in_image) != 1:

        print("Please change image: " + path_to_image + " - it has " +
str(len(face_encodings_in_image)) + " faces; it can only have one")

        exit()

    # Append the face encoding found in that image to the list of face
encodings we have

    face_encodings.append(get_face_encodings(path_to_image)[0])


# Get path to all the test images

# Filtering on .jpg extension - so this will only work with JPEG images
ending with .jpg

test_filenames = filter(lambda x: x.endswith('.jpg'), os.listdir('test/'))

# Get full paths to test images

paths_to_test_images = ['test/' + x for x in test_filenames]

# Get list of names of people by eliminating the .JPG extension from image
filenames

names = [x[:-4] for x in image_filenames]
```

```
# Iterate over test images to find match one by one
for path_to_image in paths_to_test_images:
    # Get face encodings from the test image
    face_encodings_in_image = get_face_encodings(path_to_image)
    # Make sure there's exactly one face in the image
    if len(face_encodings_in_image) != 1:
        print("Please change image: " + path_to_image + " - it has " +
str(len(face_encodings_in_image)) + " faces; it can only have one")
        exit()
    # Find match for the face encoding found in this test image
    match = find_match(face_encodings, names,
face_encodings_in_image[0])
    # Print the path of test image and the corresponding match
    print(path_to_image, match)
```

TURNITIN REPORT



Face_recognition

ORIGINALITY REPORT

2%	0%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Zhi Yang, Wei Yu, Pengwei Liang, Hanqi Guo, Likun Xia, Feng Zhang, Yong Ma, Jiayi Ma. "Deep transfer learning for military object recognition under small training set condition", Neural Computing and Applications, 2018 Publication	2%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		