1. Abstract

Face identification or recognition becomes very popular nowadays. It’s application used in many fields. Facebook, Google or Apple all the giant tech company uses face recognization in their various products. In this report paper, we discuss our face recognization procedure and it’s application in image categorization.

2. Introduction

Face recognition is a procedure to identify or verify a person from a digital image or a video frame from a video source. It is used in various fields. Today smartphones use facial recognition for access control while animated movies such as Avatar use it to bring realistic movement and expression to life. Many detective organizations try to capture criminal using face recognition. And many of us used celebrity look-a-like apps and Facebook’s auto tagger that classifies us, our friends, and our family. Recently FBI uses machine learning algorithm to identify suspects from their driver’s license.

We use face recognition to categorize images from a huge number of images.

3. Motivation

Nowadays capturing image is a popular trend. People capture a lot of images while they are visiting a place or attend a function. Very often an official photographer capture images for this purpose. Sometimes many volunteer photographer also participates in this work. But the problem arises when they want to person-wise categorize that huge amount of images. It is very time wasting and need a lot of attention. Sometimes due to lack of attention, many important images do not classify in the right position.

So, we try to implement a system which can detect a specific person from an image and can move that images which include the person into another folder or directory.

4. Methodology

Actually, face recognition is not a single algorithm. Its procedure can be divided into four parts –

1. First, look at a picture and find all the faces in it
2. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person.
3. Third, be able to pick out unique features of the face that you can use to tell it apart from other people— like how big the eyes are, how long the face is, etc.

4. Finally, compare the unique features of that face to all the people you already know to determine the person's name.

To implement these steps we use different algorithms. We try to describe these algorithms in short.

4.1 Finding all the faces

The first step is face detection. Face detection is a great feature for cameras. When the camera can automatically pick out faces, it can make sure that all the faces are in focus before it takes the picture. But we'll use it for a different purpose — finding the areas of the image we want to pass on to the next step in our pipeline. Here we use Histogram of Oriented Gradients (HOG) for this purpose.

To find faces in an image, we'll start by making our image black and white because we don't need color data to find faces. Then we'll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it. Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker: If you repeat that process for every single pixel in the image, you end up with every pixel being replaced by an arrow. These arrows are called *gradients* and they show the flow from light to dark across the entire image: we'll break up the image into small squares of 16x16 pixels each. In each square, we'll count up how many gradients point in each major direction (how many point up, point up-right, point right, etc…). Then we'll replace that square in the image with the arrow directions that were the strongest.

The end result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way:

To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces.

4.2 Posing & Projecting all the faces

we isolated the faces in our image. But now we have to deal with the problem that faces turned different directions look totally different to a computer. To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. The basic idea is we will come up with 68 specific points (called *landmarks*) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face.

4.3 Encoding Faces

We use Deep Convolutional Neural Network for encoding the specific face's data. So that our system can detect every unique face. we are going to train to generate 128 measurements for each face.

4.4 Finding the person's name from the encoding Encoding Faces

Now we have to do is find the person in our database of known people who have the closest measurements to our test image. We'll use a simple linear Support Vector Machine (SVM). We need to do is train a classifier that can take in the measurements from a new unknown or untrained image and tells which known person is the closest match.

5. Implementation

For implementation, we use openface and PyQt4.

OpenFace is a Python and Torch (another library) implementation of face recognition with deep neural networks.It is an open source library that rivals the performance and accuracy of proprietary models. While OpenFace is only a couple of years old, it's been widely adopted because it offers levels of accuracy similar to facial recognition models found in private state-of-the-art systems such as Google or Facebooks face recognition systems.

5.1. Openface install

It is better to install openface in Linux and OSX only. Openface maintains huge dependency, so it may create trouble while installing. Again, Openface is not a single library, it is combination of some libraries There are different procedures to setup openface . Below is one of them which is from source.

This project uses python2 because of the opencv and dlib dependencies. With pip2, then install numpy, pandas, scipy, scikit-learn, and scikit-image.

1. $ sudo pip install numpy

2. $ sudo pip install cython

3. $ sudo pip install scipy

4. $ sudo pip install scikit-learn

5. $ sudo pip install scikit-image

6. $ sudo pip install pandas

5.1.1 Opencv install

Then install opencv

```
$ sudo apt-get install build-essential
$ sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
$ sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev
libjasper-dev libdc1394-22-dev
```

changing the directory and downloading opencv

```
$ cd /usr/src
$ sudo git clone https://github.com/opencv/opencv.git
$ cd ~/opencv
$ sudo mkdir release
$ cd release
$ sudo cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
$ sudo make
$ sudo make install

$ pkg-config --modversion opencv
```

Commands For installing dlib

```
$ mkdir -p ~/src
$ cd ~/src
$ tar xf dlib-18.16.tar.bz2
```

$ cd dlib-18.16/python_examples

$ mkdir build

!--before proceeding further chk for the boost error--!

5.1.2 Boost install

For Boost Installation to fix error (boost error)

$ wget -O boost_1_62_0.tar.gz http://sourceforge.net/projects/boost/files/boost_1_62_0.tar.gz/download

$ tar xzvf boost_1_62_0.tar.gz

$ cd boost_1_62_0/

$ sudo apt-get update

$ sudo apt-get install build-essential g++ python-dev autotools-dev libicu-dev build-essential libbz2-dev libboost-all-dev

$ ./bootstrap.sh --prefix=/usr/local

############Unicode/ICU support for Boost.Regex?... not found.#############

Then build it with:

$ ./b2

and eventually install it:

$ sudo ./b2 install

!--Now Continue Further--!

After boost, Accessing the directory

```
$ cd ~/src
$ cd dlib-18.16/python_examples
$ cd build
```

now the leftovers commands

```
$ cmake ../../tools/python
$ cmake --build . --config Release
$ sudo cp dlib.so /usr/local/lib/python2.7/dist-packages
```

5.1.3 Torch install

Commands For installing Torch

Downloading Torch

```
$ git clone https://github.com/torch/distro.git ~/torch --recursive
$ cd ~/torch
$ bash install-deps
$ ./install.sh
```

```
$ sudo apt-get install luarocks
TORCH_LUA_VERSION=LUA52 ./install.sh
```

--For permission issues or errors with luarocks type the following commands--

```
$ sudo chmod -R 777 ~/opencv
$ sudo chmod -R 777 ~/torch
```

--For permission issues --

5.1.4 Packages install via Luarocks

packages to download via luarocks for Torch

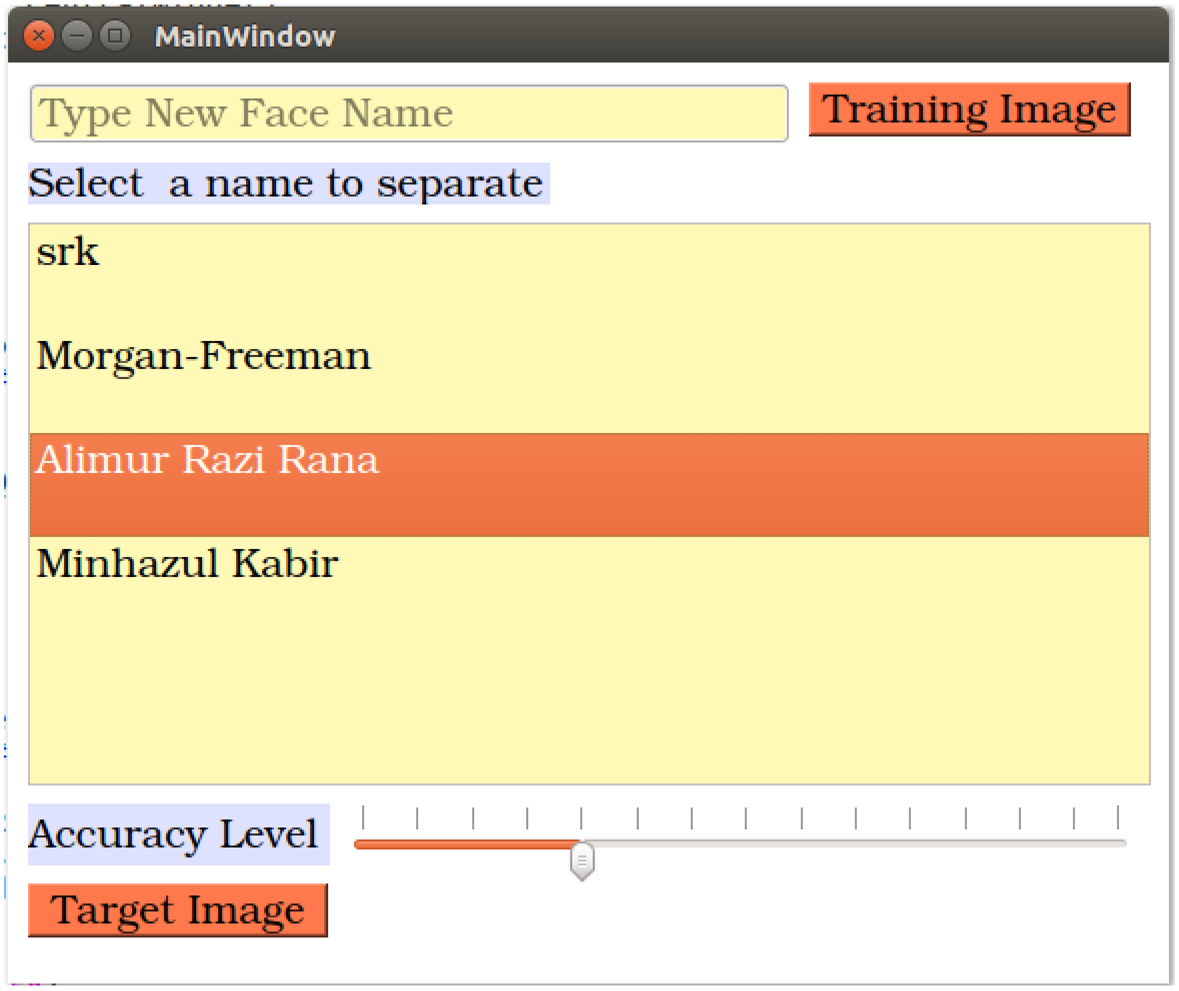
```
$ luarocks install dpnn
$ luarocks install nn
$ luarocks install optim
$ luarocks install csvigo
$ luarocks install cutorch and cunn (only with CUDA)
$ luarocks install fblualib (only for training a DNN)
$ luarocks install tds (only for training a DNN)
$ luarocks install torchx (only for training a DNN)
$ luarocks install optnet (optional, only for training a DNN)
```

5.1.5 PyQt4 install

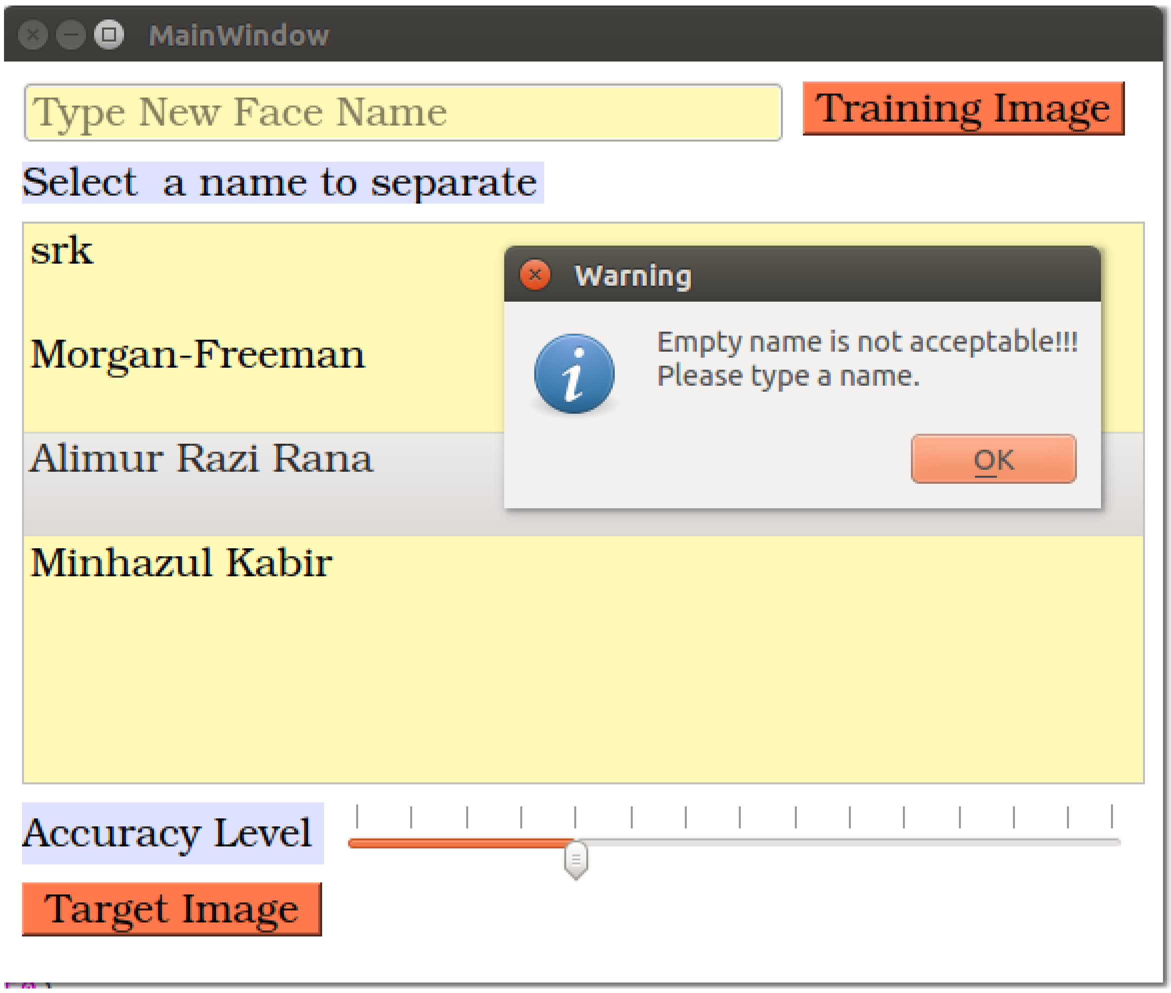
For user Interface we use PyQt4. It can be installed in this way

```
pip install PyQt4 4.11.4 cp36 cp36m win_amd64.whl
```

6. Features of the project

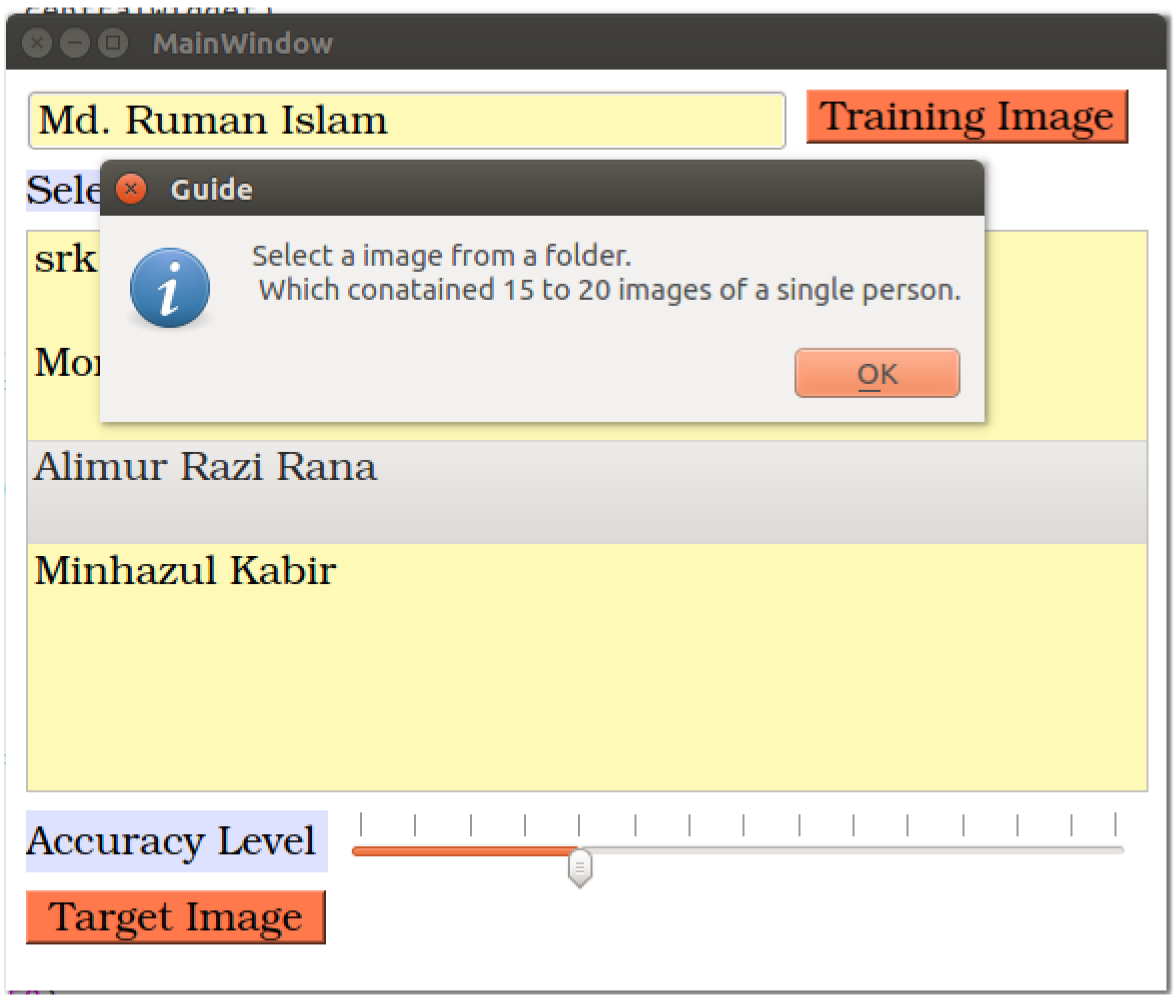


This is our project's user interface. The name listed are the person whose image are trained and the system can detect them.



Empty name is not acceptable!!!
Please type a name.

If we want to add a new person to our list , then at first his\her name must be added.


MainWindow

Md. Ruman Islam

Training Image

Sele

srk

Mo

Alimur Razi Rana

Minhazul Kabir

Accuracy Level

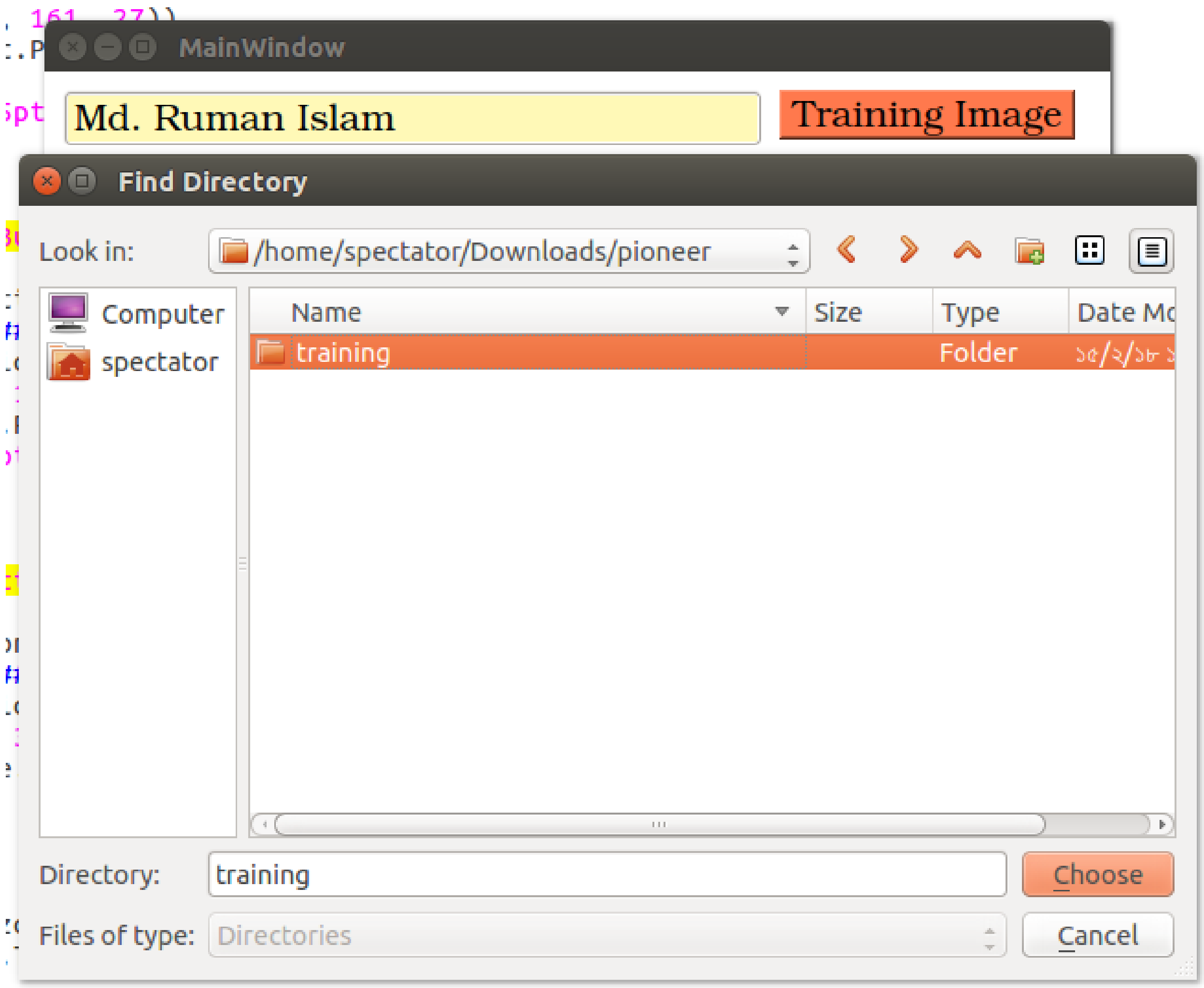
Target Image

Guide

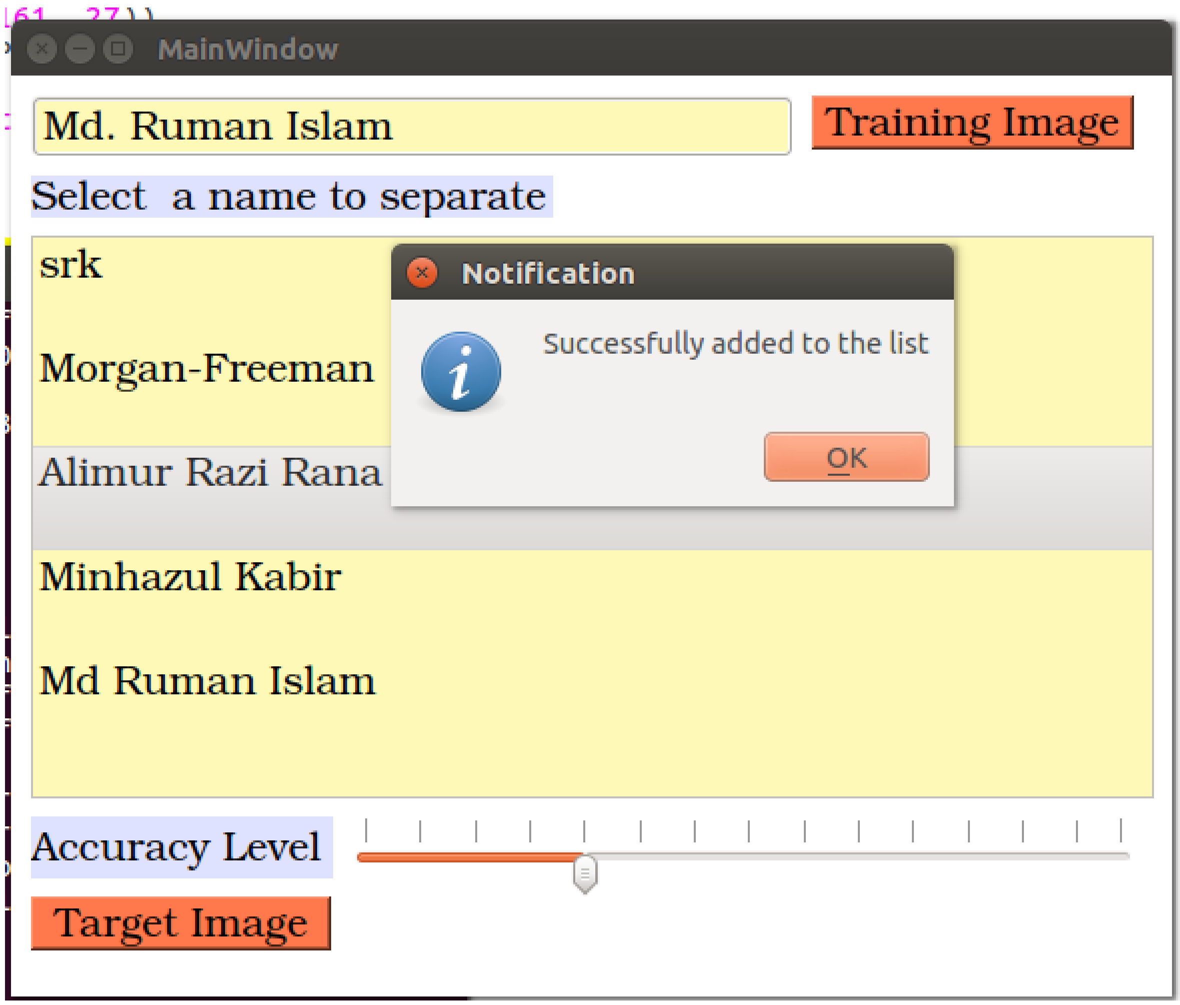
Select a image from a folder.
Which conatained 15 to 20 images of a single person.

OK

Then select a folder which contains 15-20 single photos of this specific person.



A specific folder is selected for a person. With this images the app will be trained.


MainWindow

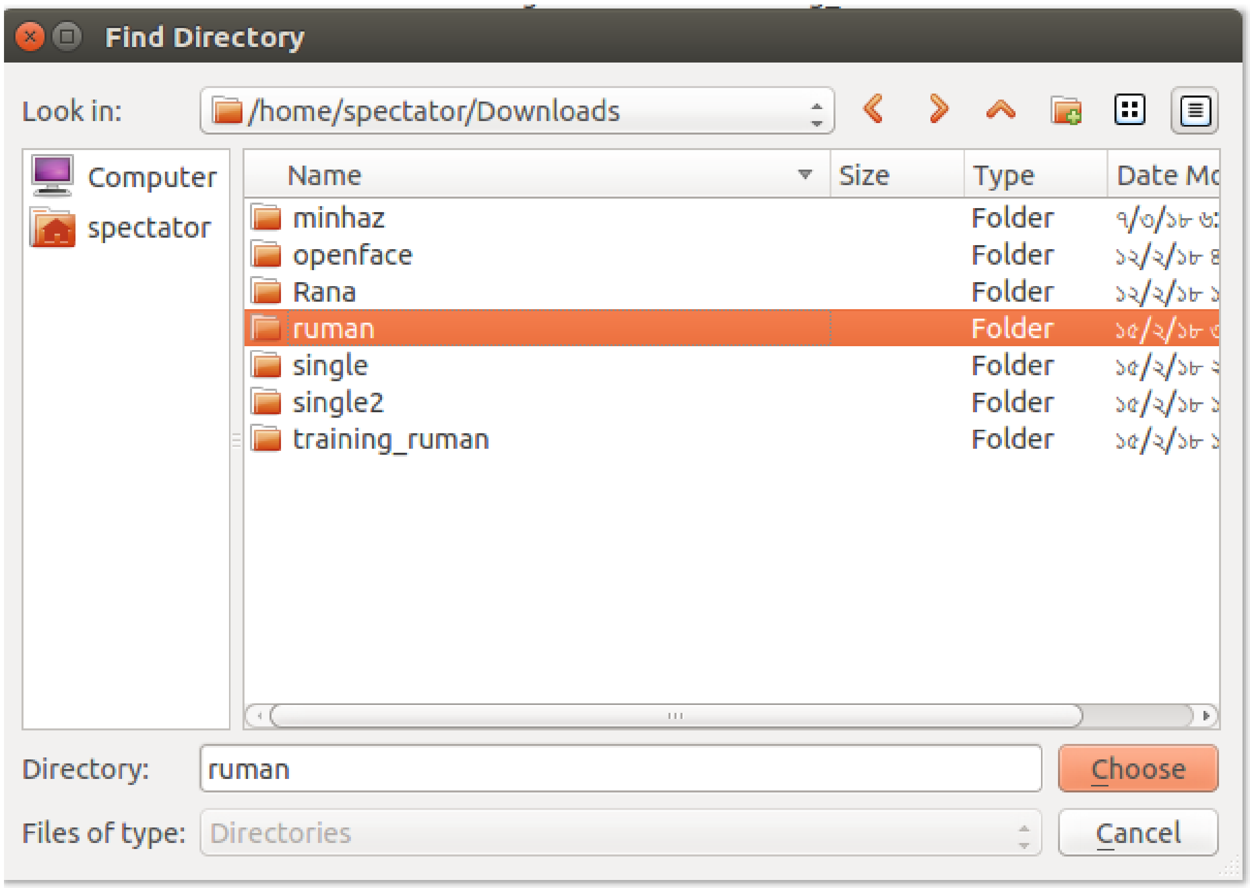
Md. Ruman Islam

srk
Morgan-Freeman
Alimur Razi Rana
Minhazul Kabir
Md Ruman Islam

Notification

Successfully added to the list

After training the person is successfully added to the list.



Find Directory

Look in: /home/spectator/Downloads

Computer
spectator

Name	Size	Type	Date Mo
minhaz		Folder	৭/৩/১৮ ৬:
openface		Folder	১২/২/১৮ ৪
Rana		Folder	১২/২/১৮ ১
ruman		Folder	১৫/২/১৮ ৩
single		Folder	১৫/২/১৮ ২
single2		Folder	১৫/২/১৮ ১
training_ruman		Folder	১৫/২/১৮ ১

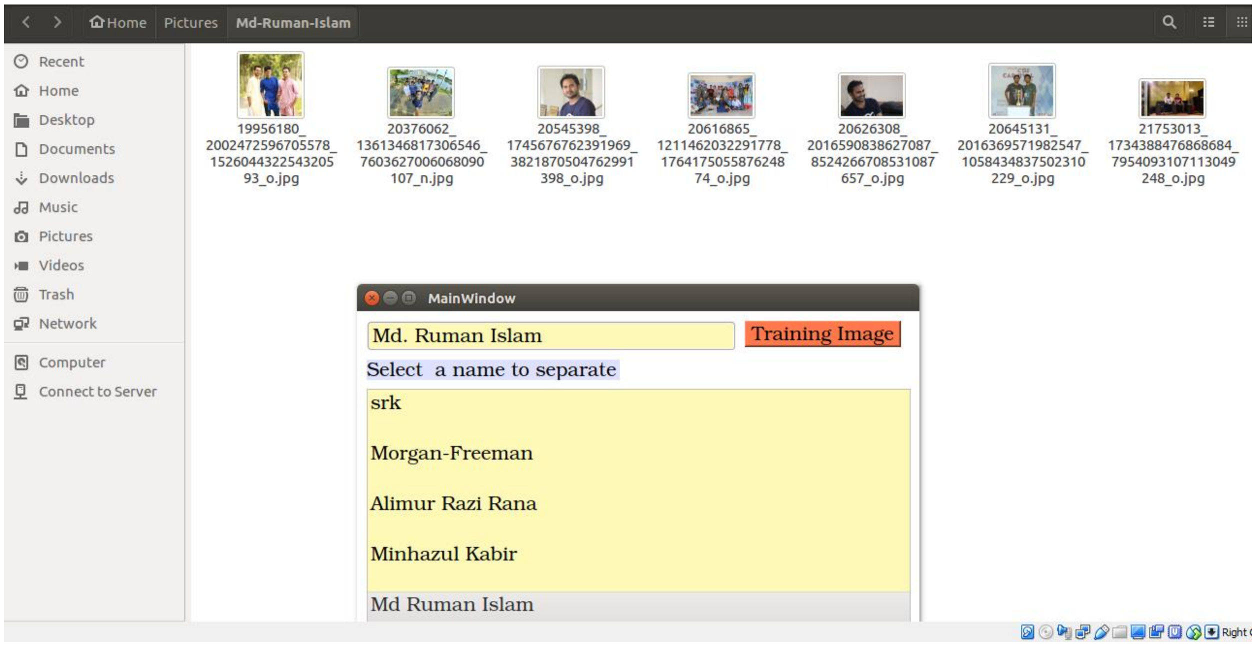
Directory: ruman

Files of type: Directories

Choose

Cancel

Then if anyone try to find out a person’s all images from a folder. Then he should click person’s name from list and target image button and then select the specific folder. Here a folder is selected for extract images.


Home
Pictures
Md-Ruman-Islam
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
19956180_2002472596705578_152604432254320593_o.jpg
20376062_1361346817306546_7603627006068090107_n.jpg
20545398_1745676762391969_3821870504762991398_o.jpg
20616865_1211462032291778_176417505587624874_o.jpg
20626308_2016590838627087_8524266708531087657_o.jpg
20645131_2016369571982547_1058434837502310229_o.jpg
21753013_1734388476868684_7954093107113049248_o.jpg
MainWindow
Md. Ruman Islam
Training Image
Select a name to separate
srk
Morgan-Freeman
Alimur Razi Rana
Minhazul Kabir
Md Ruman Islam

And we see that all the images of Md. Ruman Islam is shifted to another folder.