HomeWork 7/8 :

Face Recognition

Deep Learning for Advance Robotics Perception

Author:
Aayush Shah
Sanjuksha Nirgude
Dharini Dutia
Harsh Pathak
Namrita Madhusoodanan
Kevin Ducharme

Supervisor:
Dr. Carlos Morato

November 6, 2017

Description of files in the submission:

1. Vid2Img.m- Matlab script to extract images from videos.
2. create_dataset.py- Python script for creating the test and validation folders. The image files are shuffled in a random order and placed into these folders according to the specified train-validation split ratio.
3. CNN1.py- Python code(Keras+Tensorflow) defines the model of the best configuration, along with the code to train and test the model.
4. visualizing.py- Python code to visualize the filters of the developed model, given an input test image.
5. model_predict_face.ipynb- Code to view prediction labels for fresh test images.

Introduction

In this homework, we are using videos taken by six members of our group with 5 different backgrounds, and creating an image dataset out of that. Having parsed the imaged from a video we got a dataset with high correlation. Our challenge is to apply various techniques learned in class to make our dataset more efficient for a better learning with low bias and low variance. Moreover, having taken all the videos from cameras with different specifications. We, had to make our pre-processing unit more robust to remove extra images and use images with different resolutions.

In this assignment, for image classification we are using Convolutional Neural Networks. Since we have a small dataset with has high correlation, hence we incorporated some techniques to reduce overfitting. We are using a very small convnet with few layers and few filters per layer, alongside data augmentation and dropout.

Preprocessing and Data Handling:

1. First, we collect videos of all our group members in 5 different backgrounds, and extract approximately 1800 images per video. This is done using the matlab script Vid2Img.m. This results in a dataset consisting of around 9000 images per person. The images are resized to 108*192 for easier transfer of data.
2. Next, a train and validation folder is created from the dataset. For this purpose, a python script “create_dataset.py”, is run. This does the job of splitting the dataset into train and validation folders in a random fashion, which is essential for classification. The script is included in the submission.

Methodology :

We built a small Convolutional Neural Network with few filters per layer. Since, our dataset is small and highly correlated we used Data Augmentation in the data-preprocessing module of the assignment to reduce overfitting. We use "ImageDataGernerator", which is an in-built function in keras. This function is used to preprocess the images, convert them to tensors and augment the data. Further, our training dataset consists of approximately 5000-6000 images of each person in the group. The validation dataset consists of 3000 images of each person. In data augmentation the image is rotated, shifted, rescaled, sheared, zoomed and flipped to create a varied dataset.

Layer Topology:

1. The first trial consisted of a model with the following summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 72, 72, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 34, 34, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                1183808
_________________________________________________________________
```

```
activation_4 (Activation)    (None, 64)              0

dropout_1 (Dropout)          (None, 64)              0

dense_2 (Dense)              (None, 6)               390

activation_5 (Activation)    (None, 6)               0
```

The results obtained were:

```
Epoch 1/50
125/125 [==============================] - 264s - loss: 0.8883 - acc: 0.6930 - val_loss:
0.1239 - val_acc: 0.9762

Epoch 2/50
125/125 [==============================] - 124s - loss: 0.4191 - acc: 0.8815 - val_loss:
0.0876 - val_acc: 0.9675


Epoch 3/50
125/125 [==============================] - 127s - loss: 0.2949 - acc: 0.9185 - val_loss:
0.0180 - val_acc: 0.9975

Epoch 4/50
125/125 [==============================] - 124s - loss: 0.2154 - acc: 0.9435 - val_loss:
0.0150 - val_acc: 0.9962

Epoch 5/50
125/125 [==============================] - 121s - loss: 0.1726 - acc: 0.9520 - val_loss:
7.6551e-04 - val_acc: 1.0000
```

Since a validation accuracy of 1 was achieved, it can be concluded that only 6 epochs are necessary.

This model used the optimizer 'adam'.

2. The second trial used the same model architecture with the optimizer 'rmsprop'. These were the results obtained:

```
Epoch 1/6
125/125 [==============================] - 85s - loss: 1.0477 - acc: 0.6255 - val_loss:
0.4749 - val_acc: 0.8350
Epoch 2/6
125/125 [==============================] - 80s - loss: 0.4893 - acc: 0.8530 - val_loss:
0.0980 - val_acc: 0.9900
Epoch 3/6
```

```
125/125 [==============================] - 80s - loss: 0.2792 - acc: 0.9150 - val_loss:
0.0309 - val_acc: 0.9838
Epoch 4/6
125/125 [==============================] - 78s - loss: 0.2049 - acc: 0.9405 - val_loss:
0.0106 - val_acc: 0.9988
Epoch 5/6
125/125 [==============================] - 76s - loss: 0.1367 - acc: 0.9510 - val_loss:
0.0061 - val_acc: 0.9988
Epoch 6/6
125/125 [==============================] - 78s - loss: 0.1120 - acc: 0.9630 - val_loss:
0.0093 - val_acc: 0.9962
```

This model trains faster, has less loss and more accuracy during training, and seems to do a better job of countering overfitting.

3. For the model above, inputs were given to visualize the filters. The results were-

Input Image:

Visualization for conv2d_1

Visualization for conv2d_2

Visualization for conv2d_3

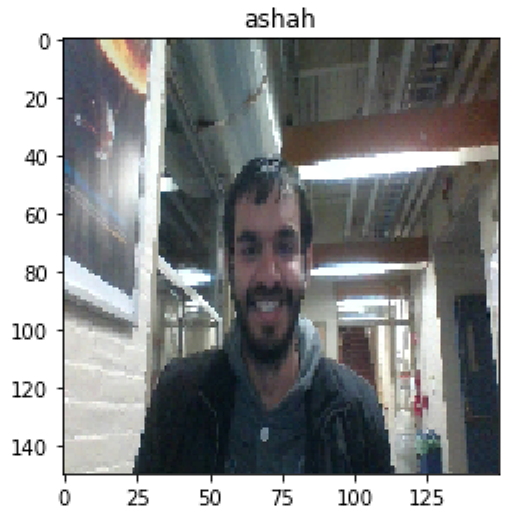
4. The model was also tested using completely new images in the validation folder, which were extracted from new videos.These new images are never seen by the model during training.
The results were-
Epoch 6/6

```
125/125 [==============================] - 57s - loss: 0.1202 - acc: 0.9655 - val_loss: 0.0014 - val_acc: 0.9988
```

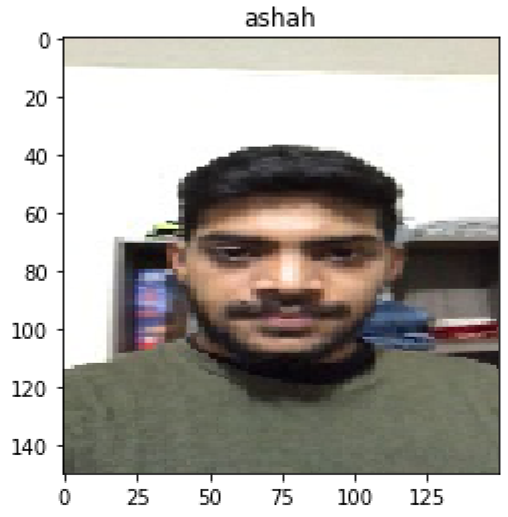
Results for Fresh Images which were not the part of data set :

Our Model was able to correctly classify 12/17 ~ fresh images that were completely different from training data.

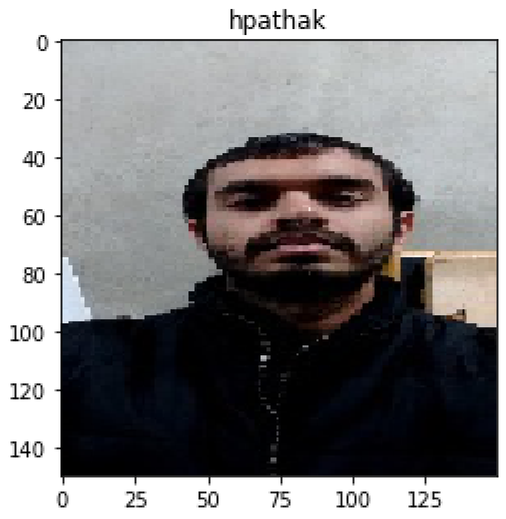
```
1/1 [==============================] - 0s
True Label : hpathak
```



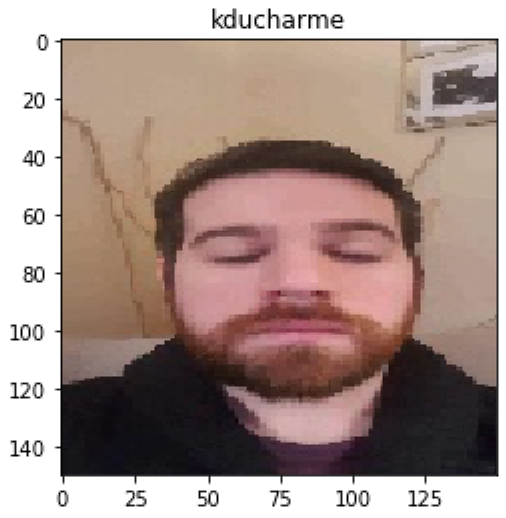
```
1/1 [==============================] - 0s
True Label : ashah
```



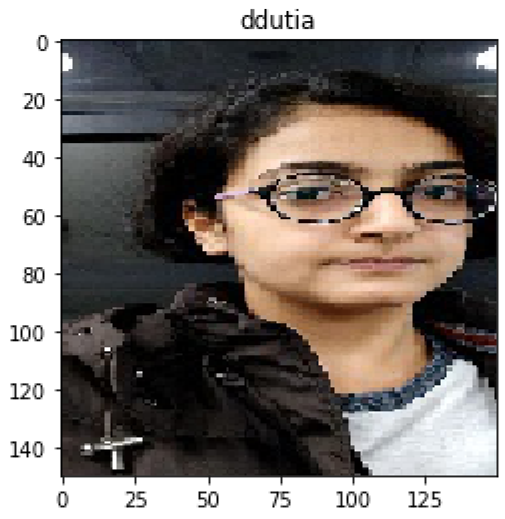
```
1/1 [==============================] - 0s
True Label : hpathak
```



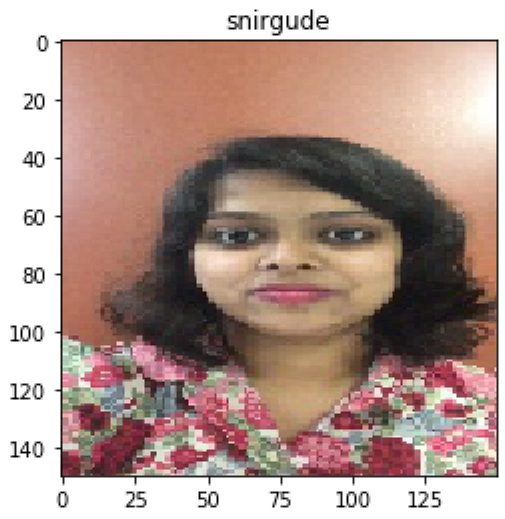
```
1/1 [==============================] - 0s
True Label : kducharme
```



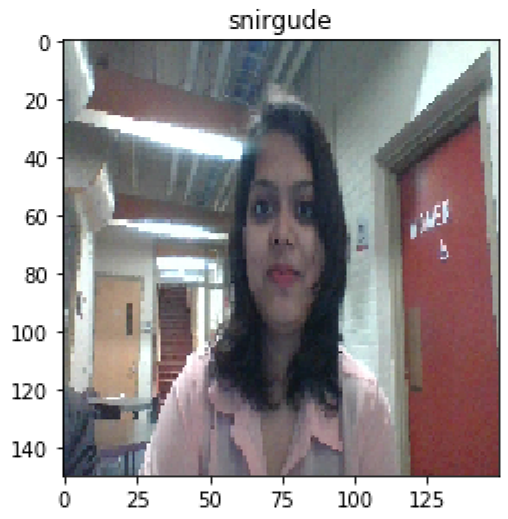
```
1/1 [==============================] - 0s
True Label : ddutia
```



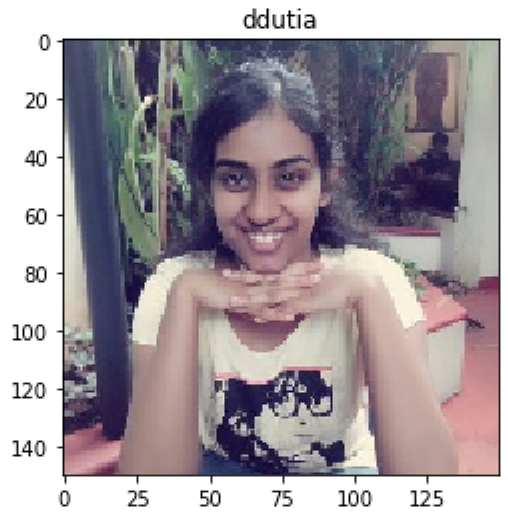
```
1/1 [==============================] - 0s
True Label : snirgude
```



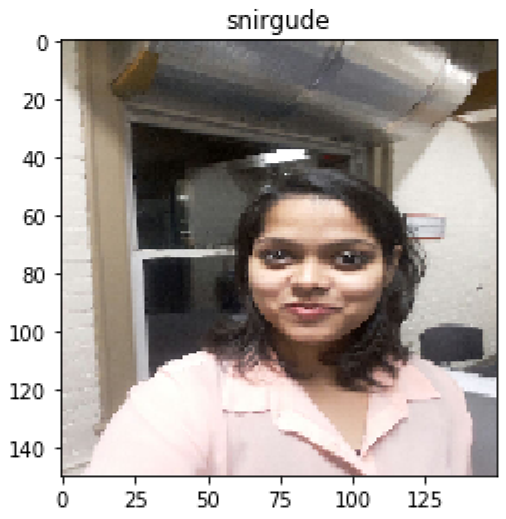
```
1/1 [==============================] - 0s
True Label : snirgude
```



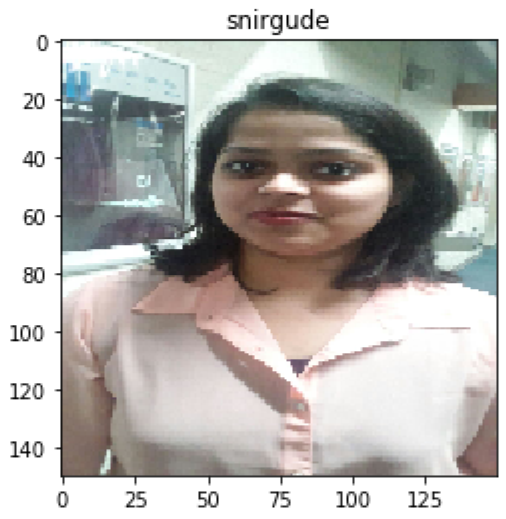
```
1/1 [==============================] - 0s
True label : namrita
```



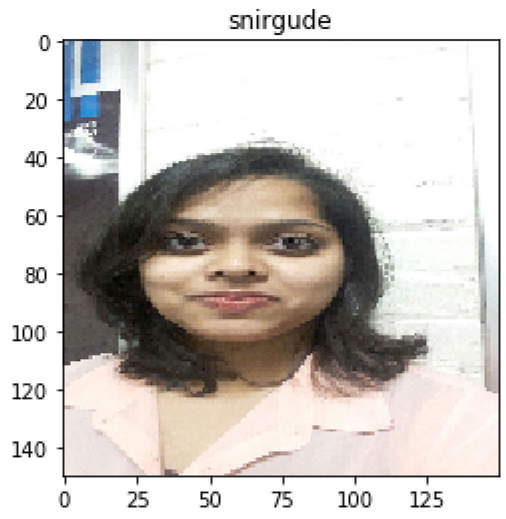
```
1/1 [==============================] - 0s
True Label : snirgude
```



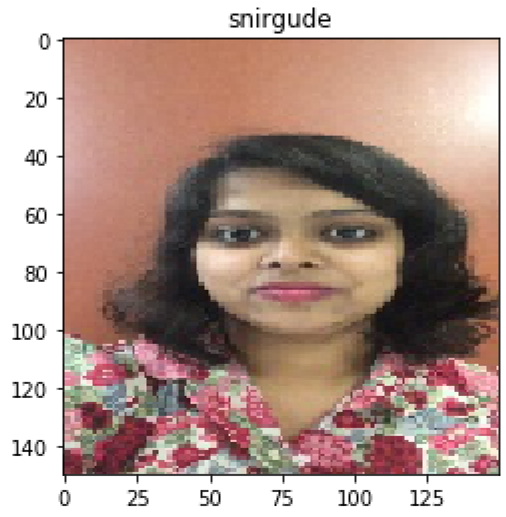
```
1/1 [==============================] - 0s
True Label : snirgude
```



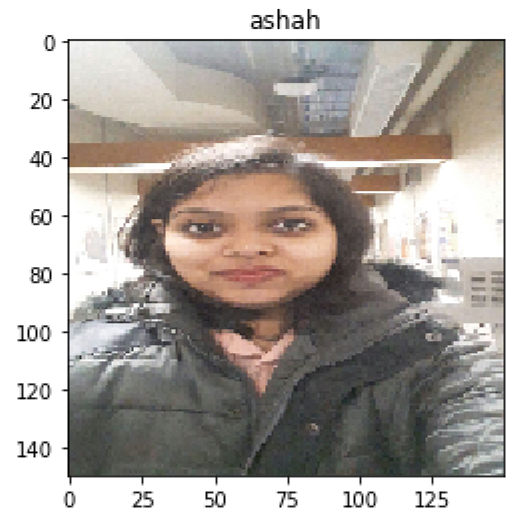
```
1/1 [==============================] - 0s
True Label : snirgude
```



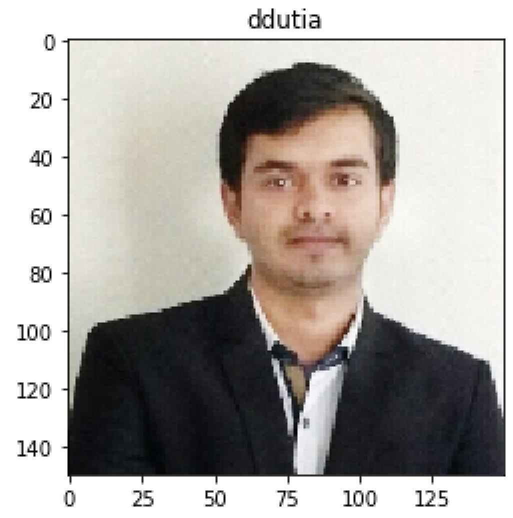
```
1/1 [==============================] - 0s
True Label : snirgude
```



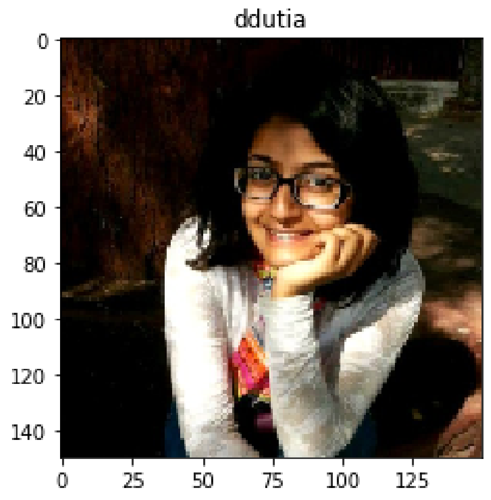
```
1/1 [==============================] - 0s
True Label : snirgude
```



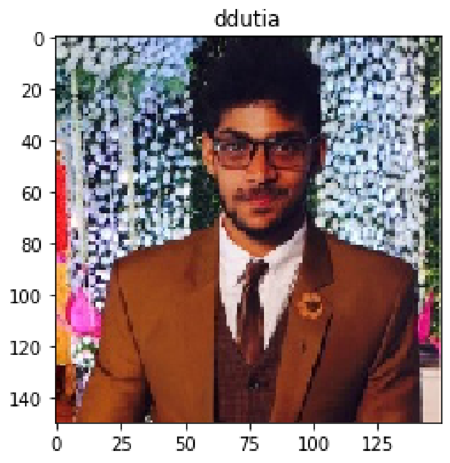
```
1/1 [==============================] - 0s
True Label : hpathak
```



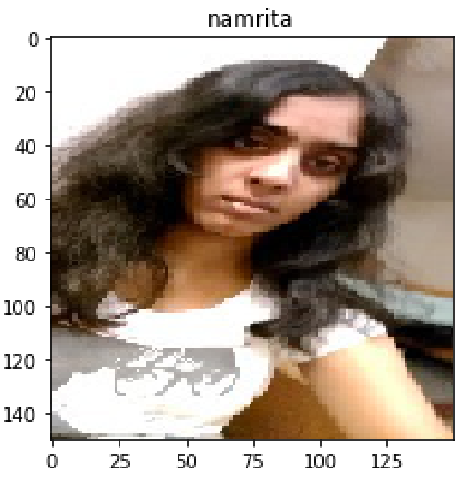
```
1/1 [==============================] - 0s
True label : ddutia
```



```
1/1 [==============================] - 0s
True Label : ashah
```



```
True Label : namrita
```



Conclusions:

The model developed gives a validation accuracy of about 99.8%. This high value is achieved because the data used for training and validation are obtained from videos,resulting in highly correlated data.

.

References :

1. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html