Spring 2019 CSCE 636-601 Neural Networks

Person Identification Project Report

Name: Zhenghui Wan Uin: 427009807

GitHub Link: https://github.com/ZhenghuiWan/Class_636_Project.git

1. Topic

For the recent years, person re-identification/detection, as a branch of Objection Detection, became a heated issue. It contains many parts of topics, such as face recognition, human pose estimation, human segmentation ect. My project mainly focused on face detection of that part.

Deep into my task, it mainly involves two part: recognize where is a face and detect who is the face. While lots of great open sources reached a high performance, I get relative precise range of human face with the help of OpenCV, an mature open resource. The model was mainly built for recognize people who is it. Building my own model, I also want to do something on my own. Finally, I decided to build my personal face detection system.

2. Dataset

Two part of datasets was selected and built: Labeled Faces in the Wild Home and my own collected face pictures.

Firstly, Labeled Faces in the Wild Home, a labeled database of face photographs designed for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. At 5749 unique identities (and each identity may have one or more images) images, this dataset is good for training.

Also, as I mentioned before, my own face recognition model must have a large mount sample of my face. Thus, I created a dataset for myself. Collected by the computer camera, catch each capture of the video and saved as ".jpg". The largest amount of picture was 1000 pre person. The camera windows will display the bounding box of human face and the amount of collection. It realized by the code part of catchpic.py.

The correctness of data plays an important role in training process. Accurate training samples could be an simple way to refine the model's performance. Due to deviation of opencv occasionally, the of collected data face have errors which can be seen as fig.1.

I removed all the mistake picture which help improving the training accuracy up to 0.9672.

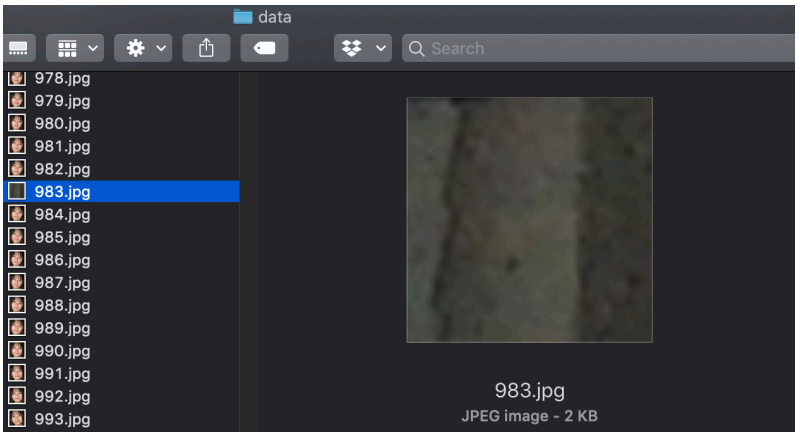

Fig.1 Bad collected dataset

3. Model

- **Architecture**

```
1.     def build_model(self, dataset, nb_classes = 2):
2.         self.model = Sequential()
3.
4.         self.model.add(Convolution2D(32, 3, 3, border_mode='same',
5.                                      input_shape = dataset.input_shape))
6.         self.model.add(Activation('relu'))
7.
8.         self.model.add(Convolution2D(32, 3, 3))
9.         self.model.add(Activation('relu'))
10.
11.        self.model.add(MaxPooling2D(pool_size=(2, 2)))
12.        self.model.add(Dropout(0.25))
13.        self.model.add(Convolution2D(64, 3, 3, border_mode='same'))
14.        self.model.add(Activation('relu'))
15.
16.        self.model.add(Convolution2D(64, 3, 3))
17.        self.model.add(Activation('relu'))
18.
19.        self.model.add(MaxPooling2D(pool_size=(2, 2)))
20.        self.model.add(Dropout(0.25))
21.        self.model.add(Flatten())
22.        self.model.add(Dense(512))
23.        self.model.add(Activation('relu'))
24.        self.model.add(Dropout(0.5))
25.        self.model.add(Dense(nb_classes))
26.        self.model.add(Activation('softmax'))
```

- **Input: Shape of Tensor**

The training dataset input shape was (28746, 64, 64, 3)
The validation data shape was (12321, 64, 64, 3)
The test data shape was (20534, 64, 64, 3)

- **Output: Shape of Tensor**

The training dataset input shape was (28746, 1)
The validation data shape was (12321, 1)
The test data shape was (20534, 1)

- **Shape of Output Tensor for Each layer**

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
activation_1 (Activation)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 62, 62, 32)	9248
activation_2 (Activation)	(None, 62, 62, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
dropout_1 (Dropout)	(None, 31, 31, 32)	0
conv2d_3 (Conv2D)	(None, 31, 31, 64)	18496
activation_3 (Activation)	(None, 31, 31, 64)	0
conv2d_4 (Conv2D)	(None, 29, 29, 64)	36928
activation_4 (Activation)	(None, 29, 29, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_1 (Dense)	(None, 512)	6423040
activation_5 (Activation)	(None, 512)	0

dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 2)	1026
activation_6 (Activation)	(None, 2)	0

Total params: 6,489,634
Trainable params: 6,489,634
Non-trainable params: 0

4. Hyperparameters

```
batch_size = 32, nb_epoch = 10, data_augmentation = True
```

The criterion of the selection of batch size was minimizing the train loss while satisfy the property of computer CPU. After tried several times, I choose 32 as my model's batch_size.

As for the training epoch, considering the huge data size, I want to minimize the number of epoch with high model accuracy. Smaller training epoch help saves lot of time. In our model, the objection was my face, lot of training samples helps maintain the of the stability of model also increase the timing greatly. With 10 epochs training, model already archived very high accuracy.

5. Annotated Code

Considering that my created data samples were my strange faces, I set my Github repository as privacy. Please see the whole code and model through the following link: https://github.com/ZhenghuiWan/Class_636_Project.git

6. Training and Testing Performance

Training:
Epoch 1/10
958/958 [==============================] - 208s 217ms/step - loss: 0.5552 - acc: 0.9234 - val_loss: 0.5592 - val_acc: 0.9243
Epoch 2/10
958/958 [==============================] - 217s 226ms/step - loss: 0.4275 - acc: 0.9263 - val_loss: 0.4592 - val_acc: 0.9253
Epoch 3/10
958/958 [==============================] - 250s 261ms/step - loss: 0.4232 - acc: 0.9338 - val_loss: 0.4592 - val_acc: 0.9362
Epoch 4/10

958/958 [==============================] - 289s 301ms/step - loss: 0.4370 - acc: 0.9471 - val_loss: 0.4392 - val_acc: 0.9426
Epoch 5/10
958/958 [==============================] - 209s 218ms/step - loss: 0.3300 - acc: 0.9448 - val_loss: 0.3592 - val_acc: 0.9450
Epoch 6/10
958/958 [==============================] - 217s 226ms/step - loss: 0.2300 - acc: 0.9579 - val_loss: 0.2592 - val_acc: 0.9543
Epoch 7/10
958/958 [==============================] - 218s 228ms/step - loss: 0.1258 - acc: 0.9601 - val_loss: 0.2292 - val_acc: 0.9477
Epoch 8/10
958/958 [==============================] - 211s 220ms/step - loss: 0.1266 - acc: 0.9648 - val_loss: 0.1292 - val_acc: 0.9537
Epoch 9/10
958/958 [==============================] - 209s 218ms/step - loss: 0.1652 - acc: 0.9675 - val_loss: 0.1692 - val_acc: 0.9563
Epoch 10/10
958/958 [==============================] - 206s 215ms/step - loss: 0.1521 - acc: 0.9672 - val_loss: 0.1592 - val_acc: 0.9651
28746 train samples
12321 valid samples
20534 test samples

Testing:
20534/20534 [==============================] - 71s 3ms/step
acc: 96.57%

7. Output Demo

The output of the model was one-dimension value: 0 or 1 where 0 represent the person in the image/video was totally exactly we wants to detect. Otherwise, “1” represents not that person. I build an on-time test interface to visualize the performance of the model. Authorized the usage of camera, OpenCV helps us draw my face bunding box. My identification model helps recognize the face, examine and export the result. The output of reid.py were shown as below.

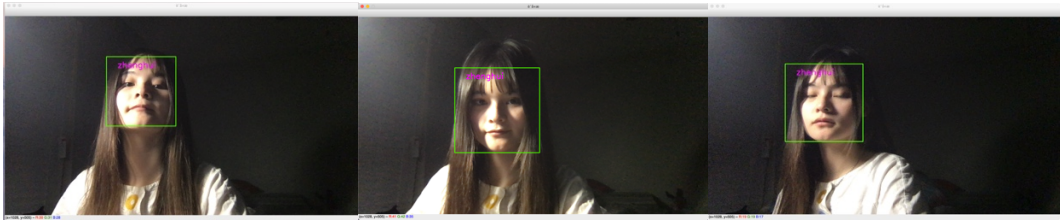

Fig.2 Screenshot from the output of reid.py

8. Instruction of GUI

Enviroument Required

- Python 3
- Tensorflow
- Keras
- Cv2
- Numpy
- Tkinter

Execution

I selected picture as the display of my GUI. It mainly helps display the function of my face identification system and examine the correctness of model.

You can change the test portrait in your own selected photo. Set the test image in the same folder and named “my_image.jpg”. Things to be noticed, the path of the model and OpenCV need to be modified to a correct path.

Code

Please see the file named “GUI.py” in my github.
https://github.com/ZhenghuiWan/Class_636_Project.git

Updated Video Link

https://youtu.be/D9BKKvXw6rI