Coding Exercise Reflection Report

Cat-Hipsterizer

Chae-uk Lim
CSCE 4201
Apr. 27, 2021

1. Introduction

Computer Vision is one of the most widely used AI technologies nowadays. It has been developed rapidly and started to be applied to our lifestyles. I feel so grateful that I could learn about the brief outline of AI technology from the great institution. However, I wanted to know not only the theoretical part of the study, but also the practical way of being applied to our society, therefore, the mini project had been chosen to be done in this semester, which is called 'Cat-Hipsterizer'.

The main goal of this project is to make a machine learn to recognize cats' faces and put glasses or sunglasses on them. The main idea was from one Youtubuer's video, he used two models, one for recognizing the cats' faces and cutting their faces only from the image, the other for setting landmarks on them, in a cascaded form. After the programs read and set the landmarks on their faces, a simple function finds out the eyes and makes the cats 'Hipsters' by putting sunglasses on them.

During the process, I found out that coding the programs using Python requires more skills, therefore, I decided to have this as a coding exercise, not a mini project. Since I was not familiar with Python, I could realize that an exercise by following the codes and analyzing them would be more helpful to myself, instead of adding more challenges to have my own project.

This report will go over the purposes of the exercise, the details of my progress, and what I've learned through this exercise.

2. Cat Hipsterizer

The model of the exercise is from a video of one Korean Youtuber.[1] The video is showing how to put glasses on cats' face images by using machine learning. The original idea he took was from an article of Dlib C++ Library, "Hipsterize Your Dog With Deep Learning"[2]. However, the Youtuber found that the program from the library was not perfect, therefore, he edited the codes in his way and introduced how he did by the video.

2.1 Developing Environment and Data Set

The technologies used in this exercise are following:

- Python
- PyCharm
- Conda
- Pandas
- OpenCV
- Keras
- Dlib
- Numpy
- Mobilenet_v2

---

[1] https://www.youtube.com/watch?v=7_97SlAigPs&list=PL-xmlFOn6TULrmwkXjRCDAas0ixd_NtyK&index=51&t=66s

[2] http://blog.dlib.net/2016/10/hipsterize-your-dog-with-deep-learning.html

The data set used for training, testing, and validation is from Kaggle[3]. There are 7 directories full of cats' face images, the number of data is over 9,000. Moreover, there are .cat files for each image that contains the coordinates of 9 landmarks of each.

2.2 Process

The image inputs of the model all have different sizes. Therefore, before the machine starts its process, a system manager needs to resize the images. The process is called 'preprocessing'. The preprocessing is necessary to expect a better performance from a machine. By cutting and filling the margin of images, the preprocessing file unifies the sizes of all images and passes to the file that trains the machine to learn.

Once the images are re-formatted, the machine will learn where the face of the cat is in the input images. However, the purpose of this exercise is not only detecting the cats' faces, but also putting glasses on them, two models are needed to perform its task. One is to detect the faces by using the numpy dataset, and the other is to draw a landmark on them. Once the machines are able to detect and draw its landmarks, another file will find the landmarks of two eyes and make them a "Hipster" by putting sunglasses on.

The number of earned datasets is 7. In this exercise, 6 of them will be used as train sets and only one will be used as a validation set.

After two models are trained with the datasets, the testing file will perform testing by applying one random cat's face image. The first model will detect its face and the other will mark the landmarks on the file. The testing file has another functionality that put glasses on the face by detecting the eyes using earned landmarks.

[3] https://www.kaggle.com/crawford/cat-dataset

3. The details of progress

Even the codes have been provided on the YouTuber's GitHub, it was not an easy task to follow his work. There are many libraries I've never used before, and even a lot of work was required on configuring the development environment.

3.1 Configuration

Before starting coding, the system configuration is necessary. Since to train a model, it requires a lot of new interpreters or libraries. Using an IDE can provide a much better environment to manage the libraries. I added the libraries and interpreters from the IDE (Pycharm, in my case), or installed them manually through terminal.

Conda is a great software to help you manage machine learning. Once you install the software from the Internet, Pycharm works with that. When you make a new project through the IDE, you may choose whether the project is based on Conda or not.

3.2 Preprocessing

If we unify the format of the inputs, the learning task will be much easier for machines. The efficiency of all parts will be highly increased and the accuracy as well. In this exercise, there is a preprocessing file for each model.

The following function is provided by the Youtuber. This function makes images into squares by giving a margin at where it needs and fills the margin with black color. The fixed size of the input is 244. Once the images are resized, the coordinates of each file are adjusted as the size has been changed. After then, the coordinate tuples are stored in a dictionary called 'dataset', which consists of three fields; 'imgs' for images, 'lmks' for landmarks, and 'bbs' for bounding boxes.

```
# Unify the sized of all input images
def resize_img(im):
    old_size = im.shape[:2]
    ratio = float(img_size) / max(old_size)
    new_size = tuple([int(x * ratio) for x in old_size])
    im = cv2.resize(im, (new_size[1], new_size[0]))
    delta_w = img_size - new_size[1]
    delta_h = img_size - new_size[0]
    top, bottom = delta_h // 2, delta_h - (delta_h // 2)
    left, right = delta_w // 2, delta_w - (delta_w // 2)
    new_im = cv2.copyMakeBorder(im, top, bottom, left, right, cv2.BORDER_CONSTANT,
                                value=[0, 0, 0])
    return new_im, ratio, top, left
```

For the secondPreprocessing.py, it creates a new bounding box that is bigger than one created by the firstPreprocessing.py because the model which will use the dataset will set the landmarks. By giving more margin to the bounding box makes the machine much easier to detect and set the landmarks on the faces.

3.3 Training

After the preprocessing is done, the dataset is passed to training files. What the training files do is building a model for each task and training them with the datasets. However, in this exercise, the pretrained model is used, which is MobileNetV2. The MobileNetV2 is a pretrained model provided by Google. It is suitable to be used at where the performance or battery performance of a computer is limited.

In this exercise, I've assigned 6 of 7 datasets as training datasets and only one for testing. By using numpy functions, I could assign each dataset as their purposes. The following screenshot is how I followed the Youtuber's step to distribute the datasets for training and testing.

```
x_train = np.concatenate((data_00.item().get('imgs'),
                          data_01.item().get('imgs'),
                          data_02.item().get('imgs'),
                          data_03.item().get('imgs'),
                          data_04.item().get('imgs'),
                          data_05.item().get('imgs')), axis = 0)
y_train = np.concatenate((data_00.item().get('bbs'),
                          data_01.item().get('bbs'),
                          data_02.item().get('bbs'),
                          data_03.item().get('bbs'),
                          data_04.item().get('bbs'),
                          data_05.item().get('bbs')), axis = 0)
print('Data reading finished.')

# Start building the model
print('Building models...')
x_test = np.array(data_06.item().get('imgs'))
y_test = np.array(data_06.item().get('bbs'))
```

The x_train is a sample question for machines to learn, and the y_train is a solution. Machines are supposed to learn how to detect the face and the landmarks by solving the problems and comparing its answer with the solution.

After then, we have to set the MobileNetV2, which is our pretrained model.

```
mobilenet_model = mobilenet_v2.MobileNetV2(input_shape=(img_size, img_size, 3),
                                           alpha=1.0,
                                           include_top=False,
                                           weights='imagenet',
                                           input_tensor=inputs,
                                           pooling='max')
```

The screen shot above is showing how I set the model in this exercise. The ‘include_top parameter’ is set as ‘False’ since the problem is a regression problem, not a classification problem, and the ‘pooling’ parameter is set as ‘max’ because this model will use the global max pooling.

```
net = Dense(128, activation='relu')(mobilenet_model.layers[-1].output)
net = Dense(64, activation='relu')(net)
net = Dense(output_size, activation='linear')(net)
```

The first dense has 128 layers with ‘ReLU’ function, the second dense has 64 with the same, and the dense has the layer of the number of output sizes for each training model with ‘linear’ function.

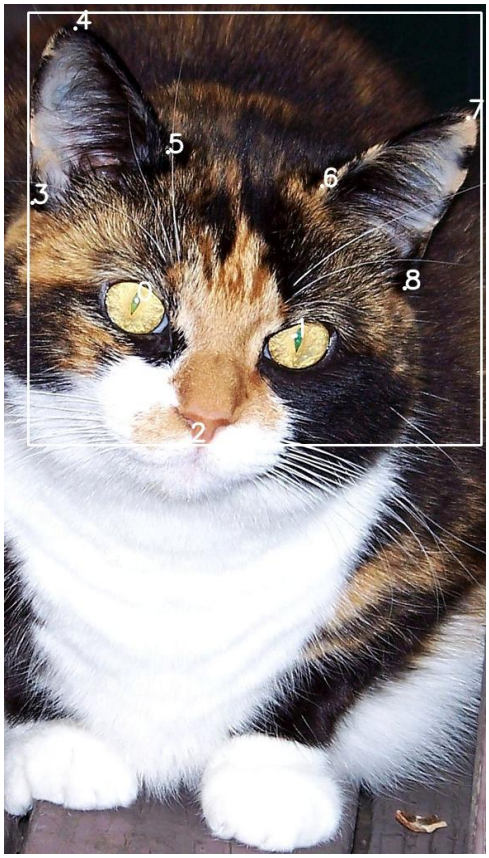
Once the setting is done, the training session starts. I used Adam optimizer and Mean Squared Error for loss while training.

It took a long time to train the models. Therefore, after I checked that my training files work, I replaced them with the models provided by the Youtuber.

3.4 Testing

After the models are trained, the testing file uses the models by implementing a sample image of a cat. The models detect the cat’s face in the image and mark the landmarks on each point. After all these processes are done, then the other functions that overlaying the glasses image on the cat image and finding the slope that the glasses should have.

The results are following:



4. Conclusion

I could learn practically of how Computer Vision technology is being used in our real life. When I was a freshman, one of my professors introduced computer science by comparing it with playing piano. What he wanted to emphasize was that all the students who are about to start studying this field need to know not only the theories, but also how to code. I am still grateful and agree with him. All the theoretical knowledge can be complete by exercising. Even though I learned about Computer Vision and AI, I didn't know what exactly the details of them really meant before I did this exercise. By this exercise, I could feel what all the components I've learned in this class really are.