Find the Phone Task with nearly 90% accuracy

Ye YUAN

(2017)450076

1356 Synergy, Irvine, CA

Python Libraries:

Sklearn, OpenCV, Numpy, matplotlib, Keras, Tensorflow

Code Execution:

The train_phone_finder.py file receives the path to training data directory to train the model. It generates a model.h5 file, which would be used later by the find_phone.py file to predict the location of phone. If the data directory is in the home directory:

Terminal Command:

```
> python train_phone_finder.py ~/find_phone
```

The find_phone.py takes the path to one image to predict the location of phone in that image

Terminal Command:

```
> python find_phone.py ~/find_phone_test_images/51.jpg
```

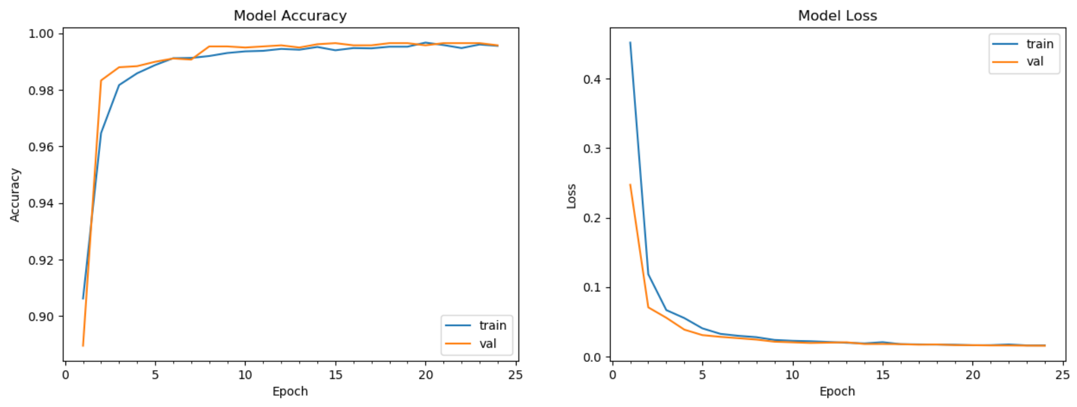
Approach:

In the beginning, the model I built a six-layer convolutional neural network. I also augmented the dataset by mirroring images to get a 4*size of original dataset and fit whole pictures into CNN. However, the neural network always faces either overfitting or underfitting problem. I think there are too main reasons: 1. The phone is too small compared with the background, which makes CNN hard to find appropriate features. 2. The dataset is too small for CNN to learn.

To solve the problem mentioned above, I learned from the method using to detect road signs in images that is to separate the overall task into two parts: 1. Build a CNN to predict the probability of whether it is a phone. 2. Get sliding windows of images and predict the probability of whether there is a phone

In the first part, for each image, I cropped the 44*44 phone patch and randomly rotate 50 times to get 50 patches of the phone and cropped 50 44*44 patches of background randomly. The overall dataset would be 12900. Then I split the dataset into training set (0.8) and validation set (0.2) and fit them into the CNN model with 2 Convolution Layer and 2 Dense Layer to get the model. The reason why I use the patch size to be 44*44 is that some phones are on the edge of the images that 44*44 is the largest patch with phone in the center I could get.

The following picture is the plot of accuracy and loss with number of epoches



In the second part, for each image, I use a sliding window with step of 4 pixels. For each window, I predict the probability of whether there is a phone and filter the windows with probability more than 0.7 and assumes that the contour in the window is the largest if the phone is in the center.

The accuracy of whole dataset is around 90% (I implemented the accuracy function in the train_phone_finder.py file. The running of accuracy function is too slow because I set the sliding window stride to be 4 pixels. That's why I didn't output the accuracy of the whole dataset)

Next steps:

1. It would be more helpful to apply more data augmentation methods such as shearing, perspective transformation and so on.
2. Some preprocessing methods could be used such as image denoise and dilation of edges before fitting images into the CNN
3. Implement pretrained YOLO model to do transfer learning
4. Split the image into several regions and skip the region with low IOU(intersection over union) to train and evaluate the model faster.