# Summary Report of Find Phone Task

*Email: linhai_li@hotmail.com*

**Development Environments:**

Python version: 3.6.5;

opencv package version: 4.1 (installed with pip install opencv-contrib-python);

keras on tensorflow-gpu backend;

Other packages: latest stable version

NOTE: if you get errors from opencv (i.e. cv2) package, it is either you are using older version or the contrib module is not included. Please uninstall the old one and reinstall.

**Problem analysis:**

This is a classification + localization problem. As long as the bounding box for the phone is found, the center is retrieved. Therefore, the key is to get the region of interest which in this problem is the region containing the phone.

**Possible solutions:**

1. Deep-learning method. Train a convolutional neural network (CNN) to get regions of interest on the whole image and pass to a full-connected neural network to get the center of phone. However, the small dataset will be insufficient to train a good deep learning CNN. Using transfer learning on pre-trained network may help, although I still do not expect this will work too well because of the small dataset. Therefore, I just tested the idea (see cnn_transfer.py) and did not tune it too much before I moved to the following ideas.
2. Image-processing method. EdgeBoxes is an excellent object proposal approaches by looking at the edges of objects on an image. The EdgeBoxes approach (Zitnick and Dollar, 2014[+]) generates a series of bounding boxes, based on which a classifier can be trained to recognize phone from these regions and only give the center of the phone. Several parameters need to be tuned for the EdgeBoxes though.
3. Data-driving method. The phones in this dataset has a black screen and two white patches at the two ends. Using such characteristics could give candidates of regions by looking at the low and high pixel values. Then a classifier can be trained to recognize the phone in the region and only give the center of that region.
4. Alternative machine learning method. Cut the images into small pieces to create a large number of sub-images. Then train a classifier (can be CNN) to recognize a phone. Then apply a sliding window over the images to look at a sub-region each time and see if it is a phone. If it is a phone, output the center of that region and break. The problem is that multiple sliding windows with different sizes may be needed and it is slow. Also there are only small number of positive classes for training the classifier although data augmentation may help a little. (Not tested as the approaches 2 and 3 already work well).

[+] Larry Zitnick and Piotr Dollar. Edge Boxes: Locating Object Proposals from Edges. ECCV, 2014.

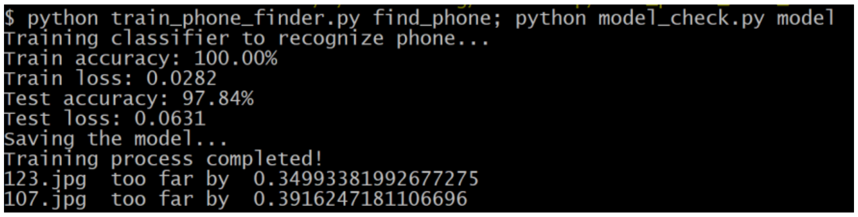
**Steps in the model development:**

1. Model tuning (model_tuning.py): tune the hyper parameters
2. Model check (model_check.py): check the model performance and understand what goes wrong for some images.
3. Repeat steps 1 and 2.
4. Finalize and save the model (train_phone_finder.py).
5. Test on a few images (find_phone.py).

Note: in the directory, there are also weight files and intermediate results saved.

**Submitted and best solution:**

The combination of solutions 2 and 3 gives best result. There are only two images (107.jpg and 123.jpg) on which the centers of the phone cannot be found in the whole dataset. The problem for both of the images are the phone laying on a dark background. Thus, the edge detection is not very successful nor the pixel value selection methods.



```
$ python train_phone_finder.py find_phone; python model_check.py model
Training classifier to recognize phone...
Train accuracy: 100.00%
Train loss: 0.0282
Test accuracy: 97.84%
Test loss: 0.0631
Saving the model...
Training process completed!
123.jpg  too far by  0.34993381992677275
107.jpg  too far by  0.3916247181106696
```

With 8 images being randomly selected as hold-out dataset, the centers of the phones are all found. (see model_tuning.py)

With such small dataset and only two images are missed, more tuning may be difficult and therefore no further tuning was done.

**Next step and future data collection:**

If only focusing on this dataset (small dataset; contains only iPhones?), collect a few more images similar to 107.jpg and 123.jpg and then tune the parameters in the EdgeBoxes to get a bounding box on the phone when the contrast is not too apparent.

If not many more images can be collected, an alternative but maybe slower method will be sliding window method. Cut each big image into small sub-images to create much larger datasets to train a classifier. To solve the problem that there are not many positive classes, i.e. sub-images containing a phone, image augmentation may help which is particularly helpful for sliding window approach. With the trained classifier, apply a few sliding windows with different sizes to the images to find the region with a phone and output the center of that region. This is worth a try in the future!

If focusing on long term, it will be nice to collect as many images as possible and to include different phones. It will be very interesting to see if a model can recognize a phone with just black screen but no white patches at the two ends. With a bigger dataset, a custom designed and trained deep neural network can be used to detect the center of the phone.