1. Brief Explanation

Based on the task description, the purpose of this task is to localize the phone in a series of 490*326 RGB image. The difficulty lies in that positions and angles of the phones are always changing. It's a typical computer vision problem associated with object detection and localization.

Therefore, I decided to find the phone location by training a Convolutional Neural Network for Regression. Convolutional Neural Networks are essential tools for image classification, however here we use CNN as a regression tool to predict 2 continuous outputs, which is the location label (x, y). We implement a 9 layers CNN to do training and validation. The network structure is shown below. Considering the limited size of this data set, we randomly choose 119 samples for training and the other 10 for validation.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 162, 244, 8)       224
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 40, 61, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 58, 16)        2064
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 19, 16)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 9, 16, 32)         8224
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 8, 32)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 4, 8, 32)          128
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
batch_normalization_2 (Batch (None, 512)               2048
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_4 (Batch (None, 128)               512
_________________________________________________________________
dense_4 (Dense)              (None, 32)                4128
_________________________________________________________________
batch_normalization_5 (Batch (None, 32)                128
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 66
=================================================================
Total params: 707,570
Trainable params: 705,650
Non-trainable params: 1,920
_________________________________________________________________
Train on 119 samples, validate on 10 samples
```

Figure 1 Network Structure and numbers of parameter

After tuning the hyperparameters, we achieve a decent performance, shown as below:

```
Training completed
Training accuracy is 0.739495798319
Validation accuracy is 0.8
```

2. Further Extensions

2.1 Possible next steps for the detector

The first thing I can think about on how to improve the performance of my detector is to ask for more data from the customer. This is because CNN, as a deep learning tool, outperforms traditional machine learning methods under big data conditions.

Secondly, the most challenging point for our phone detection data is lack of location information, because in order to achieve higher performance, most state-of-the-art object detection and localization algorithms, like HOG_SVM [1], Faster R-CNN [2] and Haar Cascades classifier [3], need bounding boxes, at least 4 numbers (x_0, y_0, width, height) as ground truth for regression. However, we only get the 2 number (x, y) here to show the center of the phone. Moreover, from customer's perspective, it's a heavy and tedious work to get the location coordinates of the phone. It would be nice if we can apply more advanced algorithm which can generate labels automatically or which doesn't even need them.

To solve this problem, I get some ideas from several papers I have read on this topic. Here shows two possible solutions:

2.1.1 YOLO algorithm

YOLO is a state-of-the-art, real-time object detection system that can detect over 9000 object categories, which outperform other state-of-the-art methods like Faster R-CNN with Resnet and SSD [4]. YOLO can make accurate predictions for object classes that don't have labelled detection data, which means it can make the data collection process much easier for the customer.

Prior detection systems, like R-CNN apply classification models to an image at various locations and scales. High scoring regions of the image are considered as the detection targets. However, YOLO use a different approach by applying a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

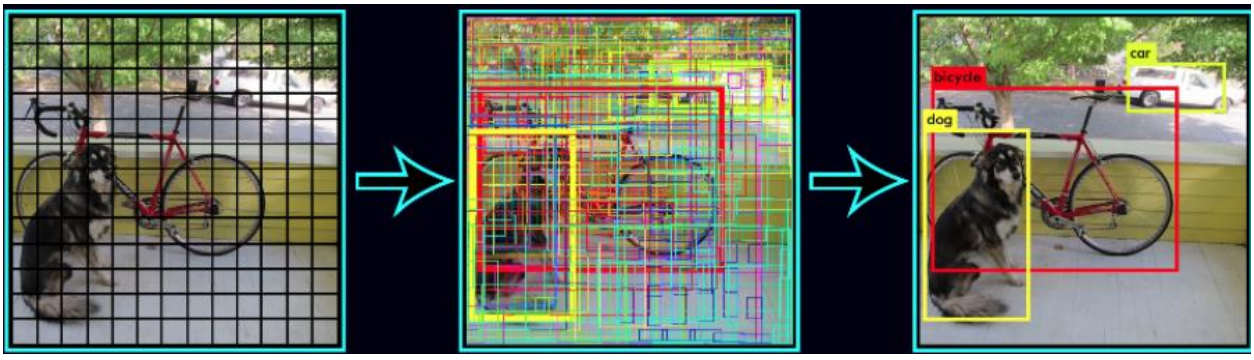

Figure 2 YOLO Object Detection

2.1.2 Fast Unsupervised Object Localization

In the technical report of Dwaraknath [5], they proposed an unsupervised object localization algorithm, which can predict the locations of objects just by taking in images without any bounding box data.

The basic idea of this algorithm is to train the image on a pre-trained CNN to identify the most important neurons via DAM heuristic, and then use neuron visualization technique “guided backpropagation” to find the corresponding regions with highest scores in the image. The overall algorithm is shown below.

Algorithm 1 Localization Algorithm

1: **procedure** FASTLOCALIZATION($k, kmax$)
2: Pass the image through the VGGNET-16 to obtain the classification
3: Identify the $kmax$ most important neurons via the DAM heuristic
4: Use ”Guided Backpropagation” to map the neuron back into the image
5: Score each region corresponding to individual neurons by passing those regions into the CNN
6: Take the union of the mapped regions corresponding to the k highest scoring neurons, smooth the image using classic image processing techniques, and find a bounding box that encompasses the union.
7: **end procedure**

Figure 3 Unsupervised Object Detection Algorithm

From Figure 4, we can see this unsupervised method provides pretty good bounding box predictions compared with the ground truth. So we can try this method to automatically generate bounding box for our phone detection data set.

Figure 4. Original Image, Algorithm output, Ground truth

References

[1] Pang, Yanwei, et al. "Efficient HOG human detection." *Signal Processing* 91.4 (2011): 773-781.

[2] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[3] Lienhart, Rainer, Alexander Kuranov, and Vadim Pisarevsky. "Empirical analysis of detection cascades of boosted classifiers for rapid object detection." Joint Pattern Recognition Symposium. Springer, Berlin, Heidelberg, 2003.

[4] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." *arXiv preprint* 1612 (2016).

[5] Dwaraknath, Anjan, Deepak Menghani, and Mihir Mongia. "Fast Unsupervised Object Localization."