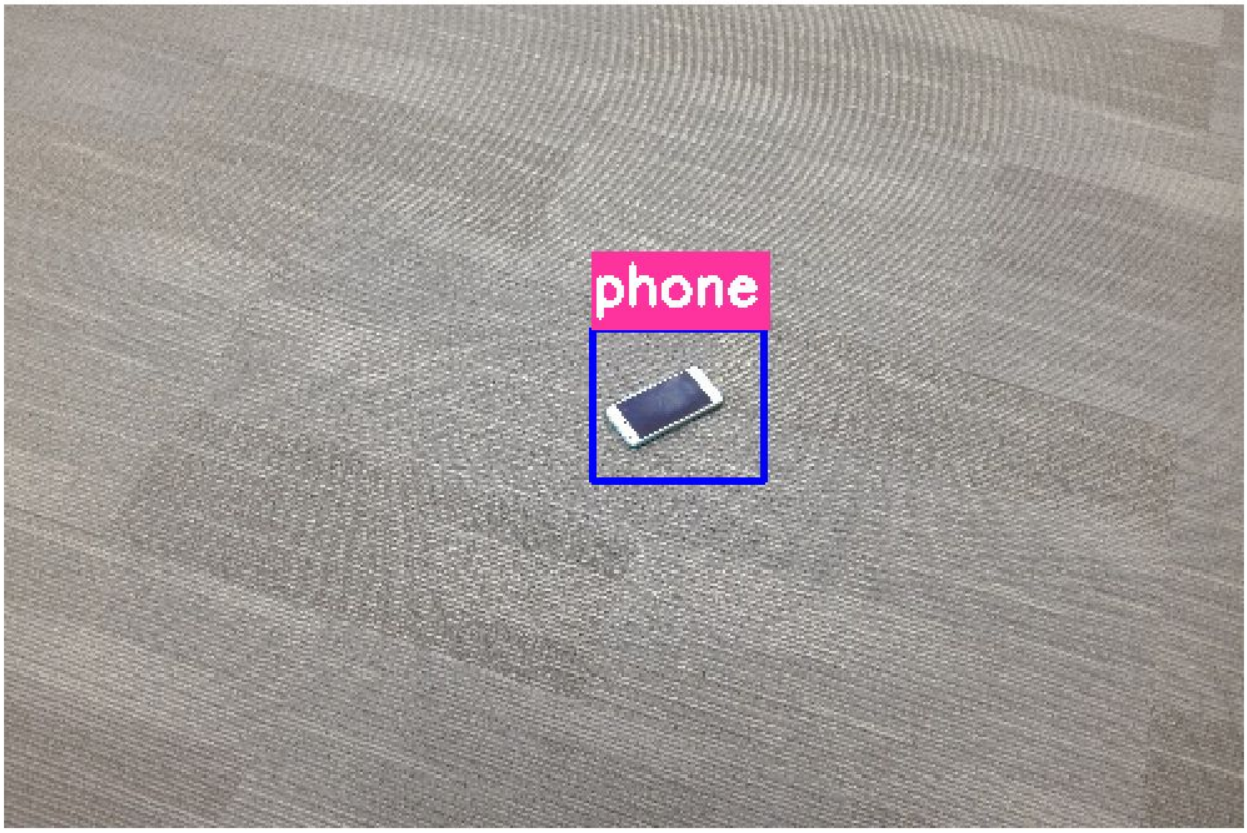

Find Phone

02.27.2018

Sheelabhadra Dey
400 Nagle Street, Apt #207
College Station, TX-77840

Python libraries used

OpenCV, Keras, Scikit-learn, Tensorflow, Numpy, Matplotlib, tqdm, h5py

Execution of the code

The *train_phone_finder.py* file accepts the training data directory, trains a model, and reports the accuracy of position prediction on the whole dataset. It generates a *model.h5* file that gets saved in the current directory which is later used by the *find_phone.py* file for locating the phone in the test image. Assuming that the data folder is in the home directory,

Terminal command:

> python train_phone_finder.py ~/find_phone

The *find_phone.py* file accepts a test image and prints the normalized coordinates of the phone in the format specified in the task description. Assuming that the test images are in the home directory,

Terminal command:

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

Approach

I treated the problem first as a object recognition task followed by object localization.

- ## Data Preparation

  I felt that The size of the training data for this task was not sufficient to train on a deep learning based model. So I used a few data augmentation techniques to generate more data. For this task since there wasn't much variability in the phone and the background images, I converted the original color images to grayscale images. This also helped eliminate unnecessary details from the images. Using the information about the location of the phone from the labels.txt file in the data folder, I cropped out images of size 44x44 corresponding to phone images and then randomly cropped out images of size 44x44 corresponding to floor images. This method provided 129 images for phone and 129 images for floor. To generate more data I randomly rotated the phone images for 50 iterations for each image. At the same time I randomly generated 50 cropped floor images from each image. This provided me a balanced dataset of 12900 cropped images that had 6450 phone images and 6450 floor images.

- ## Model implementation

Convolutional Neural Network (CNN)

CNNs are known to perform well image data. For the task of separating phone images from floor images (non-phone images) I used a 4-layer CNN. The CNN was trained on cropped images of size 44x44. 10320 images were used for training and 2580 images were used for validation (80-20 split).

- ## Finding the phone

I used a sliding window approach to locate the position of the phone in a test image. A window size of 44x44 was used with 80% overlap between adjacent windows. The window was slid across the whole image and windows where maximum matches for phone images occured were identified. These windows usually occurred in a cluster which made it easier to group them together to find the contour of the aggregated windows around the phone. To remove false positives, especially the black strips along the floor, I used a threshold value for the contour area. So, contours with area more than a threshold value were discarded and the next largest contour with an area under the threshold value was predicted to be that of the phone. The bounding box around the predicted phone was identified and the center of the bounding box was the predicted position of the phone.

- ## Accuracy

On running the prediction on all 129 images in the given dataset, the position of the phone was predicted within a normalized distance of 0.05 from the actual position for about 115 images. So, the accuracy obtained was about 89%.

Observations

I observed that most of the misclassification occurred in the presence of black strips on the floor. The model was identifying the long black strips as phones as their profile looked similar to that of the phone (black with white borders). Also, at times other objects lying on the floor were identified as phone.

This task could be done more robustly with the help of state-of-the-art object detection and localization models such as RCNN, SSD, or YOLO. These models are very deep neural networks trained on millions of images belonging to many classes (such as ImageNet

dataset). I decided to try a much simpler model as I felt that the task could be achieved using a much simpler model since the training data was low and there were only 2 classes. It also gave me an opportunity to design an end-to-end system from scratch and understand the minute details involved in object detection and localization tasks.

Next steps

It could be useful to apply more types of data augmentation techniques such as shearing, brightness augmentation, perspective transforms, etc. to the cropped images to make a richer training data for the model.

Using morphological transforms such as erosion and dilation of the contours could be helpful in removing false positives to some extent. It would separate the black strip contour and phone contour when the phone and black strips are located close to each other in the image (eg. image 123.jpg in the find_phone dataset).

Also, it would be interesting to use the provided training data to train the pre-trained weights of state-of-the-art models such as RCNN or YOLO. This should certainly lead to higher accuracy of phone detection.