ECSE 415 Project Report

McGill University, Fall 2020

Group #: *16*

Group Members:

- **Haowei Qiu (260762269)**
- **Liang Zhao (260781081)**
- **Yinghao Hu (260760428)**
- **Yujie Qin (260766685)**

Description of Person Detector (part 1):

Question: Describe the method used to detect people in part 1. State from where the method was obtained (e.g. website or github repository) and briefly describe how the method works. Provide an image showing the detections obtained on at least one image from the dataset.

The method used to detect people in part 1 is to use Detectron 2 library. Based on the suggestion provided by the Course project instruction, our team consulted the colab tutorial in the Facebook Detectron2 github repository. In brief, this method started by installing all required dependencies. Our team then obtained a pre-trained model provided by Detectron2 through Model Zoo API. During this process, we loaded a config, several weights, set the score threshold and created a DefaultPredictor. We then used the predictor to predict objects in the input image. To visualize the detection output, the method includes the use of Visualizer in Detectron2. As for Detectron2, in brief, it is a pytorch based modular object detection library originally developed from Faster RCNN.

In order to validate the Detectron2, our team followed the tutorial and selected one image from the COCO dataset. Our team retrieved the person detection result as shown in Figure 1 below. After the validation step, our team used Detectron2 for the first mall image (seq_000001.jpg) and we retrieved a detection result as shown in Figure 2 below.

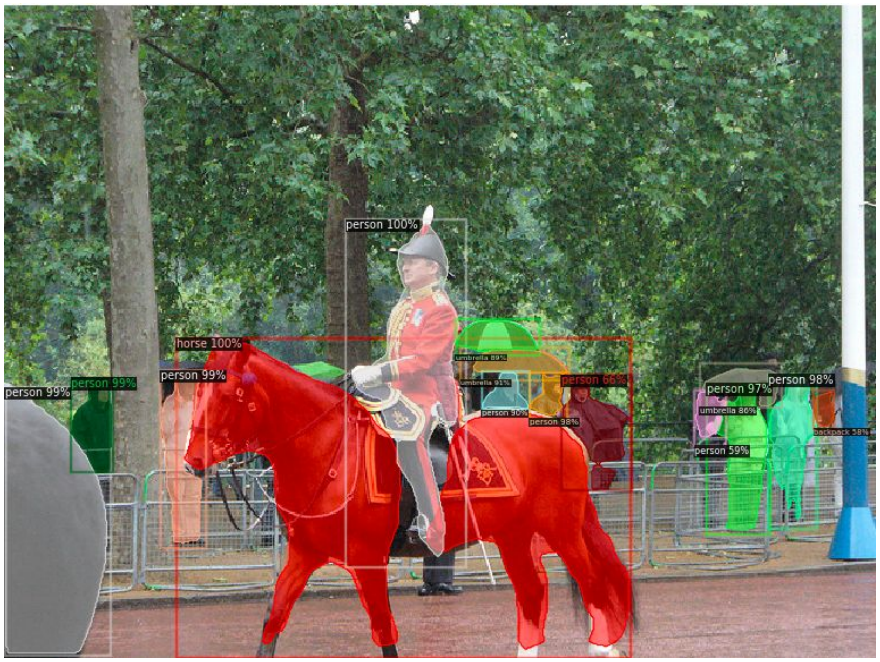

Figure 1. Using Dectron2 on image from COCO dataset

Figure 2. Using Detectron2 on image from mall dataset

Description of Person Detector (part 2):

Question: Describe the method used to detect people in part 2. Briefly describe how the method works. Describe what features you used to feed into the SVM classifier. Explain your choice of window size. Outline the process used to get the training data (positive and negative examples). Explain your choice of the number of training examples.

For the second part of the algorithm, in general, the method used to detect people is to use the Histogram of Oriented Gradients (HOG), SVM, Grid Search, and Non-Max Suppression. To be specific, our team followed the instructions on the course project instruction and first found the positive and negative bounding boxes. The positive bounding boxes are retrieved from part 1. As for window size, our team retrieved the window size by calculating the average bounding box size of the positive bounding boxes, which was 79 in length and 32 in width. After our team retrieved the average bounding box size, we then resized all positive bounding boxes to the same size (average box size). The negative bounding boxes were selected by using the sliding window technique and from the areas in the mall image where no person is presented. The size of the negative bounding box was set to be the same as the resized positive bounding boxes. There are 2997 positive bounding boxes retrieved and 3422 negative bounding boxes retrieved. Therefore, the number of training samples is 6419. The decision was made since we would like to have enough training data so that our team may retrieve good person detection results, and at the same time, control the model training time. Our team tried several amounts of training data and finally decided the number of training data. The code reference for positive bounding box is shown in Figure 3 and the code reference for negative bounding box is shown in Figure 4 below.

```
positive_boundingbox = []
for i in range(len(detected_list)):
  temp = cv2.resize(detected_list[i], (avg_width,avg_len))
  positive_boundingbox.append(temp)
```

Figure 3. Positive bounding box code

```
gray_image = cv2.cvtColor(seq_000001, cv2.COLOR_BGR2GRAY)

def sliding_window_negative(image, x_start,x_end,y_start,y_end):
  for i in range(y_start, y_end, 6):
    for j in range(x_start,x_end, 6):
      patch = image[i:i + avg_len,j:j + avg_width]
      yield (i, j), patch

corner1, negative_boundingbox1 = zip(*sliding_window_negative(gray_image,180,400 - avg_width,300, gray_image.shape[0] - avg_len))
corner2, negative_boundingbox2 = zip(*sliding_window_negative(gray_image,0,100,0, 180))
corner3, negative_boundingbox3 = zip(*sliding_window_negative(gray_image,200,460,0, 100))
corner4, negative_boundingbox4 = zip(*sliding_window_negative(gray_image,200,500,200, 300))
corner5, negative_boundingbox5 = zip(*sliding_window_negative(gray_image,400,gray_image.shape[1] - avg_width,200, 330))
```

Figure 4. Negative bounding box code

Both types of bounding boxes were stored in a list of numpy arrays. Our team then merged the two lists together as a final dataset for training. With the final dataset, our team calculated HoG features to feed into the SVM. Before training the model, we first used grid search to find out the optimized hyperparameters for SVM, and then train the model by using SVM with the optimized hyperparameters and the input training dataset. Our team finally used non-maximum suppression to filter out duplicate detection.

After the model training, as required by the Kaggle competition, we loaded the 2000 images provided, predicted the number of people in the images and finally obtained the csv file for Kaggle competition submission.

Description of the duplicate detection removal and person counting:

Question: Describe the method used to remove duplicate detections in part 2. Briefly describe how the method works. Also describe the code used to count the detections.

To remove duplicate detection in part 2, Non-Maximum Suppression was used in our algorithm. In non-maximum suppression, based on the possibility (objectiveness score) and Intersection of Union (IoU) of each duplicate bounding box, the best bounding box is selected and kept, while the other duplicate bounding boxes are suppressed. The code reference for our non-maximum suppression is shown in Figure 5 below.

```
def Non_Max_Suppression(corner_test,boundingbox_hog_test):
  nms_corner_test = []
  tensors_list = []
  tensor_score_list = []
  temp_tensor_score_list = []
  corner_test = np.array(corner_test)
  detected_persons = corner_test[labels == 1]
  penalty = np.zeros(len(detected_persons))
  detected_person_prob_list = optimized_model.predict_proba(boundingbox_hog_test)
  final_detected_persons = []
  for i in range(len(detected_person_prob_list)):
    if int(labels[i]) == 1:
      tensor_score_list.append(detected_person_prob_list[i][1])

  for i in range(len(detected_persons)):
    for j in range(i+1,len(detected_persons)):
      if abs(detected_persons[i][0] - detected_persons[j][0]) < 15 and abs(detected_persons[i][1] - detected_persons[j][1]) < 55:
        tensors_list.append([detected_persons[i][0],detected_persons[i][1],detected_persons[i][0]+avg_width,detected_persons[i][1]+avg_le
n])
        tensors_list.append([detected_persons[j][0],detected_persons[j][1],detected_persons[j][0]+avg_width,detected_persons[j][1]+avg_le
n])
        temp_tensor_score_list.append(tensor_score_list[i])
        temp_tensor_score_list.append(tensor_score_list[j])
        boxes = torch.tensor(tensors_list,dtype=torch.float32)
        scores = torch.tensor(temp_tensor_score_list,dtype=torch.float32)
        tensor_result = nms(boxes = boxes, scores = scores, iou_threshold = 0.3)
        if int(tensor_result[0]) == 0:
          penalty[j] = penalty[j] + 1
        else:
          penalty[i] = penalty[i] + 1
      tensors_list = []
      temp_tensor_score_list = []
  new_corner_test = []

  for i in range(len(penalty)):
    if int(penalty[i]) == 0:
      new_corner_test.append(detected_persons[i])
  return new_corner_test
```

Figure 5. Non-maximum Suppression to reduce duplicate detection

To count the detection, we realized this by counting the total number of "1"s occurring in the list of predicted labels, where "1" means the bounding box includes a person and "0" means the bounding box does not include a person. The code reference for detection counting is shown in Figure 6 below.

```
labels = optimized_model.predict(boundingbox_hog_test)
label_num_list.append(int(labels.sum()))
```

Figure 6. Detection counting code

Evaluation of Person Detector:

Question: Describe how you evaluated the performance of your person detector. Take the ground truth to be the bounding boxes detected in part 1. Use the IoU metric to quantify the performance of the detector. Provide an image showing the detections obtained on at least one image from the dataset.

We used IoU metric to evaluate the performance of our person detector. The code reference for our IoU is shown in Figure 7 below. After taking bounding boxes detected in part 1 as the ground truth, our team had an average IoU of 0.0827 for our SVM based person detection. Figure 8 shows the detection obtained on one image from the dataset.

```
def computeIoU(boundingbox1, boundingbox2):
  x1 = max(boundingbox1[0], boundingbox2[0])
  y1 = max(boundingbox1[1], boundingbox2[1])
  x2 = min(boundingbox1[2], boundingbox2[2])
  y2 = min(boundingbox1[3], boundingbox2[3])
  overlap = abs(max((x2 - x1, 0)) * max((y2 - y1), 0))

  if overlap == 0:
    return 0
  IoU = overlap / (abs((boundingbox1[2] - boundingbox1[0]) * (boundingbox1[3] - boundingbox1[1])) + abs((boundingbox2[2] - boundingbox2
[0]) * (boundingbox2[3] - boundingbox2[1])) - overlap)
  return IoU
```

Figure 7. Intersection of Union code

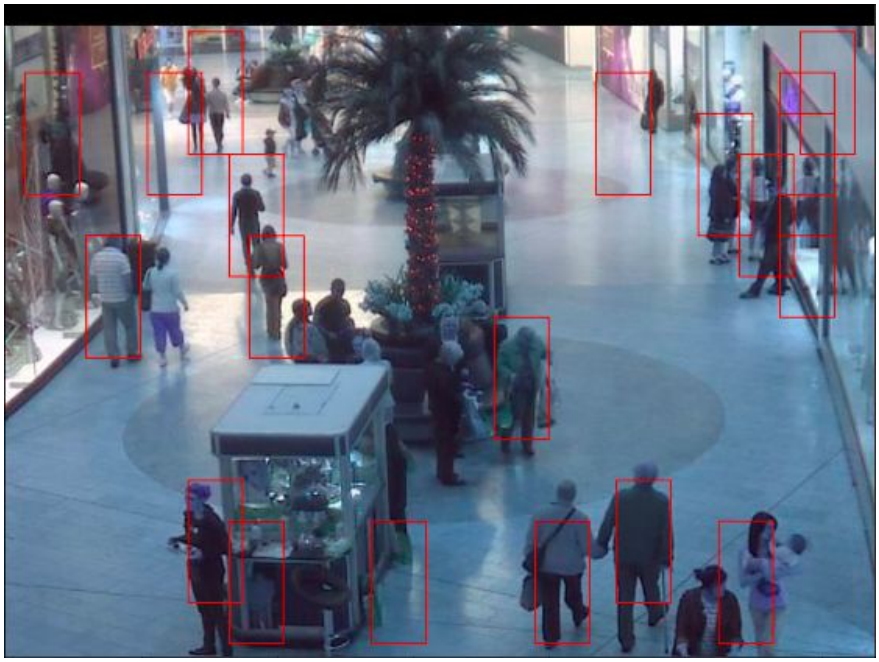

Figure 8. Person detection result

Evaluation of Person Counting:

Question: Run the person detector of part 2 on all 2000 images of the dataset. Construct the response spreadsheet and upload it to the Kaggle competition website. List the score (ranking

and metric) reported on the Kaggle web site leaderboard. (note: this will not be the final score/ranking, as that will only be reported after the end of the competition period).

As shown in Figure 9 below, in the Kaggle competition, our team finally had the score (Mean Absolute Error) of 0.27760, which was ranked 13 in a total of 23 teams.

Figure 9. Kaggle competition result

Discussion:

Question: Discuss the lessons learned during this project. How could you improve upon your results? Are there better ways to estimate how many people are in an image than the approach used in this project? Describe any difficulties you faced in the project. Discuss the suitability of different types of features (e.g. HoG vs Haar or LBP or even Deep Features provided by a CNN).

In general, there are several things that our team learned from this course project. First, we learned about various existing person detection techniques, especially the Facebook Detectron2 object detection library. Meanwhile, we learned the size of the training dataset would affect the model performance for object detection through changing the amount of bounding boxes for training in part 2. Finally, we learned that Non-Maximum Suppression is an effective method to reduce duplicate detection by comparing detection results before and after using Non-Maximum Suppression.

There are several ways to improve our result. First, we could increase the amount of positive and negative bounding boxes, which would increase the size of the training dataset. Second, we could try to improve our non-maximum suppression technique to obtain a better duplicate reduction result. Finally, we could increase the amount of hyperparameters values to be tested for grid search. To increase the amount of

hyperparameters values, we could increase the range of values to be tested or decrease the steps of increment.

There are better ways to estimate people in an image than the approach used in this project. Based on Loy, Chen, Gong, and Xiang's paper provided in MyCourses, we think that counting by clustering and counting by regression are better ways to estimate people in an image. Meanwhile, other better ways to estimate people in an image could be the use of Neural Networks, for example: Yolo, Detectron2 etc. since we found higher detection accuracy in the comparison between part 1 and part 2 detection results.

In the project, we encountered several difficulties, to start with, our team was not familiar with the existing person detection techniques for part 1 implementation. To resolve this, our team spent some time researching different person detection techniques. We compared the detection results between different techniques and finally agreed to use Facebook Detectron2 library. Another difficulty is associated with the size of the training dataset. Our team started with a smaller training dataset and retrieved bigger detection errors in prediction. Later, our team decided to increase the positive and negative bounding windows amount and retrieved a better detection result in prediction.

For the course project, our team decided to use HoG features. As for the comparison between HoG features and Haar features, Haar features return more false positives than HoG features in person detection. Meanwhile, our team attempted both HoG and Haar features and found that HoG features have a better performance. Therefore, HoG features are more suitable in human detection compared with Haar.