Egocentric RGB Hand Detection

楊程皓 B03901018
吳泓霖 B03901034
曾耕森 B03901154
鄭 煦 B03901163

1 Introduction

In this project, our work is to train a hand detector. The input of this hand detector is an RGB image containing one hand or two hands. The hand detector should output the bounding-box of each hand, and the label indicating that it is left hand or right hand.

The target images in this project contain only egocentric images. In egocentric view, the user is wearing the camera so that the camera view is the same as the user can see. In this project, the hand detector should detect the hands of the person wearing the camera. However, the hands of other people should not be detected.

We adopt YOLO structure [1, 2] to solve the problem. We also used conditional GAN to improve the quality of the dataset [3]. The details of our work are described in the next section.

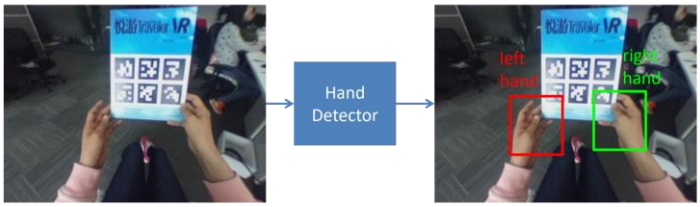

Figure 1: Illustration of the hand detection task

2 Proposed method

Our solution contains three parts. First, we use a conditional GAN structure to refine our dataset [3]. The model is trained to generate more “realistic” images based on input synthetic images. Second, we use YOLOv2 structure [2] as our bounding box detection model. The model will propose bounding box candidates with confidence scores. Finally, we use ResNet50 to classify each input image as “left hand”, “right hand”, or “both hand” image. In testing phase, this model is used to guide the bounding box selection. All three of the models are separately trained.

2.1 Domain adaptation on synthetic data

The dataset in this project contains a large collection of synthetic images and a small collection of real images. Since our ultimate goal is to detect hands in real-world egocentric images, the imbalanced dataset might impose strong bias to the hand detection model. To address this problem, we used conditional GAN to transform synthetic images into more realistic images [3].

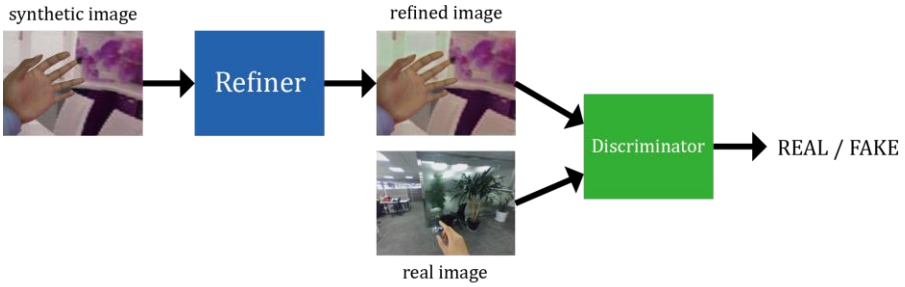

Figure 2: Structure of the conditional GAN for domain adaptation

The structure of the model is illustrated in Figure 2. We use a refiner network to generate refined image given a synthetic image input, and classify the images as real or fake by another discriminator network. During training, the refiner network and discriminator network are updated alternately. The refiner network is updated so that it can fool the discriminator to classify generated image as real image, while remaining a small L1 difference between input and refined images.

We use 5 ResNet block to refine the input image. Each block contains two convolutional layers with 7x7 filter and 64 channel. The discriminator network contains 6 convolutional layers where the final layer outputs a single value representing the probability of being a real image.

The model is trained on DeepQ-Synth-Hand dataset as synthetic images and DeepQ-VivePaper dataset as real-world images.

2.2 You-Only-Look-Once (YOLOv2)

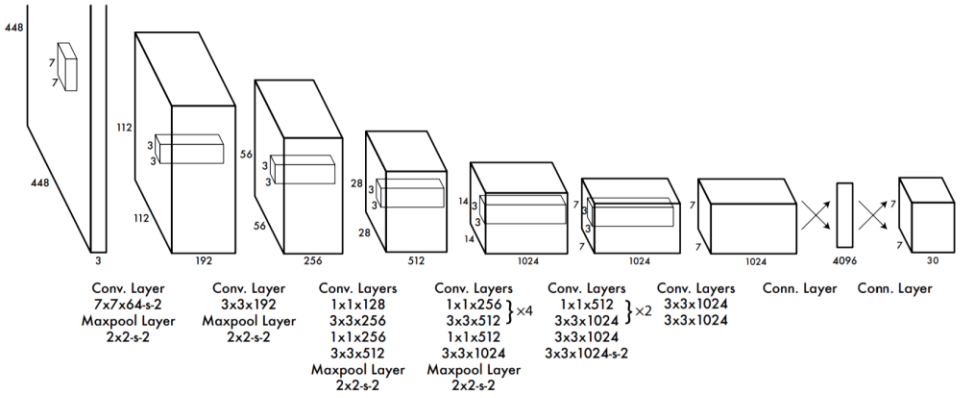

Figure 3: Structure of YOLO hand detection model

We adopt YOLOv2 [2] as our bounding box detection model. Our network has 24 convolutional layers with batch normalization at each layer's output. YOLOv2 removes the fully connected layers from YOLO [1] and use anchor boxes to predict bounding boxes. Then we removed one pooling layer from YOLO to make the output of the network's convolutional layers higher resolution. YOLOv2 predicts multiple bounding boxes per grid cell with bounding box format in (x, y, w, h, confidence). Finally, the loss function is calculated as follow:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2 + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

2.3 ResNet50 classification

In prediction phase, we use pre-trained Resnet50 [4] to detect the class label of each image. The model is pre-trained on ImageNet dataset and is available in python Keras package. Passing through the ResNet model, each image will be assigned one of the labels: “left hand”, “right hand”, “both hands”. This information will be taken by the bounding box detection model as a reference to pick the final result. When selecting bounding boxes, we simply select the one with highest confidence. For example, if the image is classified as “left hand”, then the bounding box detection model will select the box with highest confidence score among all detected boxes that are recognized as left hand.

3 Experiments

3.1 Domain adaptation

The results of our domain adaptation method are shown in Figure 4. Compare to the original images, there are slightly color changes toward green and blue and the refined images are more blurry. These changes are reasonable since we use the DeepQ-VivePaper dataset as our real-world image. The images in DeepQ-VivePaper dataset were taken in office and most of them have bluish or greenish tone. Overall, the changes in image are not very obvious and did not produce notable difference in our hand detection model performance. Possible reason may be the overstress on L1 loss in the refiner update procedure, and the size of real-world dataset.

Figure 4: Refined images generated by our conditional GAN model

3.2 Hand detection results

Both refined synthetic images and real-world images in DeepQ-VivePaper dataset are used to train our model. All images used for training hand detection model are labeled image.

We use unlabeled images in DeepQ-VivePaper dataset to test our model. The detection results are shown in Figure 5. The bounding box results in most images are not bad, but there’re some cases that the model often fail to predict accurate bounding boxes. When two hands overlap, or the hand is far away from the camera, the model often produce wrong results. In book images, when one of the hands is partly covered by the book, the model often predicts both bounding box on the other hand.

Our result get a mAP score of 0.0209194 on the OpenAI submission site.

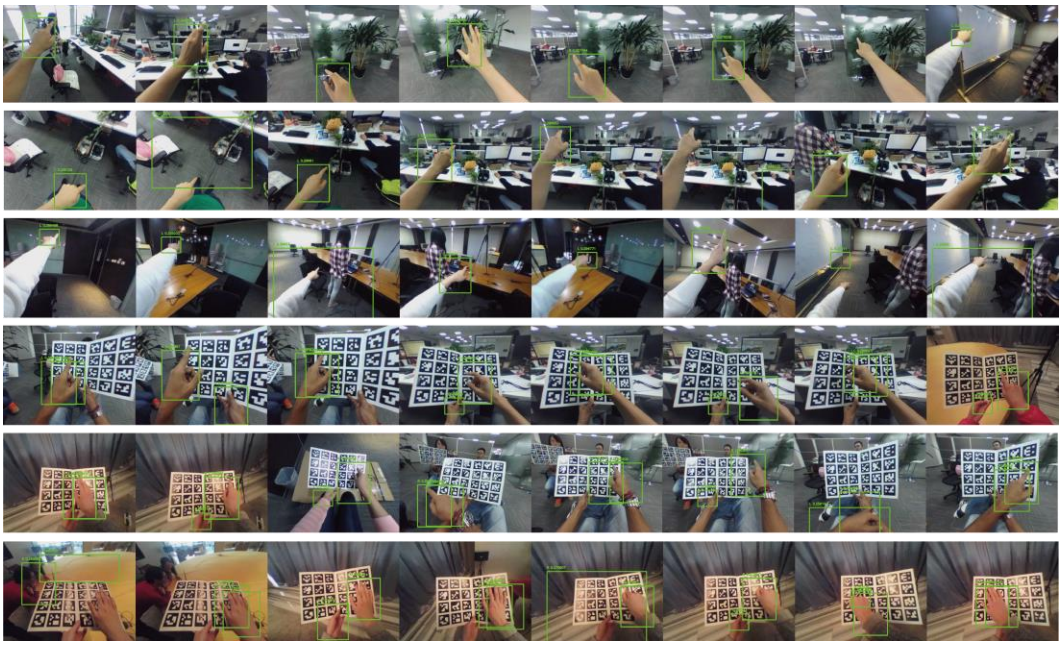

Figure 5: Our bounding box detection results on DeepQ-VivePaper dataset (unlabeled)

Since the real-world images used for training are all taken in the office, we wonder if there existed bias in our model. Hence we took some images by ourselves and tested the model. The results are shown in Figure 6. The performance on our images resembles that on the unlabeled images in DeepQ-VivePaper dataset.

Figure 6: Our bounding box detection results on real-world photos

4 Conclusion

In this project, we detect bounding boxes of hands in egocentric images. We used a three step approach to accomplish this. First, we use conditional GAN structure to refine the synthetic images in the dataset. Second, we adopt YOLOv2 object detection model as our hand detection model and output bounding box candidates with confidence score. Finally, we use a ResNet50 structure to classify each image as “left hand”, “right hand”, or “both hand” and use the information to choose the final bounding box. Our results showed that

References

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779-788).

[2] Redmon, J., & Farhadi, A. (2016). YOLO9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242.*

[3] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., & Webb, R. (2016). Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828.*

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[5] Code reference: https://github.com/experiencor/basic-yolo-keras