Video Head Detection with YOLO

Yitao Zhou(yz4171)
New York University Shanghai
yitao.zhou@nyu.edu

Abstract

Common images and videos primarily focus on people. Indeed, about 35% of pixels in movies and YouTube videos as well as about 25% of pixels in photographs belong to people (Laptev, 2013). Therefore, person detection in videos as well as photographs is a key problem for computer vision and object detection. While face detection has reached maturity, detecting people under full variation of camera viewpoints, human poses, lighting conditions and occlusions is still a difficult challenge. Especially in surveillance video data, diverse angles and poses are involved, which is valuable to analyze and worthwhile information can be extracted. In our school, all the classrooms has limits for the number of people allowed at the same time and they are equipped with video surveillance cameras. However, no detection of overrunning the people limit has been made. In this paper, I will apply the prevailing YOLO model, train it on customized data, to conduct a real-time detection of head number in videos.

1 Introduction

There are currently 350 million video surveillance cameras in the world, generating Pegabyte-level video data every day. The video data can be used in various ways, for example, road monitoring, criminal finding, and passenger flow detection. However, traditional methods of detecting useful information in the video data is to use human power, which is inefficient. With the introduction of computer-empowered object detection, low efficiency of human power checking is mitigated. Object detection has been steadily developing since its first appearance in 2001 with Viola–Jones object detection framework (Zou et al., 2019). Deep learning was enforced in 2012, with the famous AlexNet, which marked a transition from traditional detectors to CNN based two-stage detectors. But You Only Look Once(YOLO), as a milestone of CNN based one-stage detector, proposed by Redmon et al. (2016), achieves state-of-the-art mAP with extremely fast training speed.

Currently, with the number of people in our school increasing drastically, many classroom are occupied by a lot of students. The number of people often exceeds the maximum capacity of a classroom, which increases risks from happening. Meanwhile, almost all school classrooms are equipped with video surveillance cameras, but they are not fully utilized to check if the number of people is within the limit. Therefore, in this project, I use YOLO version5 designed by Ultralytics (Ultralytics, 2020) to detect the number of people in videos. As a result, an API can be built in the future so that real-time detection of the number of people in the classroom can be achieved.

To achieve the final goal of detecting heads in videos, this project is divided into two subtasks. The first is to train a customized dataset using YOLO, and then use the trained model to count how many heads exists in a picture. The second is a frontend work which separates a video into pictures, predict the number of heads in each picture, then concatenate them back into a video. The surveillance video of our school's classroom camera cannot be acquired because of privacy concerns, so the testing video is from our school website, which is *Thank you, Chancellor Yu.* Almost all frames have faces in the video.

In this paper, I will first introduce my customized dataset and data augmentation in section 2. In section 3, I will discuss model choices, project design, and details of training. The results will be presented in section 4. The conclusion remarks and future works will be summarized in section 5.

2 Dataset and Features

2.1 SCUT_HEAD

Prevailing datasets such as ImageNet usually only offers normal images, but well-labeled images of head are hard to find. Yet South China University of Technology DLVC-LAB has released an open source dataset of images and head bounding box annotations on Github in 2018. It includes 4405 images with 111251 heads, which are split into two parts. Part A(SCUT_HEAD_PART_A) of this dataset includes 2000 images from a classroom setting with 67321 head annotated, Part B(SCUT_HEAD_PART_B) of this dataset includes 2405 images crawled from the Internet with 43930 heads annotated. Both of them have a train-val-test split of 6:2:2. A problem I encountered in pre-processing the data is label format. The label format downloaded from the original dataset is Pascal VOC, but YOLO has another label format, which is a txt file containing the bounding box coordinate annotations for the corresponding image file. Therefore, I wrote a program to convert Pascal VOC label format to YOLO label format, which is included in my submission named *create_dataset.py*.

2.2 Data Annotation and Augmentation

In addition to the SCUT_HEAD data, I personally found 377 images on google image with keyword "indoor scene". I manually tagged heads in the 377 images with labelIMG (one of the most famous tools for image data labelling and annotation). It was a lot of work, but the downloaded images are quite close to the real application scene. So, I believed it would be useful during training. The detailed training evaluation and outcome will be discussed in section 3.

Then, to increase the data size and for the model to learn more general features while preventing over-fitting, data augmentation is almost indispensable. I attempted to use online data augmentation services to increase my data size. The service I used is the production of *roboflow.com*, and I augmented the data through adding Flip, Crop, Brightness and Noise, these augmentation features were randomly applied to images in the training set(which is the 377 images I gathered online), and finally producing 1131 images.

3 Model Choices, Design and Training

3.1 YOLO

In YOLO model, the image is generally split into 19*19 grids with each grid being a possible center of an object. Through training, the models learned how to spot the center of an object and how to determine the size of the bounding box. As the image is only fed to the model once (which is also the reason why the model is fast in computation), it is usually time-efficient to use YOLO for object detection. Therefore, I chose YOLO as the algorithm to train my model and modified it to fit my task. Figure 1 is the structure of YOLO v5, it mainly contains input, backbone, neck, and prediction. In the input, data was augmented by Mosaic. Auto learning bounding box anchors and adaptive image scaling was also used during the input stage. In the backbone, it mainly consists of a convolutional neural network that aggregates and forms image features at different granularities. Then, the neck is a series of layers to mix and combine image features to pass them forward to prediction. And finally the prediction consumed features from the neck and took bounding box and class prediction steps.

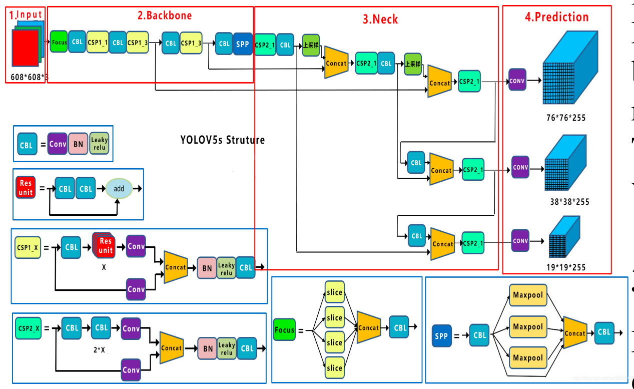

Figure1: YOLOv5 Structure

3.2 Design of Project

Data is one significant part of this project, while the design of the project is another. My project is split into several detailed works, but mainly two parts.

The first part is training. In the training part, I built a configuration file to declare the location of my training, validation and testing set. I also specified the number of class which is 1 and class name *person* in the data configuration file. In my first attempt, I used merely the training part of SCUT_HEAD as the training set. In the second attempt, I combined both SCUT_HEAD and the dataset I gathered and annotated by myself to train the model. The details of training will be discussed in Section 3.3. To evaluate the model, I used the test data from SCUT_HEAD. The testing mAP of both attempts will also be illustrated in Section 3.3.

The second part is prediction. Since I had a robust model on identifying heads in images, the next step is predicting each frames of the video. I split the video data into frames, and extracted the music of video to put it back later. The 1 min 10 second video had 24 frames per second, so in total there were 1073 frames. I modified the detection algorithm of YOLO to predict each frame, namely the bounding boxes of people's heads was labeled on each frame, and each bounding box had been combined its confidence in order to make sure the model performance. Meanwhile, during prediction, the total number of people in each frame was counted and recorded on the upper-left corner of the frame. Then I concatenated these frames back into a video along with the music.

3.3 Details of Training and Testing

Mean average precision is the most popular evaluation for object detection. And I use both *PASCAL VOC* mAP (IoU threshold of 0.5,called mAP@.5) and *COCO* mAP (IoU threshold of 10 IoU thresholds, from 0.5 to 0.95, step 0.05, called mAP@.5:.95) to measure the testing result.

For parameter initialization, I applied transfer learning with weights trained on usual COCO dataset. I used SCUT_HEAD as the training data, with image input size of 640, batch size 16 and epoch 80 during attempt 1. Therefore, since the batch size is 16, each training input contains 4*4 images. I found validation (during training) mAP converges after around 50 epochs, and the testing mAP@.5 is 0.878, mAP@.5:.95 is 0.484. During prediction, I found the result not so convincing, so I added more training data through manual labelling. In attempt 2, I used both SCUT_HEAD and manually labeled data as the training data, with aimage input size of 640, a batch size of 8, and epoch 50, achieving testing mAP@.5 0.908, mAP@.5:.95 0.526. The reasons for these changes are as follows.

Figure 2 illustrates the prediction of one image using models from attempt 1 and attempt 2. This is a bad case during the testing procedure. I found the reason is that since a batch contain 16 images, this is highly compressed, and due to YOLO's weakness in detecting small items, they are hard to be predicted. When I change the batch size to 1(so the prediction is only for one image), the result looked much better. Therefore, in attempt 2, I reduced the batch size to 8.

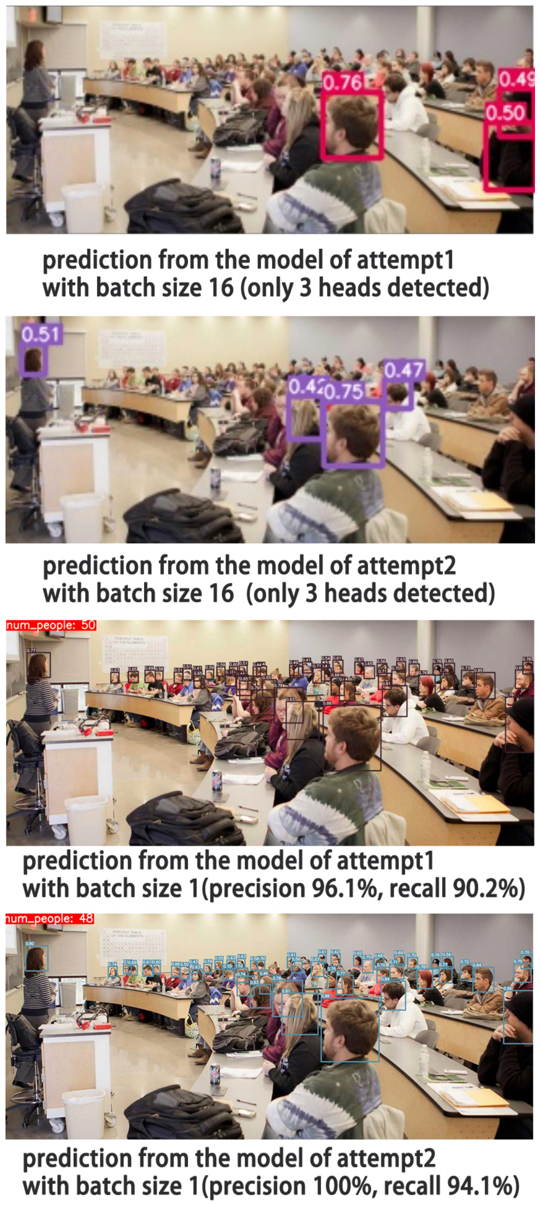
0.76
0.49
0.50
prediction from the model of attempt1
with batch size 16 (only 3 heads detected)
0.51
0.42
0.75
0.47
prediction from the model of attempt2
with batch size 16 (only 3 heads detected)
num_people: 50
prediction from the model of attempt1
with batch size 1(precision 96.1%, recall 90.2%)
num_people: 48
prediction from the model of attempt2
with batch size 1(precision 100%, recall 94.1%)

Figure 2: Prediction results for two batch sizes in two attempts

Meanwhile, from attempt 1, I found the training evaluations converge after 50 epochs. Therefore, in attempt 2, I reduced the training epochs to 50 in order to safe time and prevent overfitting. The detailed training evaluations are in Figure 3. From Figure 3, we can observe that all the training evaluations indicators for attempt 2 surpass attempt 1.

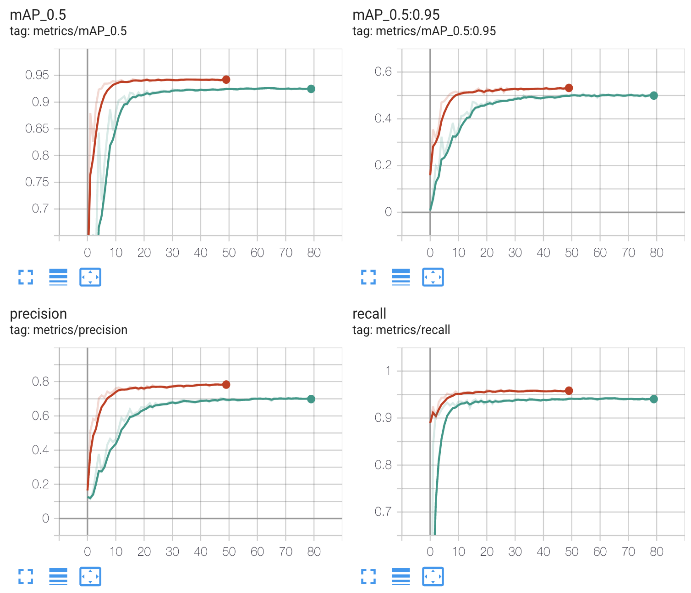

Figure 3: Training Evaluations for attempt1 (green) and attempt2 (red)

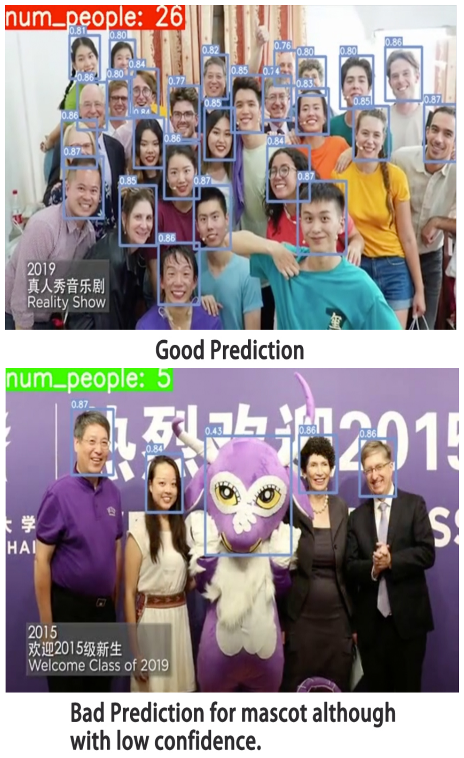

Figure 4: Sample predictions

4 Results

I used models from both attempts to make predictions on the video *Thank you, Chancellor Yu*. And I observed improvements from attempt 1 to attempt 2. In the video data, since all frames were big enough, only few heads were missed by the prediction. But some frames may contain blurred heads or humanoid mascot, these problems may lead to mistakes for the model. Meanwhile, YOLO has poor performance for small objects, although in this video prediction I rarely saw small heads missed in prediction, in Figure 3 we can see some small head being missed. A sample good prediction and bad prediction is in Figure 4. The detailed video will be played in the presentation.

Meanwhile, I added functions during the prediction to count the total number of heads in the frame. When the number of heads exceeds 20, the label on the upper-left corner will turn red, signaling that the number of heads has exceeded the maximum capacity. When the number of heads is below 20, the label will stay green. What's more, The confidence of each prediction is on the top of each bounding box, so that we can observe whether the model has high confidence for this being head or not.

5 Conclusion&Future Works

In this project, I attempted to predict the number of heads in the video, and it can be beneficial in the future for the detection of student number in our school classroom. I gained a a lot during the project: I learned how YOLO works, how to modify the model to fit my task, how to find useful data, how to debug during training, how to make predictions, and how to improve the performance. With finding out this application scenario, I also learned AI and machine learning could help benefit our society.

Yet many things can be done to enhance the performance. First, since the prediction is based on video data, I can add trackers to trace the action of each person in the video, through which I can obtain more information. Meanwhile, adding data could enhance the model robustness, I observed that detection under dim environments and of small heads is extremely hard for my model, so I could find more relevant data. Moreover, YOLO has limitations for predicting small objects, therefore, using another model SSD or combine the predictions of YOLO and SSD together might be a good attempt to increase robustness.

References

Laptev, I. (2013). *Modeling and visual recognition of human actions and interactions* (Unpublished doctoral dissertation).

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).

Ultralytics. (2020). *ultralytics/yolov5.* Retrieved from `https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data`

Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055.*