INFO 550 – Final Project options 1

Due: Friday, May 6, 2022, 6pm

10 points of final grade

Quan Gan

Graduate

1 Introduction

In this final project option 1, I am going to explore the object detection algorithm YOLOv5 [1]. YOLOv5 is based on the YOLOv3 [4] algorithm but YOLOv5 is implemented by using Pytorch [2] framework. YOLOv4 was published in the same year with YOLOv5 but in different months and deploys different methods to improve the accuracy and speed. Object detection is a Machine learning/Artificial Intelligence task in the computer vision field. Unlike the image classification task to categorize one image into a certain class or multiple classes. The object detection is the advanced technique and aims to detect the instances of a certain class within the digital images or videos. The core concept is to use deep learning and neural network to deal with the image pixels and analyze the patterns of each categories. The state-of-the-art object detection algorithms have two main types. one stage methods such as YOLO, SSD and RetinaNet, focus on the speed to detect objects. Two stage methods such as Faster R-CNN, Mask R-CNN concentrate on the accuracy. Limited by the computer performance, this project explored the YOLOv5 algorithm to detect object based on custom data.

2 YOLO Algorithm

YOLO [3] is the abbreviation of "You only look once". Just like its name, this algorithm uses a single forward propagation through the neural network to detect objects. Convolutional Neural Network (CNN) is a type of neural network model that has proven to be successful at image classification in computer vision. CNN is the core of YOLO algorithm. Beside the neutral network, another basic concept in the object detection is bounding box. Bounding box is an enclosing rectangle box to represent an instance in the images or the videos. The coordinates will help neural network to train the models.

2.1 Core Idea

Joseph et al. [3] mentioned in their paper. YOLO algorithm frames object detection task as a regression problem to separated bounding boxes and associated class probabilities rather than a classifier problem. Unlike the two stage methods to predict the potential bounding boxes first and then predict categories, YOLO predict the bounding boxes and classes simultaneously in the convolutional network, which means YOLO would be really fast.

Figure 1 is how YOLO model works.

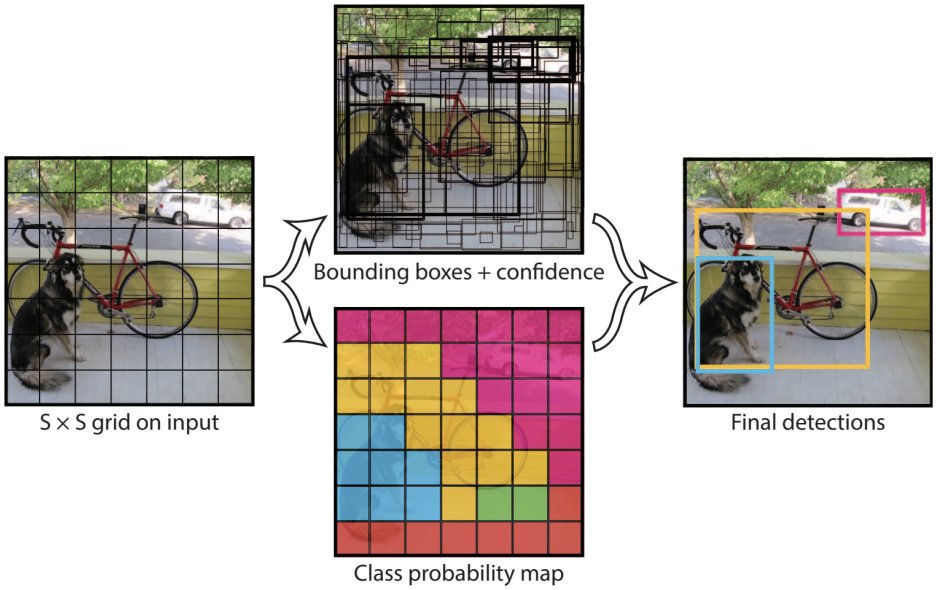


Figure 1: YOLO model [3]. The full image is divided into S x S grid. Each cell will predict the B bounding box and confidence score. At the same time, each cell also calculate the C class probabilities.

Step 1: From the left part of Figure 1, the initial full image will be divided to S x S grid.

Step 2.1: From the middle top part of Figure 1, each grid cell will predict B bounding box and confidence score.

$$\text{confidence} = Pr(\text{Object}) * \text{IOU}$$

$Pr(\text{Object})$ is the probability that the cell contains the object. IOU means Intersection over Union which is a evaluation metric between the labeled bounding box $B1$ and the predicted bounding box $B2$ by the area. So the center coordinates, weight, and height of bounding box are important parameters in the YOLO.

$$\text{IOU} = \frac{\text{B1} \cap \text{B2}}{\text{B1} \cup \text{B2}}$$

Step 2.2: From the middle bottom part of Figure 1, each grid cell will predict C conditional class probabilities, $Pr(\text{Class}_i|\text{Object})$. Each grid cell only selects the highest probability class.

By combining confidence score and class probabilities, the model could have class-specific confidence score for each bounding box and know how well the predicted bounding box fit the object.

$$Pr(\text{Class}_i|\text{Object}) * Pr(\text{Object}) * \text{IOU} = Pr(\text{Class}_i) * \text{IOU}$$

The entire task is processed in the convolutional network which yield the final trained model.

2.2 Architecture of YOLOv5

Data augmentation: is a image processing techniques to generate new synthetic train images out of provided train images but uses different ways such as Mosaic, Copy paste, MixUp, Random horizontal flip, and more to manipulate the images. The goal of data augmentation is to enhance the heterogeneity of the input images so that the built object detection model can handle images from a variety of contexts.

(a) (b) (c)

Figure 2: (a, b, c) are the generated data augmentation train images in this project.

New Network Design: The convolutional network design is based on YOLOv3 with some fine-tunings. From the Glenn Jocher [1] summary, the backbone uses New CSP-Darknet53. the neck uses SPPF and New CSP-PAN, and the head uses YOLOv3 Head. In the neural network, Backbone and Neck are both feature extracting network but Neck extracts more deep and elaborate features. Head is the network to compute and yield the final output.

2.3 YOLOv5 pretrained models

YOLOv5 [1] has 10 pretrained models. Fig 5 is from the official yolov5 repository [1] and list the 5 basic pretrained yolov5 models' performances. In this project, we will train our custom data based on the yolov5s model with CPU core.

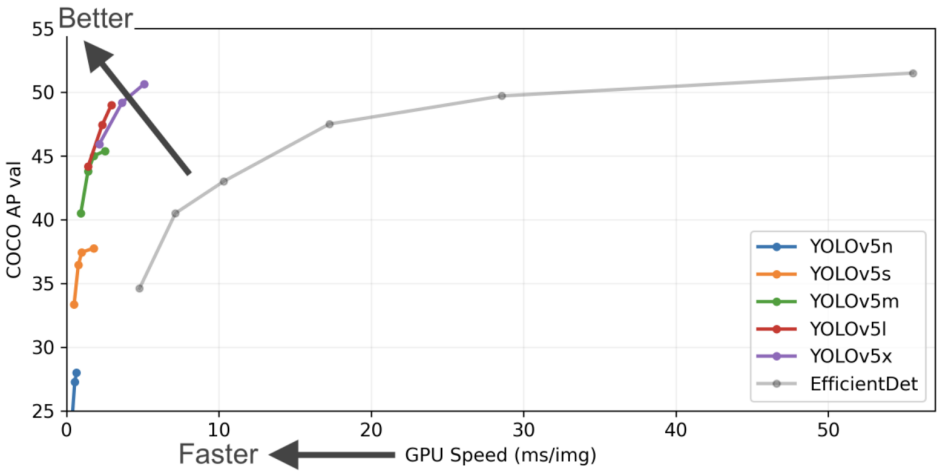
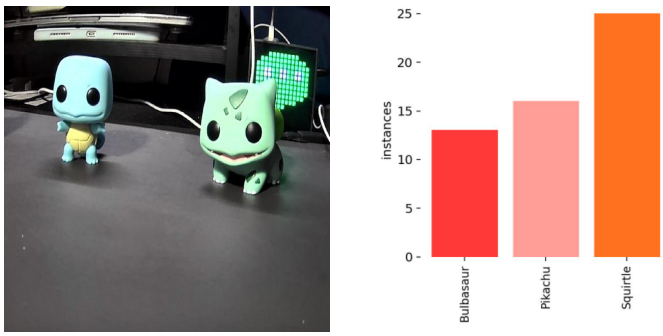


Figure 3: 5 basic pretrained yolov5 models' performances.

3 Data

The custom data was captured by my camera via Python openCV library. The data set contains 38 images. Each image contains one or more pokemon funko pop toys. There are only three pokemon labels, Pikachu, Squirtle, and Bulbasaur. Then I used Roboflow, a web-based tool, to annotate my objects of interest and convert them to yolov5 format data set. Roboflow tool generated same number text files that record the instance labels and the instance bounding box information and transformed original images to 640 x 640 pixels images. 26 images were randomly chosen as the train set and 6 images were chosen as the validation set. In addition, 6 images were chosen as the test set to test the trained model. Figure 4 is a train image example and the instance number in the train image set.



(a) (b)

Figure 4: (a) A yolov5 format train image example. The label for this image is "2 0.24140625 0.3203125 0.18984375 0.3390625" and "0 0.73046875 0.4 0.26015625 0.36328125". The first integer represents the category number and rest four floats are x center, y center, box width, and box height. (b) the instances distribution.

4 Instruction

4.1 Operation System

The model is trained by my personal computers. I tried both operation systems to train the model with Pytorch CPU core.

Windows: The system specifications are as follows: Processor - **Intel^{R}Core^{TM}** i7-7700 CPU @ 3.60GHz. RAM - 16.0 GB. System type - 64-bit. Graphics - GeForce GTX 1080. **Mac:** MacOS: Monterey version 12.3.1. Chip: Apple M1. Memory: 8GB.

4.2 Environment Requirements

Python requirements >= 3.7.

```
pip -m venv ODenv
.\ODenv\Scripts\activate # For windows
source ODenv/bin/activate # For mac
```

There is a **README.md** file in the provided folder about how to setup the project. To simplify the process, I exported the libraries to the **requirements_win.txt** and the **requirements_mac.txt** in the root folder. Another **requirements.txt** file in the **yolov5** folder is also fine but lacks the Jupyter notebook library. I provided both Jupyter notebook version and Python script version to visualize the result. If you want to run Jupyter notebook version, Please also install *ipykernel* in the same virtual environment.

```
# In the root folder
pip install -r requirements_win.txt # for windows
pip install -r requirements_mac.txt # for mac
```

4.3 Train Model

Thank Glenn Jocher yolov5 repository[1]. We don't need to write complex setup code for the convolutional network. We could train our custom model with provieded **train.py** in the yolov5 repository with different setup parameters.

```
# go to yolov5 folder
cd yolov5

# train custom model
python train.py --img 640 --batch 16 --epochs 300 --data data_pokemon.yaml
--weights yolov5s.pt
```

In order to achieve best accuracy, I choose epochs as 300, which will take more than **1.5 hours**. The result showed 200 epochs also work for this data. To reproduce the train process and reduce the running time, please try fewer epochs. The result of train model can

be found in "yolov5/runs/train/exp" folder. If you train a new model, it will be "exp1". The **data_pokemon.yaml** is a file to describe the the data information.

If you receive the error information (AttributeError: 'NoneType' object has no attribute '_free_weak_ref') after the command finishes, don't worry. It's the model bug because of the update of new packages but doesn't impact the trained model.

4.4 Visualize Model

In order to visualize the result and run object detection, I wrote a simple Python script and also a Jupyter notebook file. This instruction only provides the Python commands.

```
# go to Python_code folder
cd Python_code

# to see available commands
python main.py -h

# predict the test images and the result images will be saved in the result folder
# If you will train the model again, the model path should be "../yolov5/runs/train/exp1"
python main.py -i -c ../yolov5/runs/train/exp -p ../pokemon_data/test/images

# To start a real time detection with camera
python main.py -c ../yolov5/runs/train/exp
```

5 Evaluation

5.1 Model Train

During the model train, the yolov5 would also provide result for each epoch. All available result can be found in the yolov5/runs/train/exp folder. Figure 5 displays the performance during the model train.

1. box_loss is the bounding box regression loss (MSE).
2. obj_loss is the object loss (Binary Cross Entropy).
3. cls_loss is the classification loss to measures the correctness of the classification of each predicted bounding box (Cross Entropy).
4. mAP means mean Average Precision.The followed number is the threshold for IOU.

From the figure 5, the train/validation loss results have sharp decreasing during the first 100 epochs. The precision, recall, and mAP_0.5 reach 1.0 near 150th epoch but the mAP_0.5:0.95 is over 0.9 after 200 epochs. The further epochs have no obvious improvement on the accuracy.

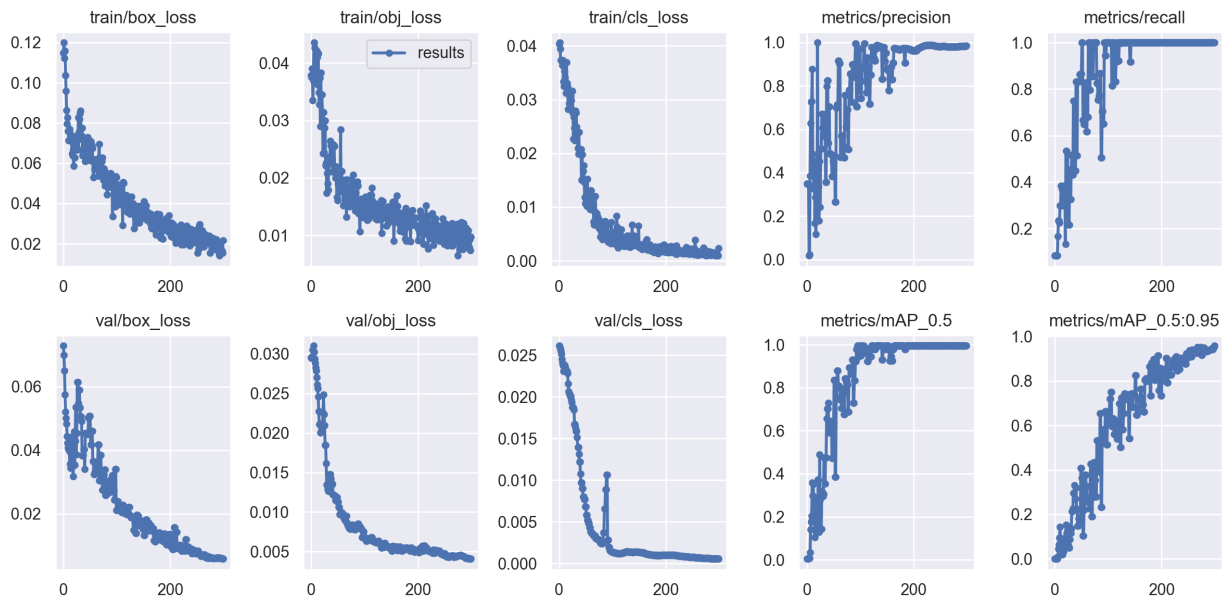
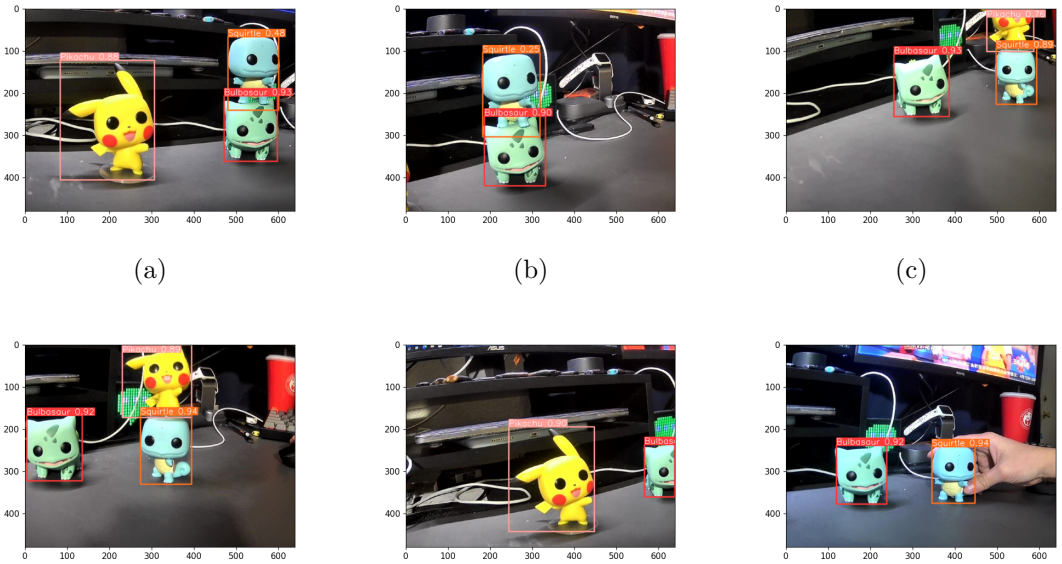


Figure 5: The model performance on 300 epochs

5.2 Image Test Set



(a) (b) (c)

(d) (e) (f)

Figure 6: The results for the test images.

Figure 6 is the predicted result images from the test image set. The trained model can correctly detect all pokemon toys with different confidence scores. As we can see, the figure

(a - d) contain stacked toys, which are not in the train set or validation set. The trained model can still detect them well. The model can also detect partial object in the figure (e) and object with a noise by using hand to hold in the figure (f).

5.3 Real time detection

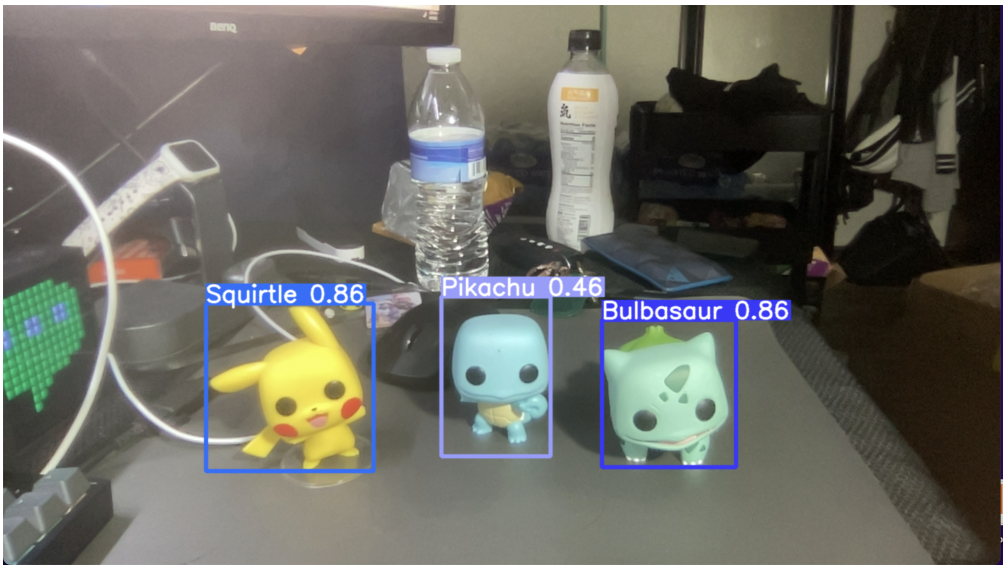


Figure 7: The real time detection.

Figure 7 is one frame of real time detection by my camera. The result shows the model misdetect the Squirtle and Pikachu. In deed, the model preforms bad to detect Pikachu. Except Bulasaur toy, the mode always classifies toys as Squirtle. The possible reason may be caused by the camera qualification and the light.

6 Other Consideration

Firstly, The train data is small in this project and unbalanced. The official documentation recommends to use 1500 images per class. Secondly, The weight model in this project is the small pretrained model yolov5s. We may use better pretrained model to weight model.

7 Additional Instruction

```
# go to Python_code folder
cd Python_code

# Run yolov5 pretrained model directly in real-time detection and have fun!
python main.py
```

References

[1] G. Jocher. Yolov5. `https://github.com/ultralytics/yolov5`, 2020.

[2] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[4] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.