Weapon Detection Of Secure Camera

Bingjiun Miu 015235233

December 11, 2022

Project option: Application Track

1 Introduction

In this project assignment, the final goal is to build the entire machine learning pipeline of the weapon detection application from data collection, data preprocessing, model training, model evaluation, model deployment, and user interface application(fig 1). I created the custom dataset of weapons in this project, and the machine learning algorithm I utilized is yolov7. The environment I used to train is an anaconda virtual environment in order to specify the packages' versions. The GPU device I utilized in the training process is NVIDIA GeForce RTX 3050.

Code repo: https://github.com/joseph123223/WeaponDetection

Demo video: https://youtu.be/3IsdND7GT6M

2 Environment Settings

Before we started to get into the application, there were some settings that I needed to introduce. I will separate these settings independently in this section for a more straightforward explanation.

2.1 Yolov7 Algorithm

In order to accomplish the training of the machine learning model, the algorithm is one of the essential components. Yolo algorithm is one of the most popular and robust algorithms frequently used in object detection, pose estimation, and image classification. Yolov7 is the latest version of the Yolo algorithm, which came out in July 2022. To utilize this excellent machine learning model, we need to download it

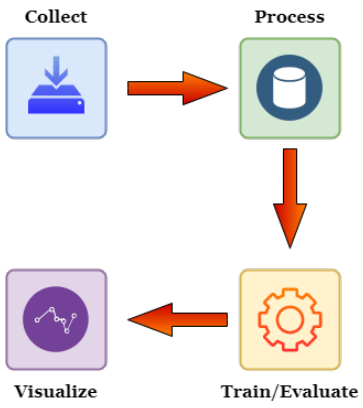

Figure 1: Application pipeline architecture

```
39 lines (34 sloc)  |  950 Bytes
```

```
# Usage: pip install -r requirements.txt

# Base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.1
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
torch>=1.7.0,!=1.12.0
torchvision>=0.8.1,!=0.13.0
tqdm>=4.41.0
protobuf<4.21.3

```

Figure 2: requirements.txt of yolov7

into our training environment. The fastest way to do that is by using the git clone command to clone the source code from yolov7 official documentation[1].

2.2 GPU setting

In this project, I used my GPU to accelerate the training process. However, if we want to utilize our GPU in training, there are some settings we need to set up. In the yolov7 official documentation (fig 2), they mention the requirements of the packages that utilize in training. Therefore, if we want to use GPU to train the model, we need to make sure that the version of torch and torchvision fit the requirements.

3 Data collection

Once we have completed all the settings of our training environment, the first step is to create our custom dataset. In this project, the custom dataset I created is the dataset of weapons. Data collection is one of the most challenging parts of this project because finding and downloading free-use images suitable for training is extremely difficult. Even though I found some images, they can only be used for training after processing. Therefore, the methodology I used to download the images I need is the simple-image-download[2] package that python's library provided. This package can download images from google efficiently. However, many overlapping pictures and images are not suitable for training, so it is necessary to clean all of these useless data before training our model.

4 Data Processing

After collecting all the needed image data, the next step is to create the label for all the images. The label-creating tool I used in this project is LabelImg[3]. LabelImg is one of the popular image annotation tools we could use to create labels for our image data. In this weapon detection project, ensure that the algorithm choice is yolo since I was using yolov7 to train my model(fig 3).

5 Settings For Custom Dataset

After cloning the source code of the yolov7 algorithm and preparing all the image data I needed, we still require some modifications to enable this algorithm for my custom dataset. The official yolov7

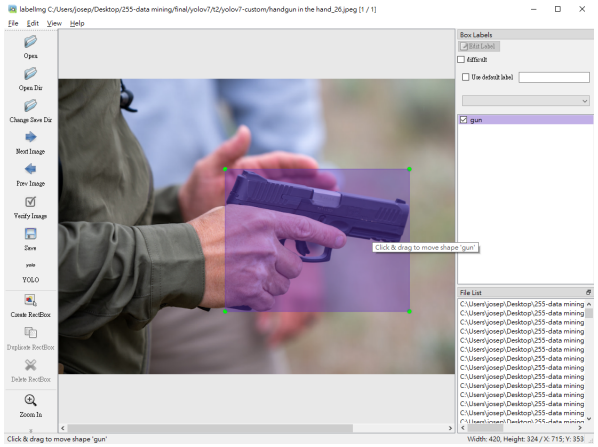

Figure 3: labelImg

```
data > coco.yaml
1  # COCO 2017 dataset http://cocodataset.org
2
3  # download command/URL (optional)
4  download: bash ./scripts/get_coco.sh
5
6  # train and val data as 1) directory: path/images/, 2) file: path/images.txt, or 3) list: [path1/images/, path2/images/]
7  train: ./coco/train2017.txt  # 118287 images
8  val: ./coco/val2017.txt  # 5000 images
9  test: ./coco/test-dev2017.txt  # 20288 of 40670 images, submit to https://competitions.codalab.org/competitions/20794
10
11 # number of classes
12 nc: 80
13
14 # class names
15 names: [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
16         'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
17         'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
18         'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
19         'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
20         'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',
21         'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
22         'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',
23         'hair drier', 'toothbrush' ]
24
```

Figure 4: Coco.yaml

documentation includes the default dataset, which is the coco dataset. However, I will not utilize this coco dataset because I will create one. Therefore, we need to modify all the settings about the targeting dataset to my custom dataset.

The first file I need to change is the yaml files used to train the model. The default yaml file is used to train the coco dataset(fig 4). Therefore, we could follow this template to create our yaml file to utilize my custom dataset. Next, we must modify the dataset's directory and ensure the directory is the correct path with our custom dataset. Another modification is the number of classes and the name of the classes(fig 5). Since the gun is the only item that my model will detect, the class number will become 1, and the class name will be "gun." We also need to change the number of classes in the training config file to make the number of classes identical. One of my application's future scopes is to add more weapon classes to my models. If we desire to achieve this feature, this is the location to add additional classes.

The next thing we need to complete before training is downloading the yolov7 models we like to use. We could choose many models provided by yolov7's official documentation, such as yolov7-X. In this project, I used the fundamental one, which is yolov7, to accomplish the training.

6 Train the model

After we have accomplished all the settings, we can finally begin to train the model for the custom dataset. Based on the command of training provided by yolov7's documentation, modify all the default file names into our custom training file names. The total number of images in my custom dataset is

```
5   # number of classes
6   nc: 1
7
8   # class names
9   names: [ 'gun' ]
```

Figure 5: Custom.yaml

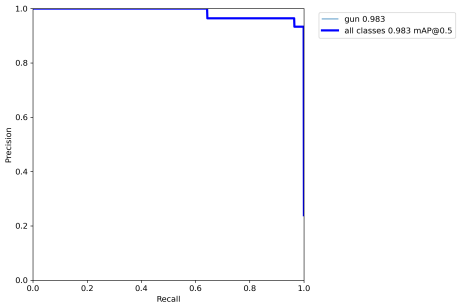

Figure 6: PR curve

236, 216 for training, and 20 for validation. I tried to modify the dataset's size and tested many different epochs during the entire training process. The range of epochs I attempted was from 150 to 800. Time was from 26 minutes to 6 hours and 20 minutes.

7 Performance

The best performance of all my trained model is the epochs of 800. With this number of epochs, the training result has the highest performance and the lowest misreport rate. Furthermore, the precision and recall curve(fig 6) and F1 curve(fig 7) indicate that this model's performance is excellent.

The yolov7 algorithm also will generate the result graphs with different epochs during the training process(fig 8). We can use these graphs to decide the model in which epochs have the most outstanding performance.

8 Model deployment and UI

After I obtain the well-performance model, the next step is to deploy this model and construct the application to allow users to utilize my model in the user interface. One of the most prevalent methods is using Flask to build the application. First, I created a simple user interface to allow users to upload

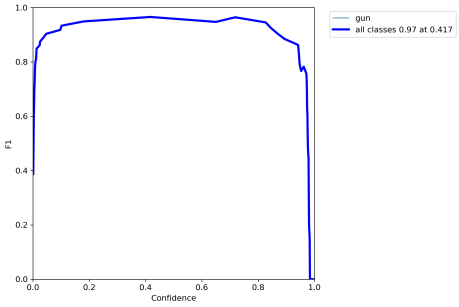

Figure 7: F1 curve

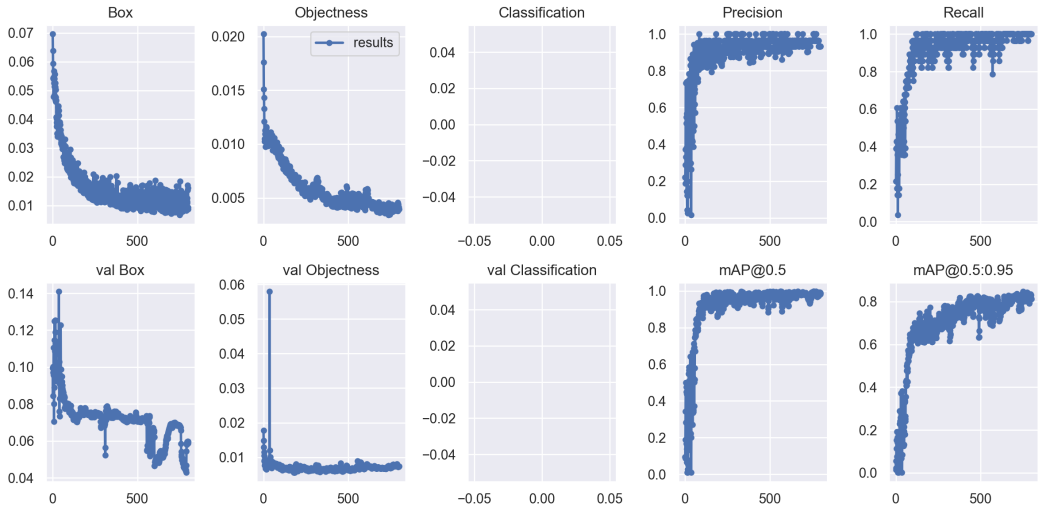

Figure 8: Result graphs

Figure 9: User interface

their input images or videos(fig 9). I modified the method of utilizing the model based on the default detect.py file provided by official yolov7 documentation. Once users submit the image or video file they selected, the backend part of my application will store the input file in the upload directory. After that, the backend code will begin the prediction using this input file. I also constructed the destination directory to store all the prediction outputs in the same file and utilized a download option for users to download the prediction result. All the code details will be included in my Github repository and visualized through my demo video.

9 Conclusion

In conclusion, this project is a super valuable and helpful experience for me because this is my first time accomplishing the entire pipeline of a machine learning application. Furthermore, the entire learning process gives me more understanding of how the machine learning model be utilized in the industry. For instance, how to train the model efficiently and correctly, and how to deploy the model into the production environment. I also learned how to utilize yolov7, the super powerful algorithm in object detection, in machine learning. Overall, these experiences would be helpful for my future development in the machine learning field.

10 References

[1] Wong, K. Y. (n.d.). Wongkinyiu/Yolov7: Implementation of paper - yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. GitHub. Retrieved December 11, 2022, from https://github.com/WongKinYiu/yolov7

[2] Simple-image-download. PyPI. (n.d.). Retrieved December 11, 2022, from https://pypi.org/project/simple-image-download/

[3] Heartexlabs. (n.d.). Heartexlabs/labelimg: LabelImg is now part of the label Studio Community. the popular image annotation tool created by Tzutalin is no longer actively being developed, but you can check out label studio, the open source data labeling tool for images, text, hypertext, audio, video and time-series data. GitHub. Retrieved December 11, 2022, from
https://github.com/heartexlabs/labelImg

[4] (Useful tutorial) Grinberg, M. (n.d.). Handling file uploads with flask. miguelgrinberg.com. Retrieved December 11, 2022, from
https://blog.miguelgrinberg.com/post/handling-file-uploads-with-flask

[5] (Useful tutorial) Cairocoders. (2021, April 9). Python flask upload and display image. YouTube. Retrieved December 11, 2022, from
https://www.youtube.com/watch?v=I9BBGulrOmoamp;t=317samp;ab$_c hannel = Cairocoders$