Chapter 4 implementation

4.1 project design

4.1.1 Resource Requirements

* Hardware:  
  A. 64-bit, x86 desktops or laptops with dedicated Nvidia graphics card. We used a 3070 RTX Nvidia graphics card.  
    
  B. Alternatively, use PyCharm.

C. A toy gun, knife, and pistol

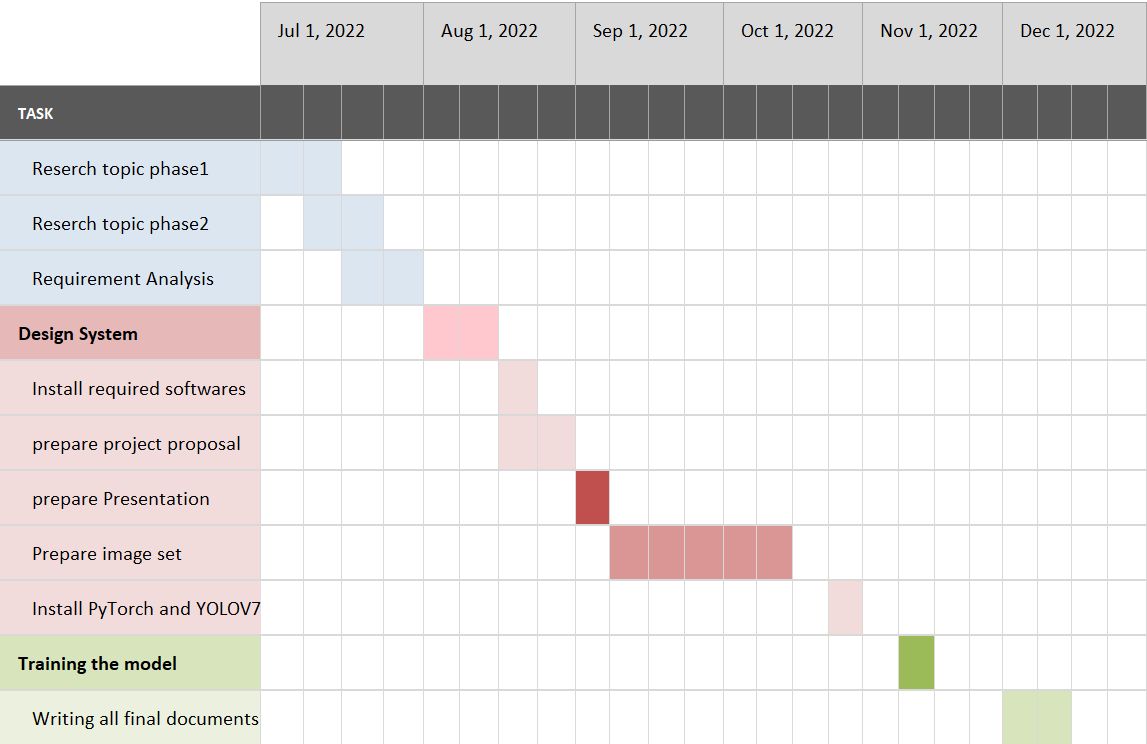
* Software:

1. PyTorch with GPU support (CUDA with PyTorch)
2. MakeSense website for labeling images with bounding boxes
3. Python 3.10

* Resources:

1. Images dataset
2. 5000 images for knife
3. 5000 images for gun
4. 5000 images for pistol

* Schedule



4.1.2 Image collection

In the image collection step of system design, image collection is required to get a sufficient number of images for training and testing purposes.

The number of images for both training and testing are shown in Table 3-1.

|  |  |  |  |
| --- | --- | --- | --- |
| Number of images | Knife | Gun | Pistol |
| Training | 3,950 | 3,950 | 3,950 |
| Validation | 459 | 459 | 459 |
| Test | 494 | 494 | 494 |

Table 3-1 Number of images\_for each class

4.1.3 Image labelling

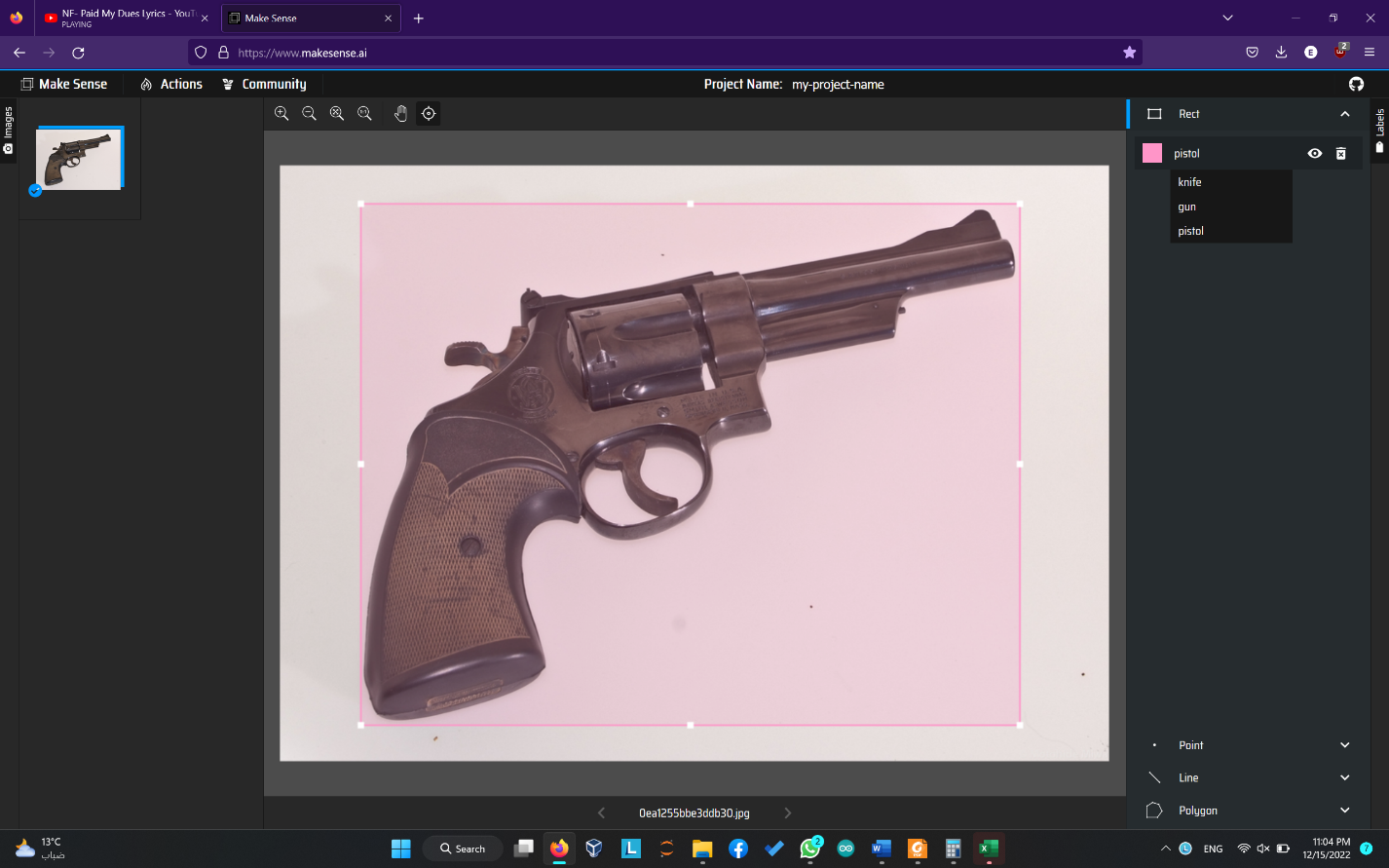
In the next step image labeling, a website and software called “Make sense” it’s Open source and free to use under GPLv3 license. it is used to draw a bounding box and give a label on each image, then save the details of that into a different type of format like JSON, XML, CSV and what we use TXT yolo format.

Figure 3-2 Make Sense demonstration

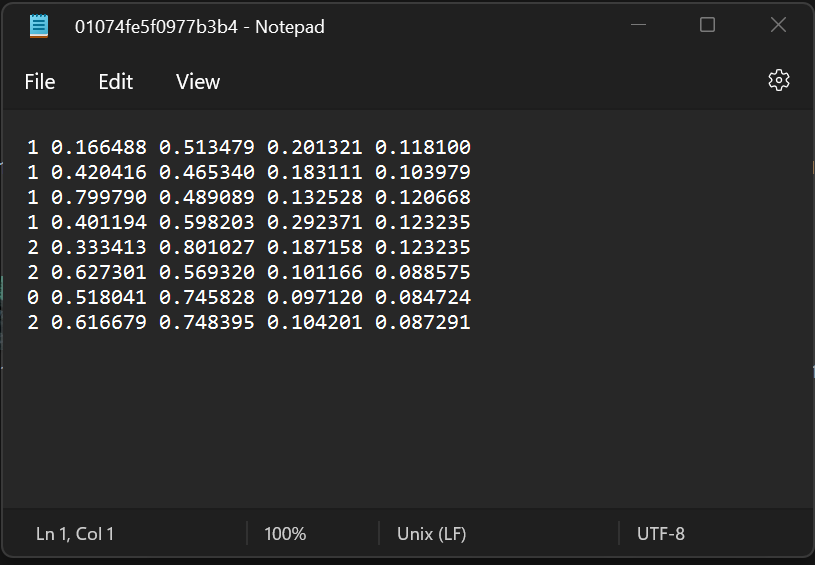


Figure 3-3 Example of TXT yolo format

4.1.4 Image separation

There is no regular rule on how much you should divide the image dataset into training and testing, normally the common ratios are 80/20, 90/10, or 70/30 for training and testing datasets. By reading resources online, a common rule is applied that the bigger the dataset is, the smaller the testing dataset should be. the partition ratio chosen for the dataset is 75/12/13 (training /validation/test )sets and this could be done by using a python script.

4.2 project Implementation

4.1.2 Model training

Once the dataset is well prepared, the dataset is then fed into the model to start training,

PyTorch, YOLO object detection and PyCharm are tools chosen to implement in this research. To start training YOLO algorithm is installed into PyCharm the first time, and then install the required library by the following code:

1. Install YOLO V7 :

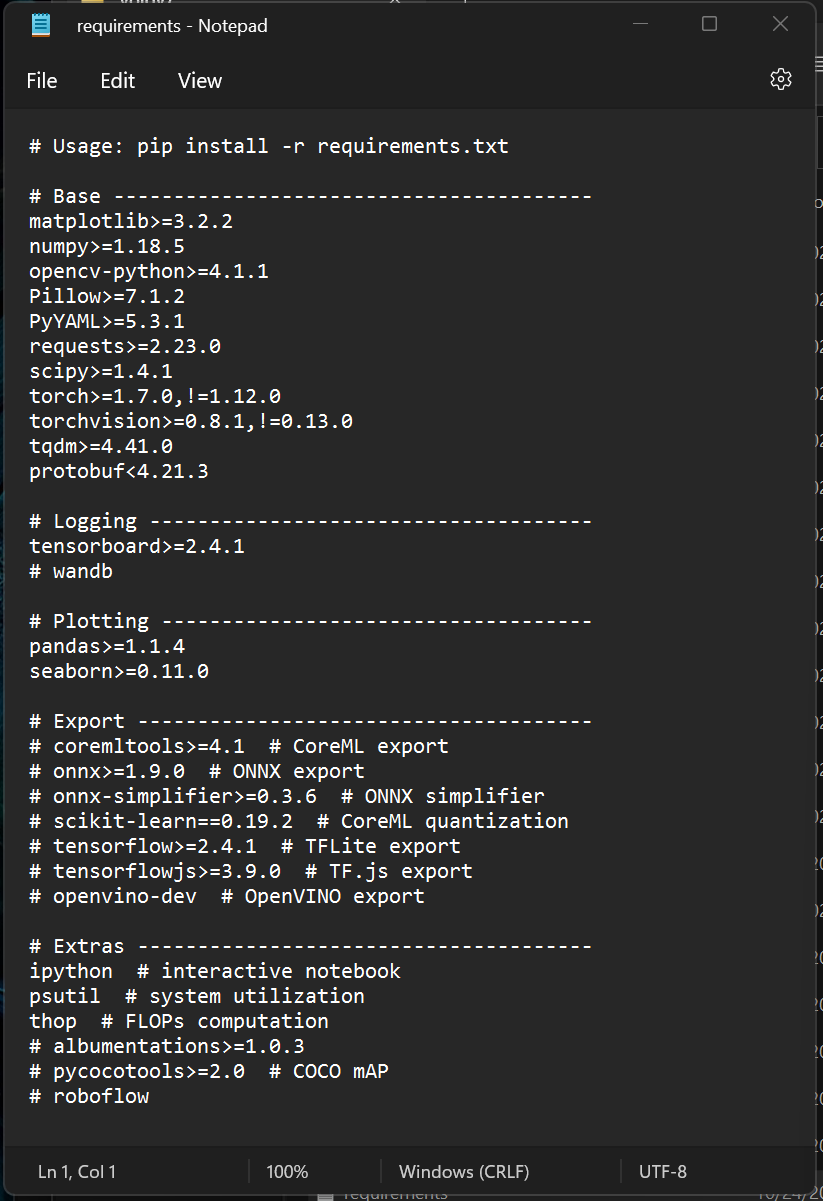
We have to clone the URL to the terminal and use the Git function as follows:



When it’s finished, we will get the folder conation all the required files for the YOLOv7 algorithm.

After we install YOLOv7, now we have to install the required library from requirement.txt folder as follows:



The requirements.txt contains the following library :

To run the model, a configuration file cfg is required, this file contains the information about the model such as model type, feature extractor type, adjustable parameters for training and testing, and post-processing score converter such Initially configuration file usually contains optimal configurations from previous training on a different dataset.

Another necessary file is the “custom. YAML” file, this file basically contains an identification number for each class to train in the model. Figure [] is an example of a YMAL file in this project.

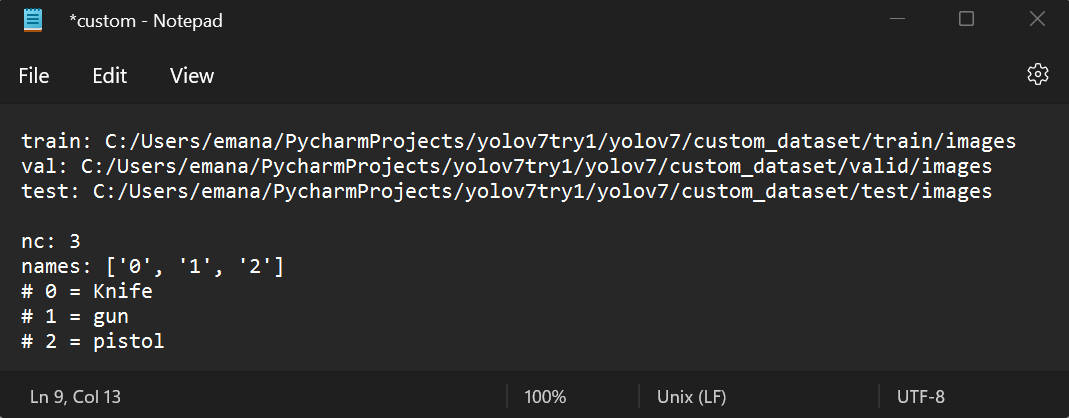
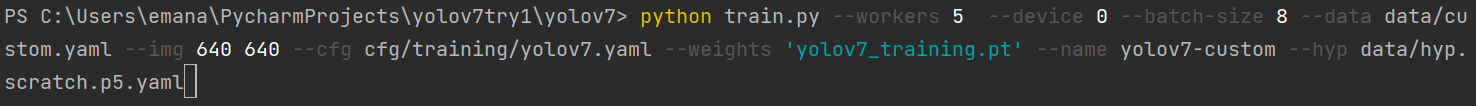
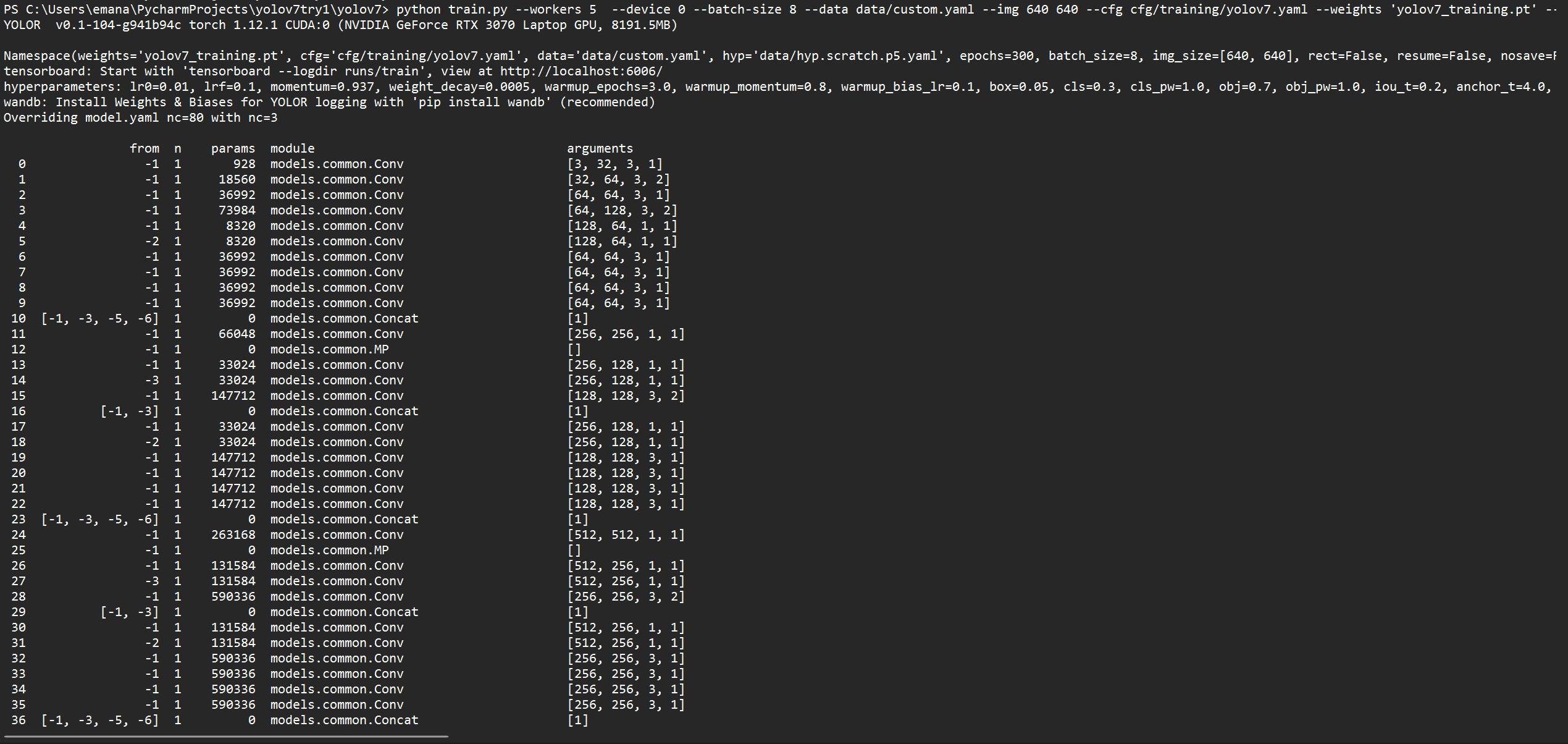


Figure 3-5 Example of a YAML file

Now YOLO is required to use CUDA with PyTorch to tun the code in GPU (Parallel) as follow



Once all the necessary documents and images are prepared, the next step is to run a command to run “train.py” in the object detection folder. this script will then load all the necessary files that have been installed to the PyCharm folder and the necessary library such as PyTorch to start training. Figure 3-7, is the demonstration of the training process on PyCharm



The figure here shows the start of the training.

4.2.2 Model Inferencing

In this stage, the trained model will be tested on a new input dataset taken from a webcam to check whether the model does what it is designed for. We used the “detect.py” with source = 0 to denote to webcam and the weight result from training “yolo\_custum.py”



4.2.3 internal Design

After the model is sufficiently trained and exported as an inference graph, the internal design of the system 1s illustrated in Figure 3-8, Firstly, the system loads the trained model and activates the webcam, and then takes input data from the webcam which is passed to the trained model. Then the models will make a prediction on how likely the image is a gun, knife, or pistol after it processed the input data from the webcam. Finally, the system displays the output of the objects detected with predicted bounding boxes, labels, and percentage of accuracy on the webcam video.

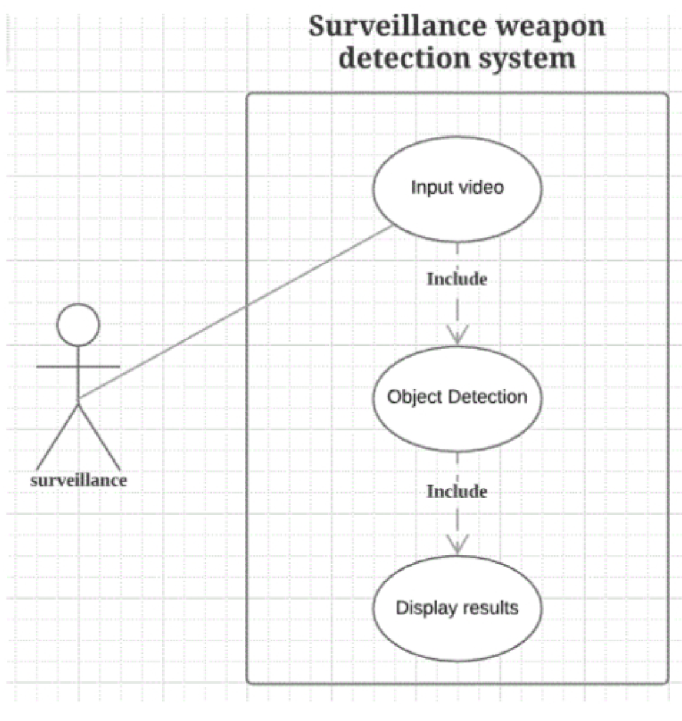
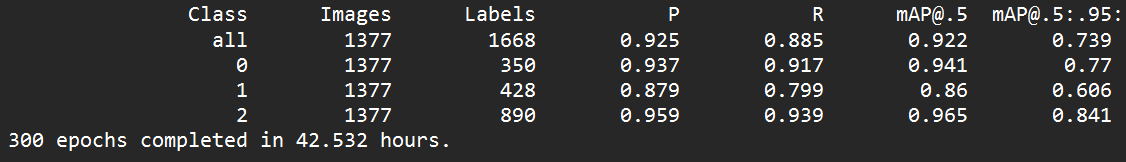


Figure 3-8 Use case of the system

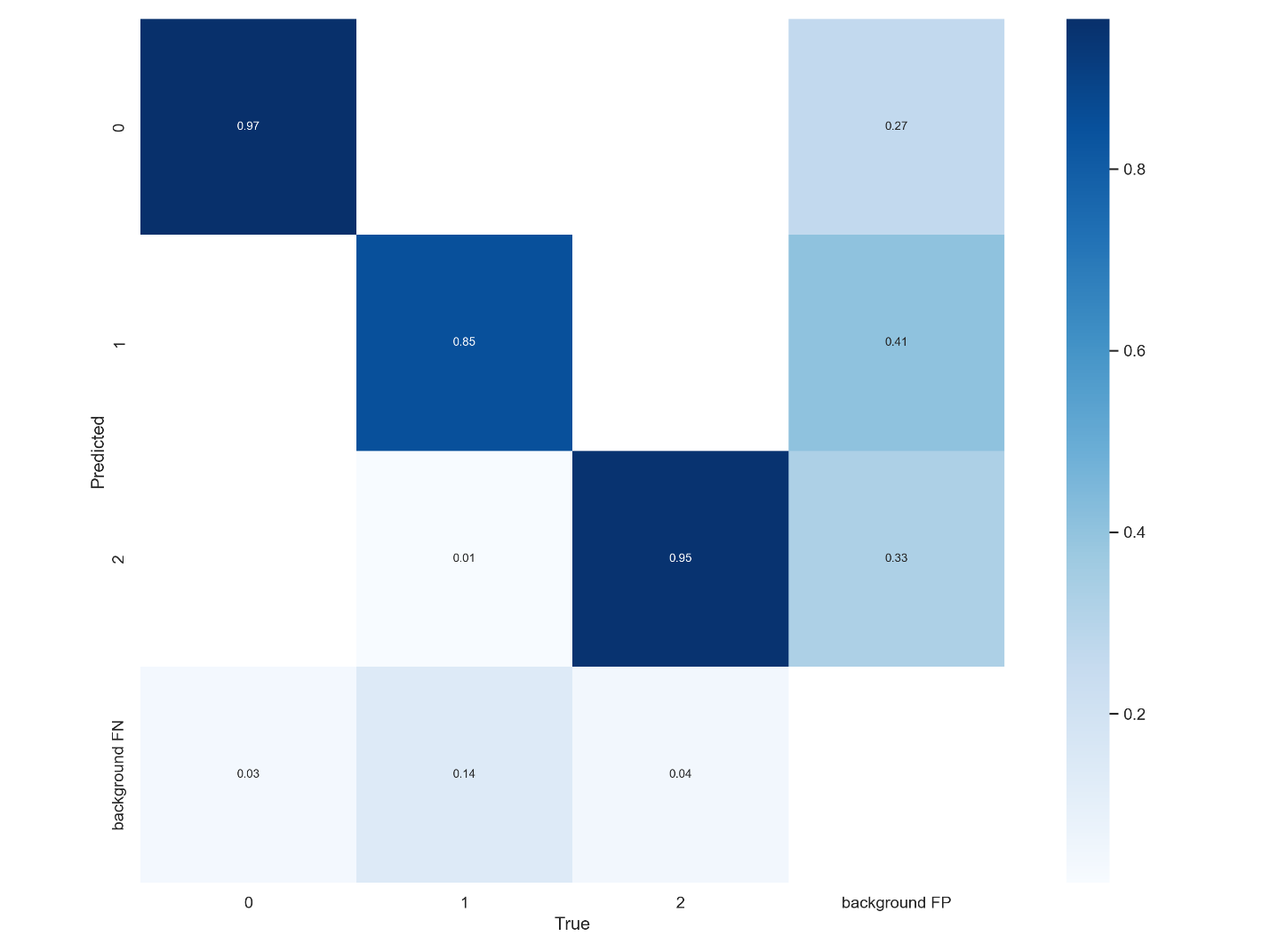
4.3 Results

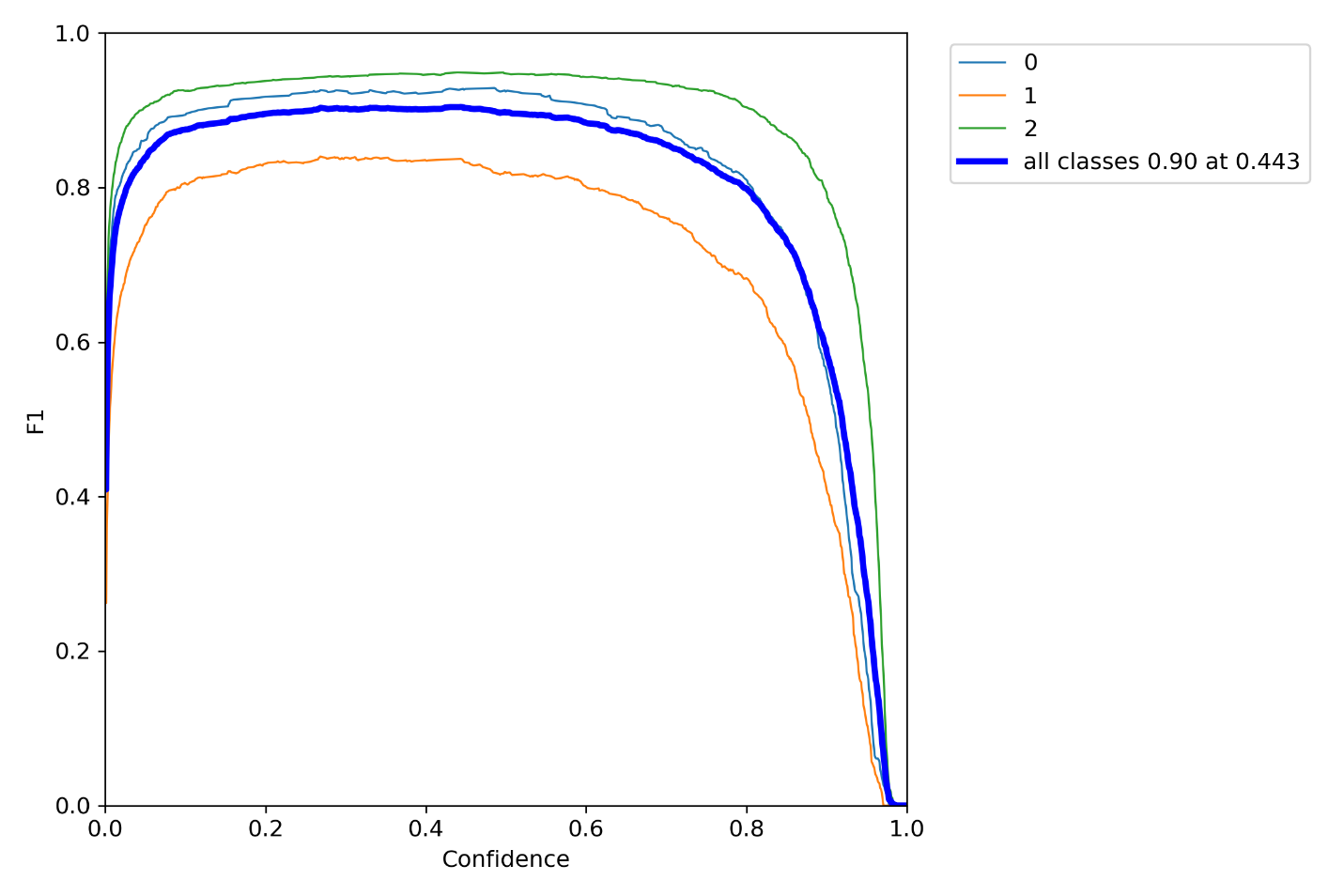
4.1.1 Introduction  
In this chapter, the results obtained from the training and testing process of the trained model YOLOv7 are presented as well as the inference result from the trained model

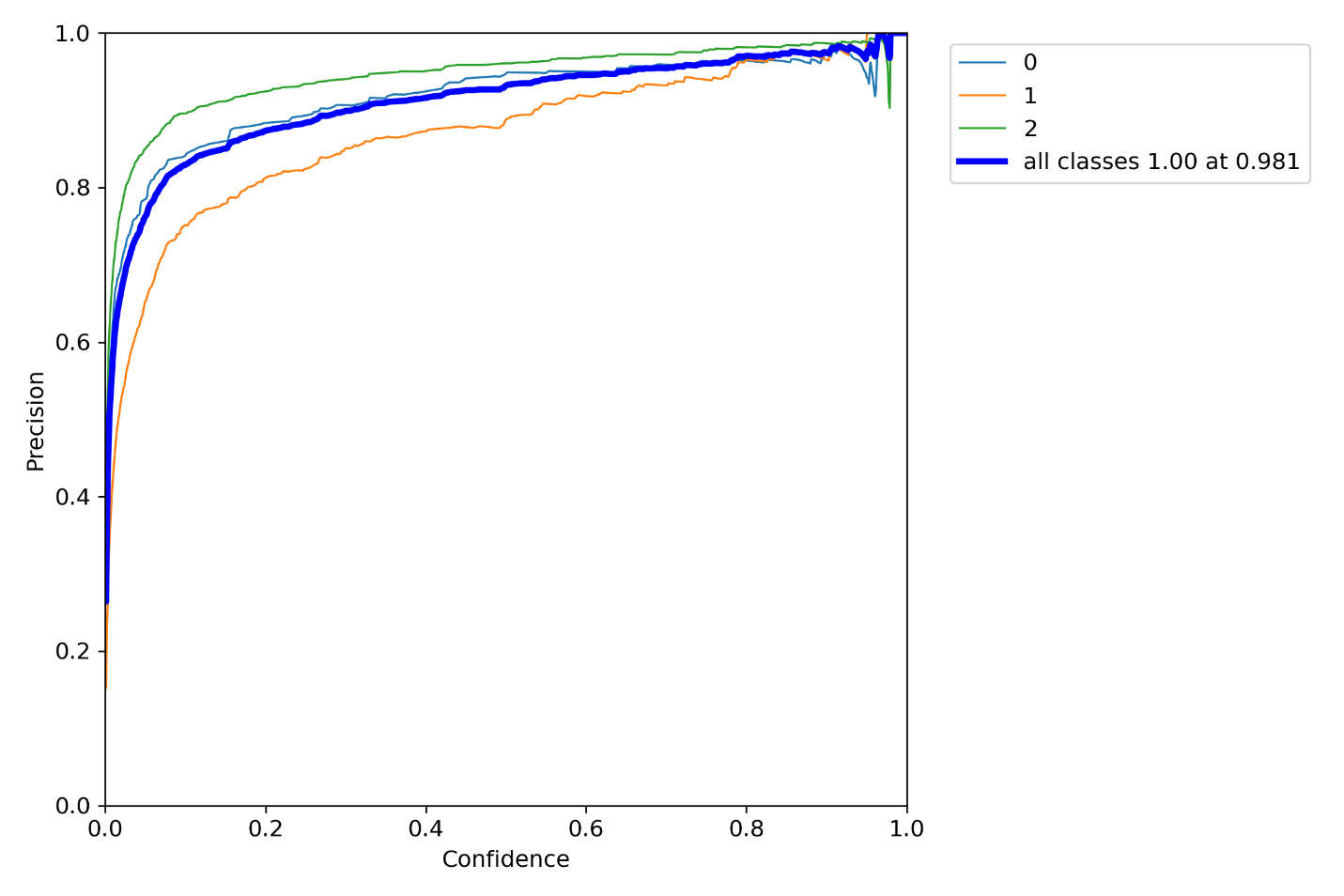
4.1.2 Training Result

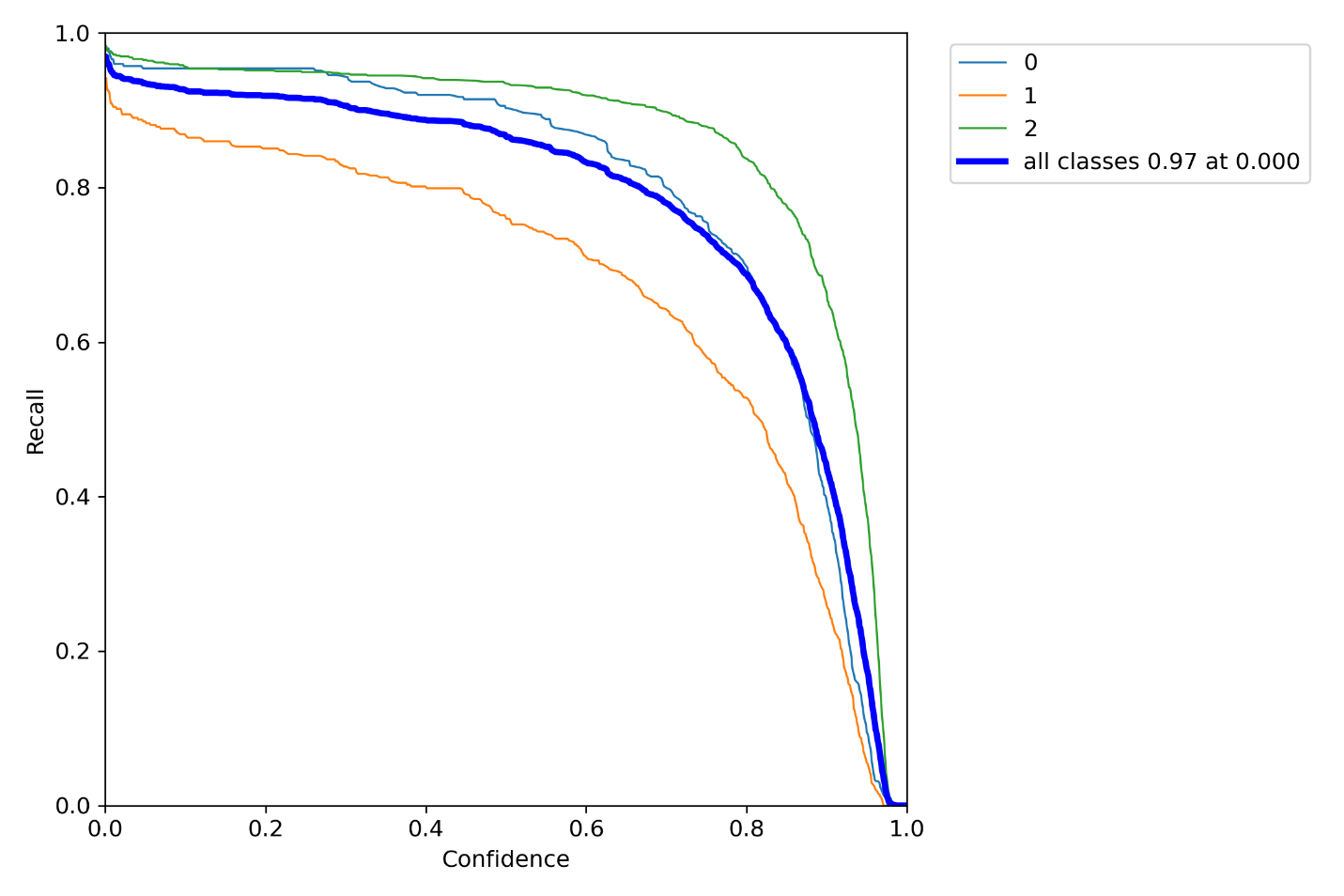
During the process of training, the total Precision, Recall, mAP@5, mAP@0.5:0.95: F1 for each global step is recorded and saved in event files, The training time taken for this research is completed in 42.532 hours. the image [] shows the result for the training.

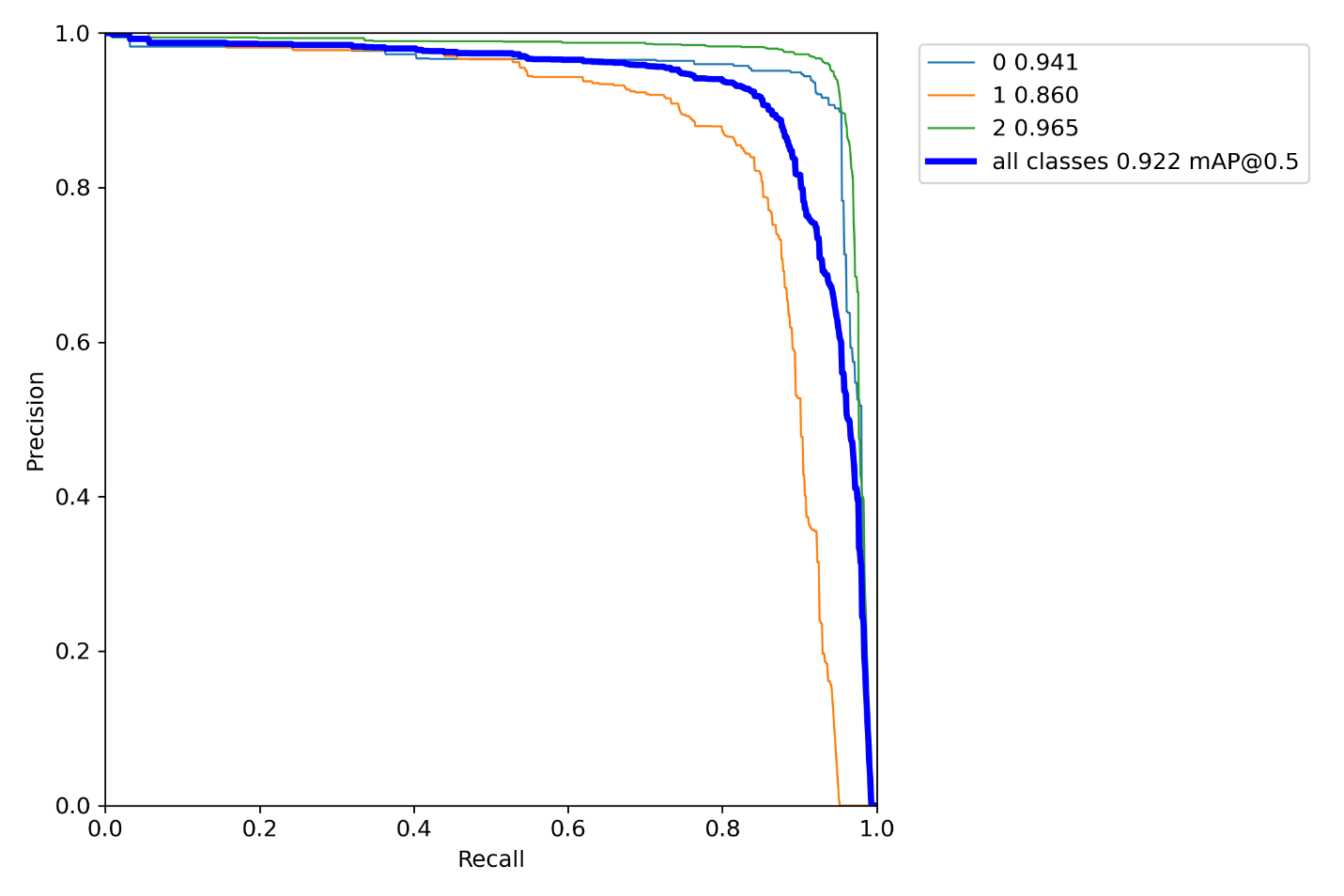
1. The confusion matrix figure is shown below:



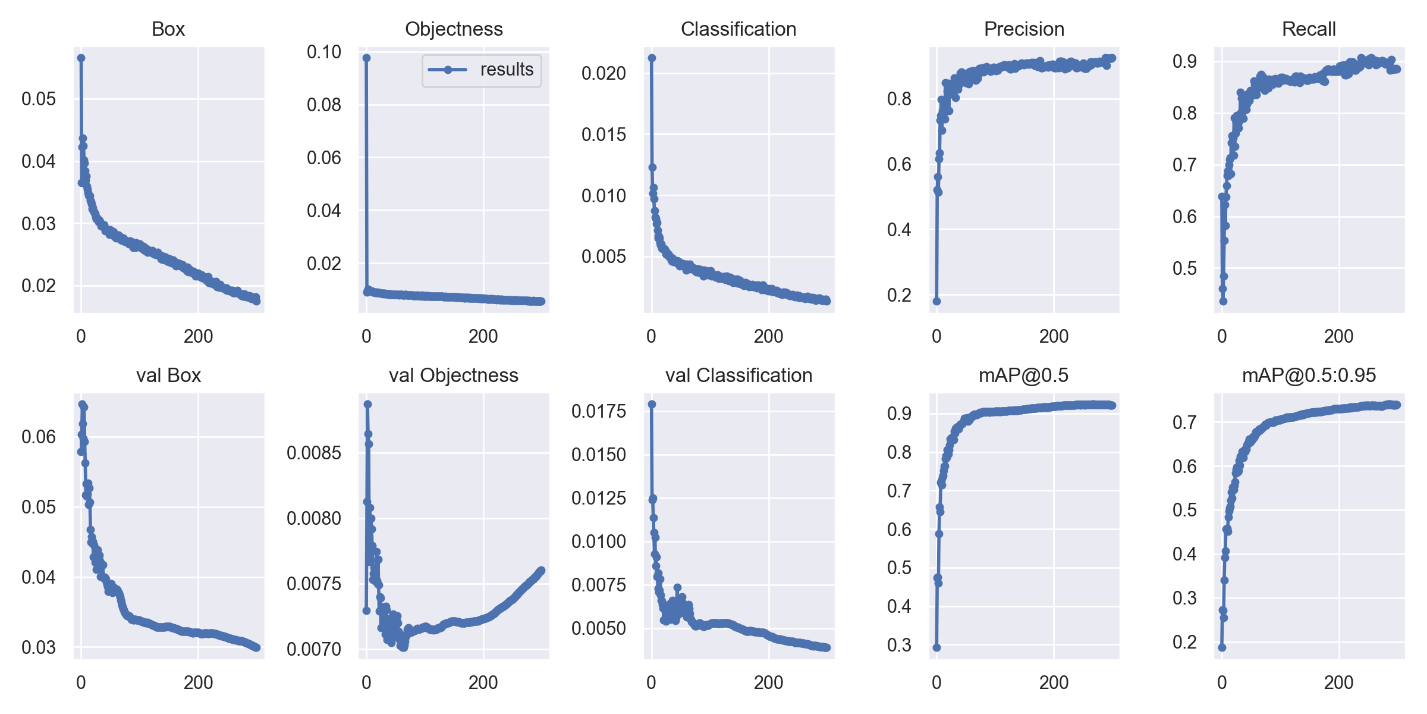
1. F1 curve figure is shown below:
2. Precision figure is shown below:



1. Recall figure is shown below:
2. Precision-Recall curves figure is shown below:



1. Results figure is shown below:



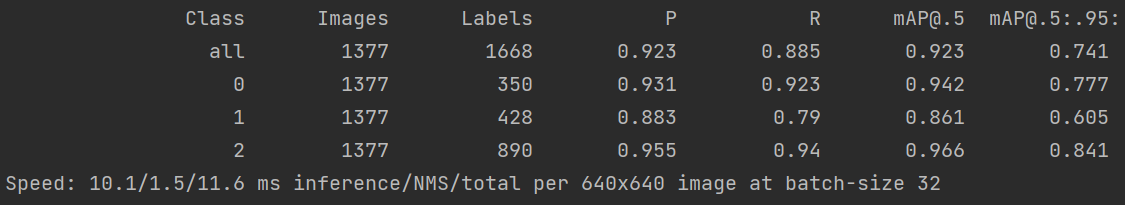
And here we can see some labeled data before and after training in figure [] :

1. original data :
2. Predicted data :



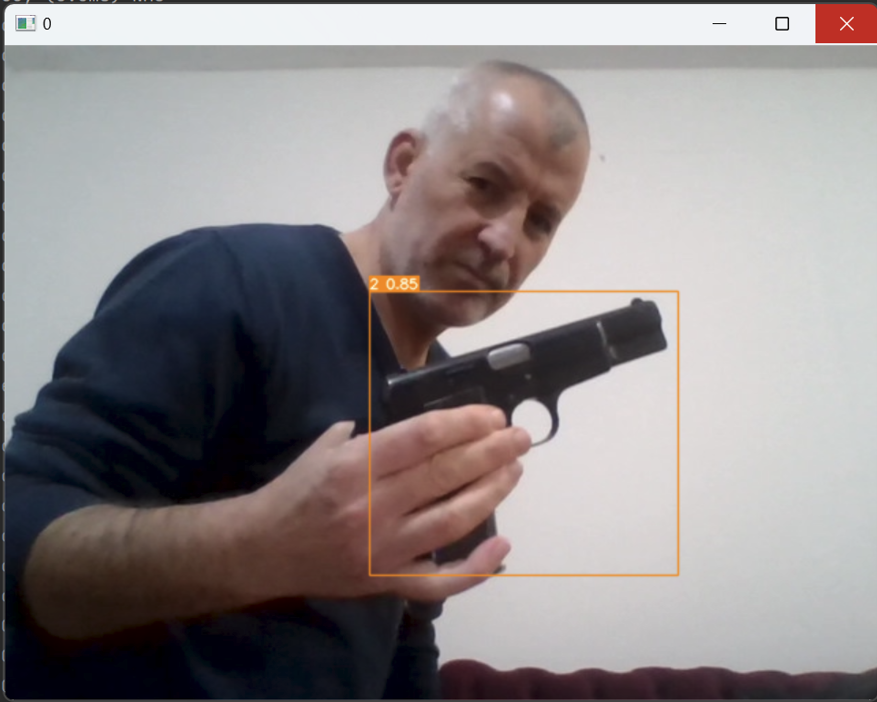
4.1.2 Testing Results

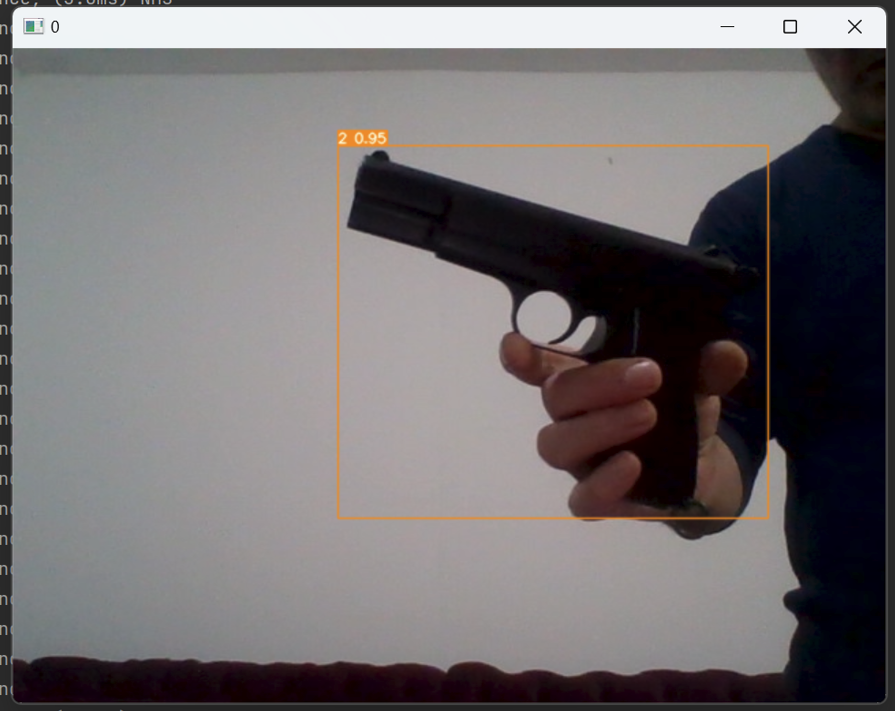
The output of the testing results was obtained after the testing process finished processing  
the 1,482 test images of all classes. The time taken for the testing process is less than one minute for 1,482 test images. The images below show the test result.



4.1.3 Inferencing Results

In this step, the trained YOLO-V7 model will be used to perform inferencing on the input video from a webcam frame by frame. The following figures are showing the inferencing results from weapon detection taken from the webcam. Figures [][] show the case of true positive (TP) for the knife, The Inception model correctly detected the knife as a knife with an accuracy of 85%, and 95% accuracy.





Chapter 5: Discussion & Recommendation