Configuration Manual

MSc Research Project
Data Analytics

Daeun Sim
Student ID: x21209162

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Daeun Sim
Student ID:	x21209162
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Hicham Rifai
Submission Due Date:	14/08/2023
Project Title:	Configuration Manual
Word Count:	1097
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Daeun Sim
Date:	18th September 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Daeun Sim
x21209162

1 Introduction

This manual is for building the personal protective equipment (PPE) detection model which is trained with the YOLOv8x model. The configuration steps include a dataset collection, labelling, build the model, and evaluate the model using a specific python library.

2 System Configuration

2.1 Software Requirements

The two cloud Integrated Development Environment (IDE), which are Kaggle Notebook, and Google Collaboratory were used to build the object detection model. Both platforms are a form of website, so they can be used without installation. And the two platforms allow use free GPU. This research project was run on the GPU to accelerate training speed. As a programming language, python was adopted, and Google Chrome browser was used as shown in Figure 1. All libraries that are used to train the model are based on Python module.

Language	Python	
Browser	Google Chrome	
Cloud Platform	Kaggle Kernels	Google Colaboratory
GPU Specification	NVIDIA Tesla T4 GPUS	NVIDIA Tesla T4 GPU

Figure 1: Software specifications

2.2 Configuration setup

- Kaggle Notebook
  On the Kaggle Notebook, one environment setup and two libraries installation needed to download dataset and train the model. Before installing libraries, the Internet usage option setting must be enabled. Figure 2 shows the step to activate the Internet usage setting: (1) Verify the phone number. This is set on the Account Settings and can be accessed by click on the profile. (2) Activate the Internet usage. This is set on the Notebook Settings. By clicking the right-side arrow icon, the setting menu is displayed. The Internet option is in the Notebook options.

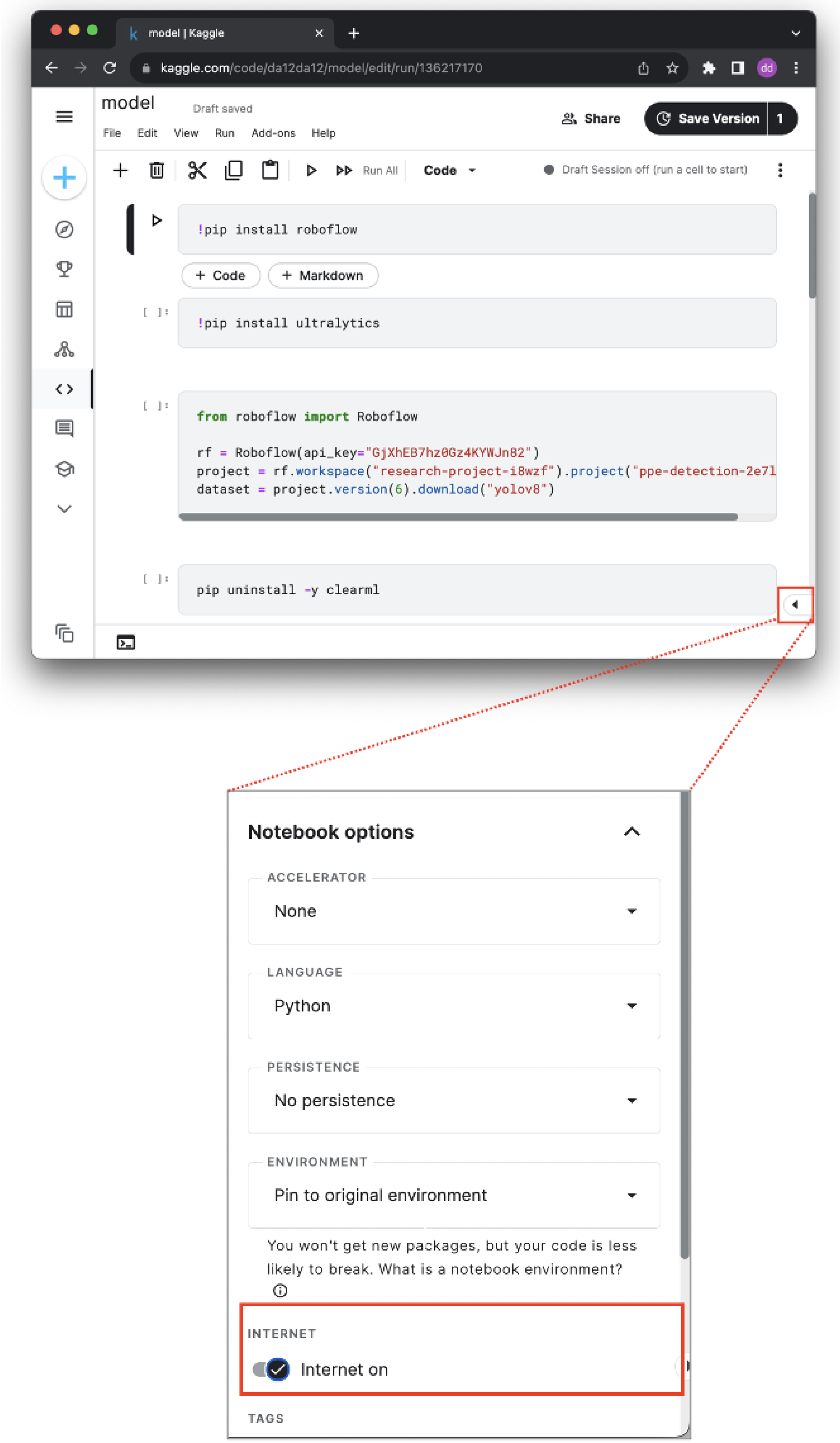

Figure 2: Enable the Internet usage setting in Kaggle Notebook

- Activate GPU accelerator in the Kaggle Notebook
  The three-dot icon on the right-side in the Notebook give access the environment configuration, as shown in Figure 3. In the sixth menu, in the Accelerator sub-menu, the GPU T4 x2 type is selected.

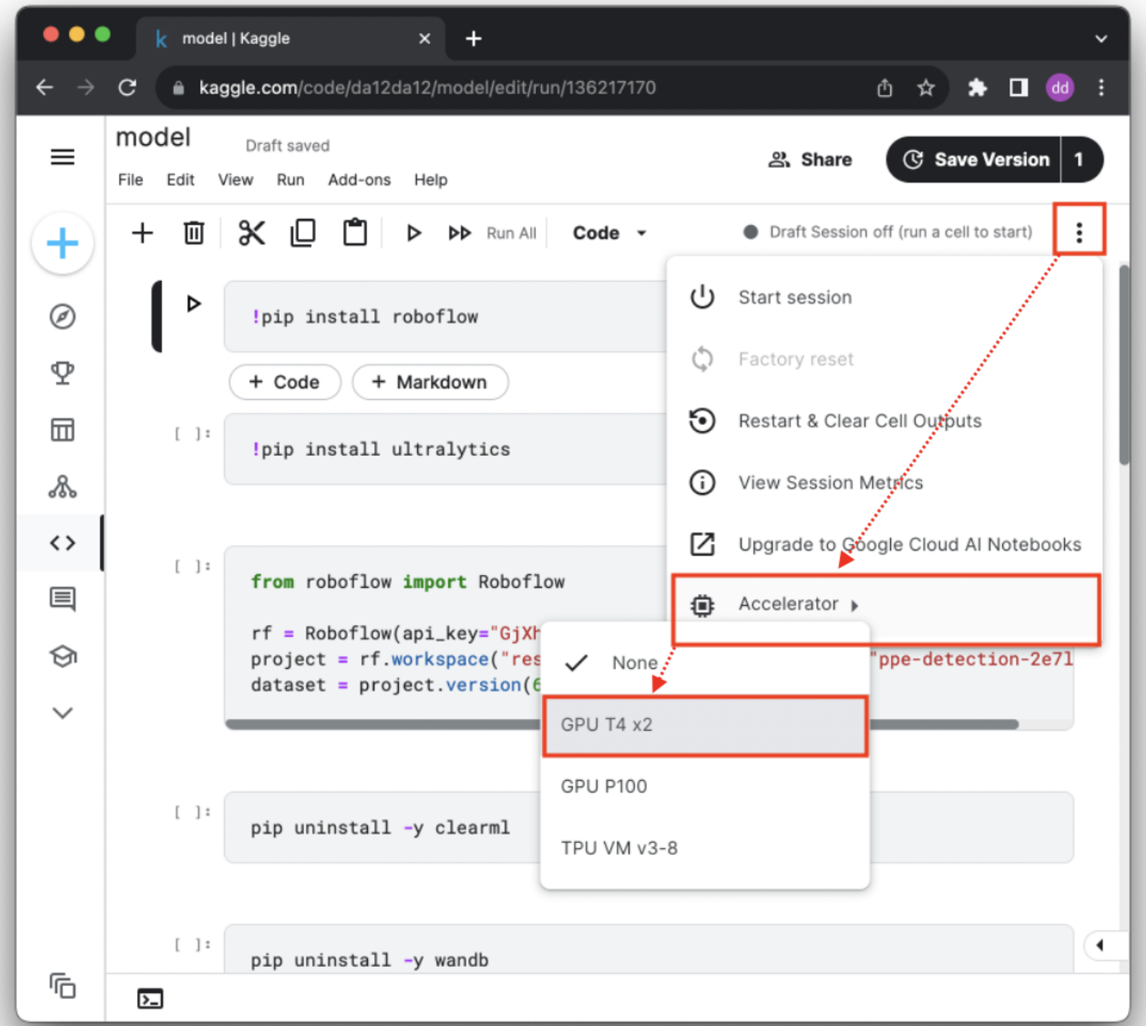

Figure 3: Enable GPU accelerator in Kaggle Notebook

- Google Colab Notebook
  On the Google Colab Notebook, two configuration is needed: Activate GPU usage and mount Google Drive. The prediction phase was executed using the GPU, and the test dataset and the result for the prediction were stored in Google Drive. This can keep the files even if the notebook is closed. Figure 4 shows the Hardware Accelerator settings in the Colab Notebook: (1) Click Runtime menu and Change runtime type. (2) Select Python3 and T4 GPU. To access the Google Drive can be executed the code in the Figure 5. This must be run whenever the notebook is opened.

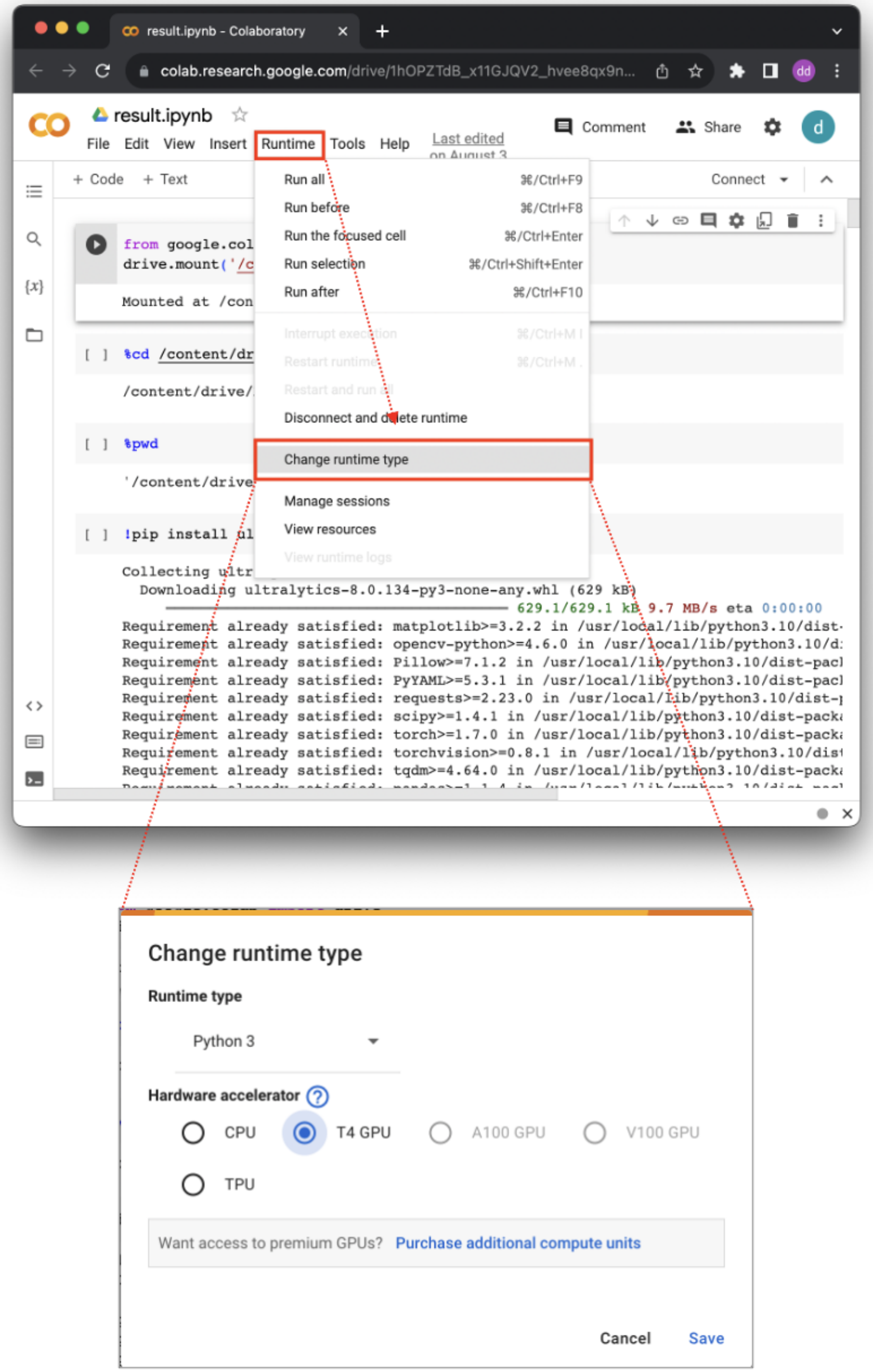

Figure 4: Enable GPU accelerator in Google Colab Notebook

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

Mounted at /content/drive
```

Figure 5: Setup access to Google Drive in Google Colab Notebook

3 Data collection

The dataset was obtained from an open-source platform.[1] There are various topics of image datasets for research project. By using a keyword "PPE detection", the dataset was found in the platform. It was downloaded in the local file system to labelling task. The platform is a website that provides graphic interface. So, the download step was executed to click "Download the Dataset" button.

4 Data pre-processing

4.1 Data labelling

The dataset was downloaded contained 13 classes. This study needed the four items of safety equipment. Therefore, the new classes were named to have only four items using an annotation tool.[2] The new names of the items are "helmet", "vest", "glove", and "boots". The step to capture bounding boxes of the four objects consists of the two steps. Firstly, the dataset is needed to upload the annotation platform. Lastly, dragging boxes is where the objects are in the images and then entering class's name as shown in Figure 6.

Figure 6: Example of labelling for helmet, vest, and boots

[1] https://universe.roboflow.com
[2] https://roboflow.com/annotate

4.2 Data augmentation

After labelling task, the dataset is 1,490 images. Three data augmentation techniques were applied to increase and improve the detection performance. This can be applied in the same platform where the labelling task was performed.[2] When the labelling step was completed, the new dataset was deployed to the platform where the dataset was collected. As the tool feature, the augmentation techniques are applied after splitting dataset into the three sub-datasets, which are the training, validation, and test dataset. The dataset was divided according to the percentage set in the tool.

5 Implementation

5.1 Install Python Packages and Import Dataset

Two libraries were installed for downloading dataset and pre-trained model in Figure 7 and 8. The dataset that was completed pre-processing phase was stored in the same platform where it is acquired. The platform provide Application Programming Interface (API) key to download the dataset in the Notebook. Figure 9 shows the download code using API. The dataset was stored in Output folder in Kaggle Notebook.

```
!pip install roboflow

Collecting roboflow
  Downloading roboflow-1.1.0-py3-none-any.whl (57 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.0/57.0 kB 2.1 MB/s eta 0:00:00
Collecting certifi==2022.12.7 (from roboflow)
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 155.3/155.3 kB 7.7 MB/s eta 0:00:00
Collecting chardet==4.0.0 (from roboflow)
  Downloading chardet-4.0.0-py2.py3-none-any.whl (178 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 178.7/178.7 kB 14.0 MB/s eta 0:00:00
Collecting cycler==0.10.0 (from roboflow)
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting idna==2.10 (from roboflow)
  Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.8/58.8 kB 5.5 MB/s eta 0:00:00
```

Figure 7: Install roboflow library

```
!pip install ultralytics

Collecting ultralytics
  Downloading ultralytics-8.0.131-py3-none-any.whl (626 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 626.9/626.9 kB 11.8 MB/s eta 0:00:00a 0:00:01
```

Figure 8: Install ultraytics library

```
from roboflow import Roboflow

rf = Roboflow(api_key="GjXhEB7hz0Gz4KYWJn82")
project = rf.workspace("research-project-i8wzf").project("ppe-detection-2e71f")
dataset = project.version(6).download("yolov8")
```

```
loading Roboflow workspace...
loading Roboflow project...
Dependency ultralytics<=8.0.20 is required but found version=8.0.131, to fix: `pip install ultralytics<=8.0.20
`
Downloading Dataset Version Zip in PPE-Detection-6 to yolov8: 100% [348193514 / 348193514] bytes

Extracting Dataset Version Zip to PPE-Detection-6 in yolov8:: 100%|██████████| 7212/7212 [00:02<00:00, 3327.39
it/s]
```

Figure 9: Download dataset to Kaggle Notebook

5.2 Build the YOLOv8x Model

```
from ultralytics import YOLO

path = "/kaggle/working/PPE-Detection-6/data.yaml"

# Load a model
model = YOLO('yolov8x.pt')  # load a pretrained model (recommended for training)

# Train the model
train_result = model.train(data=path, batch=8, epochs=100, imgsz=640, save_period=10)
print(train_result)
```

```
                   from  n    params  module                                       arguments
  0                  -1  1      2320  ultralytics.nn.modules.conv.Conv             [3, 80, 3, 2]
  1                  -1  1    115520  ultralytics.nn.modules.conv.Conv             [80, 160, 3, 2]
  2                  -1  3    436800  ultralytics.nn.modules.block.C2f             [160, 160, 3, True]
  3                  -1  1    461440  ultralytics.nn.modules.conv.Conv             [160, 320, 3, 2]
  4                  -1  6   3281920  ultralytics.nn.modules.block.C2f             [320, 320, 6, True]
  5                  -1  1   1844480  ultralytics.nn.modules.conv.Conv             [320, 640, 3, 2]
  6                  -1  6  13117440  ultralytics.nn.modules.block.C2f             [640, 640, 6, True]
  7                  -1  1   3687680  ultralytics.nn.modules.conv.Conv             [640, 640, 3, 2]
  8                  -1  3   6969600  ultralytics.nn.modules.block.C2f             [640, 640, 3, True]
  9                  -1  1   1025920  ultralytics.nn.modules.block.SPPF            [640, 640, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  3   7379200  ultralytics.nn.modules.block.C2f             [1280, 640, 3]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  3   1948800  ultralytics.nn.modules.block.C2f             [960, 320, 3]
 16                  -1  1    922240  ultralytics.nn.modules.conv.Conv             [320, 320, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  3   7174400  ultralytics.nn.modules.block.C2f             [960, 640, 3]
 19                  -1  1   3687680  ultralytics.nn.modules.conv.Conv             [640, 640, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  3   7379200  ultralytics.nn.modules.block.C2f             [1280, 640, 3]
 22        [15, 18, 21]  1   8721820  ultralytics.nn.modules.head.Detect           [4, [320, 640, 640]]
Model summary: 365 layers, 68156460 parameters, 68156444 gradients
```

Figure 10: The summary of the YOLOv8x model

The above code is for importing the pre-trained model, YOLOv8x model and setup the hyperparameters to train the model with the dataset. 'path' variable is to indicate where the training, validation, and test dataset is located in. When the above code is executed, the outline of the model's architecture is provided.

```
100 epochs completed in 6.667 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 136.7MB
Optimizer stripped from runs/detect/train/weights/best.pt, 136.7MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.131 🚀 Python-3.10.10 torch-2.0.0 CUDA:0 (Tesla T4, 15110MiB)
Model summary (fused): 268 layers, 68127420 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 10/10 [00:06<00:00,
1.62it/s]
                   all        152        670      0.956      0.945      0.965      0.695
                 boots        152        146      0.892      0.846        0.9      0.652
                 glove        152        216      0.981      0.965      0.986      0.652
                helmet        152        152      0.972          1      0.989       0.74
                  vest        152        156      0.978      0.968      0.982      0.738
Speed: 1.6ms preprocess, 24.8ms inference, 0.0ms loss, 2.6ms postprocess per image
Results saved to runs/detect/train
```

Figure 11: Training result and validation of the YOLOv8x model

Figure 11 is given when the training is completed. The trained model is saved the form of a 'best.pt' file. The trained model is examined with the validation dataset. In addition to the time spent on learning, the number of images for each class is displayed and the accuracy is calculated.

6 Assessment the trained model

The test dataset is used to evaluate the trained model. The test dataset and the 'best.pt' file are needed to assess the model. This step can be executed in the Google Colab Notebook to use GPU. Mounting Google Drive, installing libraries, and downloading dataset is consistent with the code in Figures 5, 7, 8 and 9.

6.1 Prediction with the trained model

The trained model is loaded as shown in Figure 12. And then the prediction is made of what class each object is. Figure 13 shows the codes for the prediction and for displaying the prediction result.

```
from ultralytics import YOLO
model = YOLO('/content/drive/MyDrive/best.pt')

WARNING ⚠️ /content/drive/MyDrive/best.pt appears to require 'dill', which is not in ultralytics requirements.
AutoInstall will run now for 'dill' but this feature will be removed in the future.
Recommend fixes are to train a new model using the latest 'ultralytics' package or to run a command with an official YOLOv8 model,
requirements: Ultralytics requirement ['dill'] not found, attempting AutoUpdate...
Collecting dill
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 4.7 MB/s eta 0:00:00
Installing collected packages: dill
Successfully installed dill-0.3.6

requirements: AutoUpdate success ✅ 5.0s, installed 1 package: ['dill']
requirements: ⚠️ Restart runtime or rerun command for updates to take effect
```

Figure 12: Load the trained model

```
path = '/content/PPE-Detection-6/test/images'
pred_results = model.predict(source=path, conf=0.5, iou=0.5, save=True, save_conf=True, save_txt=True)
print(pred_results)
```

```
image 284/298 /content/PPE-Detection-6/test/images/ppe_0596_jpg.rf.b0486e1e0afabe277fd1df0df8de876f.jpg: 640x640 2 helmets, 1 vest, 3997.6ms
image 285/298 /content/PPE-Detection-6/test/images/ppe_0597_jpg.rf.1d526e931f9591224f4b798b1e8b8fdc.jpg: 640x640 2 bootss, 1 glove, 2 helmets, 5860.4ms
image 286/298 /content/PPE-Detection-6/test/images/ppe_0598_jpg.rf.1908819d100e25440c807eb2b8f72764.jpg: 640x640 2 helmets, 2 vests, 4043.1ms
image 287/298 /content/PPE-Detection-6/test/images/ppe_0601_jpg.rf.b39f4fba3f644adb63a649e5f9b890e9.jpg: 640x640 1 boots, 2 helmets, 2 vests, 4064.8ms
image 288/298 /content/PPE-Detection-6/test/images/ppe_0612_jpg.rf.04c715e31ad79c6c0b8f083e1c89b39e.jpg: 640x640 2 helmets, 1 vest, 5982.4ms
image 289/298 /content/PPE-Detection-6/test/images/ppe_0620_jpg.rf.057d2219a4fb338a230ffe9d5da3f0d0.jpg: 640x640 6 helmets, 4023.0ms
image 290/298 /content/PPE-Detection-6/test/images/ppe_0628_jpg.rf.8031cffbf9ae0d5499ee4d1bd5b81ce8.jpg: 640x640 3 helmets, 3979.5ms
image 291/298 /content/PPE-Detection-6/test/images/ppe_0636_jpg.rf.1e768b9b6262aa092d4f35d73828cf3e.jpg: 640x640 3 bootss, 4 gloves, 2 helmets, 2 vests,
image 292/298 /content/PPE-Detection-6/test/images/ppe_0637_jpg.rf.bb59ad23cb2f258c85c96e2aa6b76405.jpg: 640x640 2 helmets, 4145.3ms
image 293/298 /content/PPE-Detection-6/test/images/ppe_0638_jpg.rf.13b44762adbaf1fa2656b786a116318e.jpg: 640x640 2 helmets, 1 vest, 4014.3ms
image 294/298 /content/PPE-Detection-6/test/images/ppe_0640_jpg.rf.67660ea7db1e3d7fafcdb3b7acee07ee.jpg: 640x640 2 helmets, 3 vests, 5432.2ms
image 295/298 /content/PPE-Detection-6/test/images/ppe_0648_jpg.rf.efbd63d94e5bf16ffdaf9e6e897a883d.jpg: 640x640 5 helmets, 5 vests, 4505.7ms
image 296/298 /content/PPE-Detection-6/test/images/ppe_0661_jpg.rf.b9bbccd7746e194474e260a5f12cab36.jpg: 640x640 3 helmets, 4008.5ms
image 297/298 /content/PPE-Detection-6/test/images/ppe_0668_jpg.rf.3693dfd25d4d8457e90ead742d38d61f.jpg: 640x640 1 glove, 2 helmets, 2 vests, 5141.4ms
image 298/298 /content/PPE-Detection-6/test/images/ppe_0676_jpg.rf.051fac13bb671ee443496c2b0343d4bf.jpg: 640x640 2 helmets, 2 vests, 4956.2ms
Speed: 3.2ms preprocess, 4761.1ms inference, 1.5ms postprocess per image at shape (1, 3, 640, 640)
Results saved to runs/detect/predict
279 labels saved to runs/detect/predict/labels
[ultralytics.yolo.engine.results.Results object with attributes:
```

```
import os, glob
from IPython.display import Image, clear_output, display

path = "/content/drive/MyDrive/runs/detect/predict"
os.chdir(path)

for f in glob.glob("*.jpg"):
    img = Image(filename=f"{path}/{f}", width=600)
    display(img)
```

Figure 13: The prediction process and the prediction result

6.2 Evaluation the trained model

The evaluation library is required to evaluate the prediction result. mAP and F1 score are calculated using Object-Detection-Metrics library.[3] For using the library, the two requirements are needed. Converting the text file which is the YOLO format to the xml file, which has PASCAL VOC format and the library's source codes.[4] The python files can be copied to Google Drive. The converting code is obtained from Kaggle and is modified

[3]https://github.com/eypros/Object-Detection-Metrics
[4]https://github.com/ultralytics/ultralytics/issues/2042

for this project's dataset.[5] '!python' command can be used to run the evaluation library in the Colab Notebook as shown in Figure 14. The F1 score is calculated from the results generated from the code in Figure 14. The F1 score calculation code is shown in Figure 15.

```
!python /content/drive/MyDrive/Object-Detection-Metrics/pascalvoc.py -g /content/drive/MyDrive/gt/ -d /content/drive/MyDrive/pt/ -t 0.5

/content/drive/MyDrive/Object-Detection-Metrics/pascalvoc.py:432: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if len(errors) is not 0:
Figure(640x480)
AP: 0.50760 (0)
Figure(640x480)
AP: 0.54074 (1)
Figure(640x480)
AP: 0.93327 (2)
Figure(640x480)
AP: 0.92215 (3)
mAP: 0.72594
```

Figure 14: Code for calculating mAP

```
def GetMetricsValues(content):
    content = content.replace(']', '')
    content = content.replace('[', '')
    str_line = content.split(':')
    str_list = str_line[1].split(',')
    num_list = [eval(i) for i in str_list]
    #print(len(num_list))
    return num_list

with open('/content/drive/MyDrive/Object-Detection-Metrics/results/results.txt') as f:
  lines = f.readlines()
  content = [line.strip() for line in lines]
  f.close()

pre_list = []
rec_list = []
for i in range(0, len(content)):
  if "Precision:" in content[i]:
    pre_list.append(GetMetricsValues(content[i]))
  elif "Recall:" in content[i]:
    rec_list.append(GetMetricsValues(content[i]))
```

```
for i in range(0, len(pre_list)):
  img_score = 0
  total = len(pre_list[i])
  for j in range(0, total):
    precision = float(pre_list[i][j])
    recall = float(rec_list[i][j])
    score = (2*precision*recall)/(precision+recall)
    img_score = img_score + score
  print(f'class {i}')
  print(f'sum: {img_score}, total: {total}')
  print(f'f-1 score: {img_score / total}')
  print('\n')
```

Figure 15: Code for calculating F1 score

[5]https://www.kaggle.com/code/siddharthkumarsah/convert-yolo-annotations-to-coco-pascal-voc