Pepito, Alyssa Mae & Soleño, Keziah Antonette BM1

Group 18

M1 – SA: YOLOv5 Modeling

I. Libraries and Dependencies Installation

Before the Roboflow dataset, that was produced in the previous weeks, is used to create YOLOv5 models, certain files and environments were prepared before testing the data. First, we cloned the *YOLOv5 from Ultralytics YOLOv5 Repository* and installed its dependencies. Since the dataset that will be used in this model will come from Roboflow, we also installed Roboflow's designated library, as seen in the code snippet below.

```
#clone forked YOLOv5 from Ultralytics YOLOv5 Repository
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow # install roboflow

import torch
import os
from IPython.display import Image, clear_output  # to display images

print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.g
et_device_properties(0).name if torch.cuda.is_available() else 'CPU'})
")
```

The figure below displays the results of the above code snippet.

```
Cloning into 'yolov5'...
remote: Enumerating objects: 14308, done.
remote: Counting objects: 100% (66/66), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 14308 (delta 32), reused 49 (delta 23), pack-reused 14242
Receiving objects: 100% (14308/14308), 13.63 MiB | 17.40 MiB/s, done.
Resolving deltas: 100% (9832/9832), done.
/content/yolov5
     |████████████████████████████████| 182 kB 27.2 MB/s
     |████████████████████████████████| 62 kB 522 kB/s
     |████████████████████████████████| 1.6 MB 58.0 MB/s
     |████████████████████████████████| 42 kB 912 kB/s
     |████████████████████████████████| 145 kB 45.3 MB/s
     |████████████████████████████████| 138 kB 74.1 MB/s
     |████████████████████████████████| 54 kB 3.2 MB/s
     |████████████████████████████████| 178 kB 70.2 MB/s
     |████████████████████████████████| 67 kB 6.8 MB/s
     |████████████████████████████████| 62 kB 1.6 MB/s
  Building wheel for wget (setup.py) ... done
Setup complete. Using torch 1.12.1+cu113 (Tesla T4)
```

Once the Roboflow dependency had been successfully installed, we imported Roboflow by applying the following code snippet:

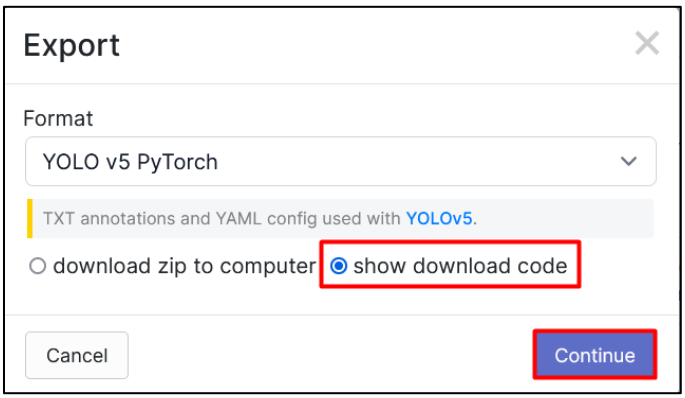
```
from roboflow import Roboflow #import roboflow for our datasets
rf = Roboflow(model_format="yolov5", notebook="ultralytics")
```

This code snippet will produce a result that will lead the programmer to a link where they can get an API key from Roboflow.

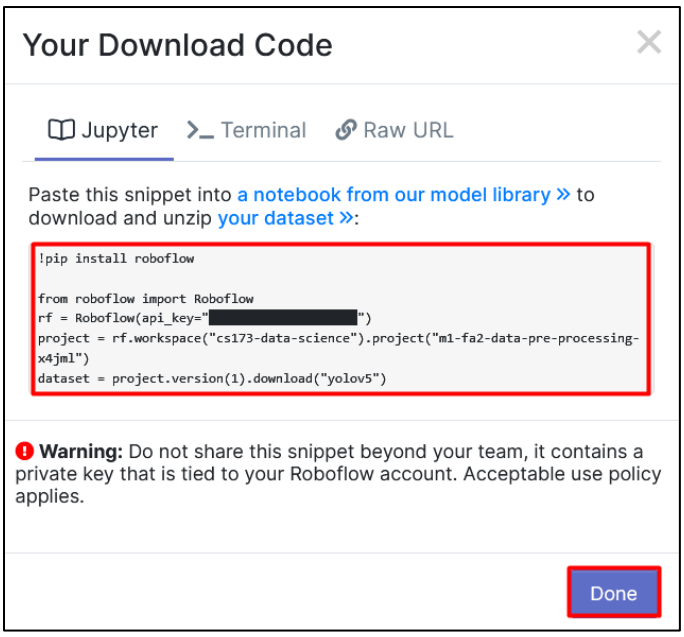
```
[3]  from roboflow import Roboflow #import roboflow for our datasets
     rf = Roboflow(model_format="yolov5", notebook="ultralytics")

     upload and label your dataset, and get an API KEY here: https://app.roboflow.com/?model=yolov5&ref=ultralytics
```

After getting the API key, you must go to your project's dataset and click on *Export*. Then, make sure that "show download code" is selected and click on *Continue*.



Copy the code snippet being shown below and click on Done.



Back to the Google Colab notebook, we first set up the environment before pasting the code snippet we just copied.

```
# set up environment
os.environ["DATASET_DIRECTORY"] = "/content/datasets"
```

Then, paste the code snippet and run it. We wait for the Roboflow dataset to be downloaded.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="RU6eGLCNP3Q4UwQqKWha")
project = rf.workspace("cs173-data-science").project("m1-fa2-data-pre-processing-x4jml")
dataset = project.version(1).download("yolov5")
```

II. Training the models (small, medium, and large)

After all necessary files and dependencies have been installed, we trained the model by implementing the *train.py* from the Ultralytics repository. The hyperparameters that were set for this model are as follows:

1. Image size = 416 pixels
2. Batch size = 16
3. Epochs = 50
4. YOLOv5 Version = YOLOv5**s** (Small)

Note: The "s" after "YOLOv5" indicates the version scale (size) of the model. One requirement of this project is to create three YOLOv5 models for scales small, medium, and large; therefore, we adjusted the YOLOv5 versions by changing the last letter to the first letter of their appropriate version scale. Since the process for each model will be the same for each model of a different weight, the model with version YOLOv5s will only be discussed in this section.

To apply the ideal hyperparameters mentioned above, the following code snippet was run, yielding the results from the succeeding figures:

```
!python train.py --img 416 --batch 16 --epochs 50 --
data {dataset.location}/data.yaml --weights yolov5s.pt --
name yolov5s_results -cache
```

```
train: weights=yolov5s.pt, cfg=, data=/content/datasets/M1---FA2-Data-Pre-Processing-1/data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=50, batch_size=16, img
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-21-ga1b6e79 Python-3.8.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0
ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLOv5 🚀 in ClearML
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 176MB/s]
Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 304MB/s]

Overriding model.yaml nc=80 with nc=5

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
```

```
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     26970  models.yolo.Detect                      [5, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]]
Model summary: 214 layers, 7033114 parameters, 7033114 gradients, 16.0 GFLOPs

Transferred 343/349 items from yolov5s.pt
AMP: checks passed ✅
optimizer: SGD(lr=0.01) with parameter groups 57 weight(decay=0.0), 60 weight(decay=0.0005), 60 bias
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid_size=(8, 8))
train: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/train/labels... 1799 images, 123 backgrounds, 0 corrupt: 100% 1799/1799 [00:00<00:00, 1984.61it/s]
train: New cache created: /content/datasets/M1---FA2-Data-Pre-Processing-1/train/labels.cache
train: Caching images (0.9GB ram): 100% 1799/1799 [00:10<00:00, 175.50it/s]
val: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels... 180 images, 17 backgrounds, 0 corrupt: 100% 180/180 [00:00<00:00, 609.53it/s]
val: New cache created: /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels.cache
val: Caching images (0.1GB ram): 100% 180/180 [00:01<00:00, 101.44it/s]

AutoAnchor: 4.75 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✅
Plotting labels to runs/train/yolov5s_results/labels.jpg...
Image sizes 416 train, 416 val
Using 2 dataloader workers
Logging results to runs/train/yolov5s_results
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       0/49      1.71G    0.09907    0.03477    0.03224         40        416: 100% 113/113 [00:23<00:00,  4.72it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:02<00:00,  2.29it/s]
                   all        180        802      0.825     0.0712     0.0164    0.00435

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       1/49      2.07G    0.07245    0.02921    0.01683         52        416: 100% 113/113 [00:19<00:00,  5.83it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  5.32it/s]
                   all        180        802      0.686     0.0944      0.059     0.0217

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       2/49      2.07G    0.06545     0.0249    0.01445         49        416: 100% 113/113 [00:19<00:00,  5.93it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  5.57it/s]
                   all        180        802      0.478      0.135     0.0737     0.0298
```

```
                 all        180        802      0.465      0.213      0.215      0.109

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      46/49      2.07G    0.02083    0.01162    0.00762         40        416: 100% 113/113 [00:20<00:00,  5.63it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:00<00:00,  6.07it/s]
                   all        180        802       0.47      0.204      0.209      0.103

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      47/49      2.07G    0.02044    0.01096    0.00753         35        416: 100% 113/113 [00:18<00:00,  6.18it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  5.90it/s]
                   all        180        802      0.462       0.21      0.216      0.108

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      48/49      2.07G    0.02077     0.0117    0.00738         45        416: 100% 113/113 [00:18<00:00,  6.02it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  5.79it/s]
                   all        180        802      0.462       0.22      0.219      0.109

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      49/49      2.07G    0.02038    0.01132   0.007088         41        416: 100% 113/113 [00:18<00:00,  6.08it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  5.94it/s]
                   all        180        802      0.457      0.215      0.218      0.109

50 epochs completed in 0.283 hours.
Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 14.3MB

Validating runs/train/yolov5s_results/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:02<00:00,  2.27it/s]
                   all        180        802      0.479      0.217      0.224      0.111
            Full-Faced        180        178      0.374      0.365      0.318      0.104
            Half-Faced        180        158      0.223      0.228      0.144     0.0485
               Invalid        180         27          0          0     0.0162    0.00572
    Not Wearing Helmet        180         29          1          0     0.0158    0.00875
                 Rider        180        410      0.796      0.494      0.627      0.389
Results saved to runs/train/yolov5s_results
```

Below are the codes and results for the **medium-scaled** model.

```
!python train.py --img 416 --batch 16 --epochs 50 --
data {dataset.location}/data.yaml --weights yolov5m.pt --
name yolov5m_results --cache ram
```

```
train: weights=yolov5m.pt, cfg=, data=/content/datasets/M1---FA2-Data-Pre-Processing-1/data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=5
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-21-ga1b6e79 Python-3.8.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05
ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLOv5 🚀 in ClearML
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 93.3MB/s]
Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5m.pt to yolov5m.pt...
100% 40.8M/40.8M [00:05<00:00, 7.44MB/s]

Overriding model.yaml nc=80 with nc=5

                from  n    params  module                                  arguments
  0                -1  1      5280  models.common.Conv                      [3, 48, 6, 2, 2]
  1                -1  1     41664  models.common.Conv                      [48, 96, 3, 2]
  2                -1  2     65280  models.common.C3                        [96, 96, 2]
  3                -1  1    166272  models.common.Conv                      [96, 192, 3, 2]
  4                -1  4    444672  models.common.C3                        [192, 192, 4]
  5                -1  1    664320  models.common.Conv                      [192, 384, 3, 2]
  6                -1  6   2512896  models.common.C3                        [384, 384, 6]
  7                -1  1   2655744  models.common.Conv                      [384, 768, 3, 2]
  8                -1  2   4134912  models.common.C3                        [768, 768, 2]
  9                -1  1   1476864  models.common.SPPF                      [768, 768, 5]
 10                -1  1    295680  models.common.Conv                      [768, 384, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
```

```
 13                -1  2   1182720  models.common.C3                        [768, 384, 2, False]
 14                -1  1     74112  models.common.Conv                      [384, 192, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  2    296448  models.common.C3                        [384, 192, 2, False]
 18                -1  1    332160  models.common.Conv                      [192, 192, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  2   1035264  models.common.C3                        [384, 384, 2, False]
 21                -1  1   1327872  models.common.Conv                      [384, 384, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  2   4134912  models.common.C3                        [768, 768, 2, False]
 24      [17, 20, 23]  1     40410  models.yolo.Detect                      [5, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 9(
Model summary: 291 layers, 20887482 parameters, 20887482 gradients, 48.3 GFLOPs

Transferred 475/481 items from yolov5m.pt
AMP: checks passed ✅
optimizer: SGD(lr=0.01) with parameter groups 79 weight(decay=0.0), 82 weight(decay=0.0005), 82 bias
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), til
train: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/train/labels... 1799 images, 123 backgrounds, 0 corrupt: 100% 1799/1799 [00:(
train: New cache created: /content/datasets/M1---FA2-Data-Pre-Processing-1/train/labels.cache
train: Caching images (0.9GB ram): 100% 1799/1799 [00:10<00:00, 176.03it/s]
val: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels... 180 images, 17 backgrounds, 0 corrupt: 100% 180/180 [00:00<00:(
val: New cache created: /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels.cache
val: Caching images (0.1GB ram): 100% 180/180 [00:01<00:00, 106.00it/s]

AutoAnchor: 4.75 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✅
Plotting labels to runs/train/yolov5m_results/labels.jpg...
Image sizes 416 train, 416 val
Using 2 dataloader workers
Logging results to runs/train/yolov5m_results
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       0/49      2.93G    0.09309    0.03582    0.03039         40        416: 100% 113/113 [00:30<00:00,  3.69it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:02<00:00,  2.14it/s]
                   all        180        802      0.847     0.0532     0.0327      0.013

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       1/49      3.47G    0.06917    0.02783    0.01633         52        416: 100% 113/113 [00:26<00:00,  4.28it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  4.17it/s]
                   all        180        802      0.679     0.0914     0.0661     0.0252

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       2/49      3.47G     0.0632    0.02317    0.01427         49        416: 100% 113/113 [00:24<00:00,  4.57it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  4.23it/s]
                   all        180        802       0.48       0.14     0.0795     0.0344

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       3/49      3.47G    0.05482    0.02079    0.01267         57        416: 100% 113/113 [00:24<00:00,  4.60it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  4.20it/s]
                   all        180        802      0.573      0.168      0.149     0.0678

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       4/49      3.47G    0.04713    0.01881    0.01179         38        416: 100% 113/113 [00:24<00:00,  4.58it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  4.18it/s]
                   all        180        802      0.629      0.226      0.192     0.0897

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      47/49      3.47G    0.01714   0.008929   0.005369         35        416: 100% 113/113 [00:24<00:00,  4.55it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  4.22it/s]
                   all        180        802      0.498      0.267      0.253      0.125

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      48/49      3.47G     0.0174   0.009438   0.005209         45        416: 100% 113/113 [00:24<00:00,  4.54it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  4.23it/s]
                   all        180        802      0.494      0.262      0.251      0.126

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      49/49      3.47G    0.01705   0.009126   0.005106         41        416: 100% 113/113 [00:24<00:00,  4.53it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  4.26it/s]
                   all        180        802      0.303      0.264      0.256       0.13

50 epochs completed in 0.381 hours.
Optimizer stripped from runs/train/yolov5m_results/weights/last.pt, 42.1MB
Optimizer stripped from runs/train/yolov5m_results/weights/best.pt, 42.1MB

Validating runs/train/yolov5m_results/weights/best.pt...
Fusing layers...
Model summary: 212 layers, 20869098 parameters, 0 gradients, 47.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:04<00:00,  1.45it/s]
                   all        180        802      0.303      0.264      0.256       0.13
           Full-Faced         180        178      0.398      0.433      0.411       0.16
           Half-Faced         180        158      0.261      0.272      0.193     0.0734
              Invalid         180         27     0.0592      0.111     0.0193     0.0103
    Not Wearing Helmet        180         29          0          0     0.0188     0.0086
                Rider         180        410      0.799      0.503      0.637      0.398
Results saved to runs/train/yolov5m_results
```

Lastly, these are the codes and results for the **large-scaled** model.

```
!python train.py --img 416 --batch 16 --epochs 50 --
data {dataset.location}/data.yaml --weights yolov5l.pt --
name yolov5l_results --cache ram
```

```
train: weights=yolov5l.pt, cfg=, data=/content/datasets/M1---FA2-Data-Pre-Processing-1/data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=5
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-21-ga1b6e79 Python-3.8.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05
ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLOv5 🚀 in ClearML
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=5

                 from  n    params  module                                  arguments
  0                -1  1      7040  models.common.Conv                      [3, 64, 6, 2, 2]
  1                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  2                -1  3    156928  models.common.C3                        [128, 128, 3]
  3                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  4                -1  6   1118208  models.common.C3                        [256, 256, 6]
  5                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  6                -1  9   6433792  models.common.C3                        [512, 512, 9]
  7                -1  1   4720640  models.common.Conv                      [512, 1024, 3, 2]
  8                -1  3   9971712  models.common.C3                        [1024, 1024, 3]
  9                -1  1   2624512  models.common.SPPF                      [1024, 1024, 5]
 10                -1  1    525312  models.common.Conv                      [1024, 512, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  3   2757632  models.common.C3                        [1024, 512, 3, False]
 14                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  3    690688  models.common.C3                        [512, 256, 3, False]
 18                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 19         [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  3   2495488  models.common.C3                        [512, 512, 3, False]
 21                -1  1   2360320  models.common.Conv                      [512, 512, 3, 2]
 22         [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  3   9971712  models.common.C3                        [1024, 1024, 3, False]
 24      [17, 20, 23]  1     53850  models.yolo.Detect                      [5, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90
Model summary: 368 layers, 46159834 parameters, 46159834 gradients, 108.3 GFLOPs

Transferred 607/613 items from yolov5l.pt
AMP: checks passed ✅
optimizer: SGD(lr=0.01) with parameter groups 101 weight(decay=0.0), 104 weight(decay=0.0005), 104 bias
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), til
train: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/train/labels.cache... 1799 images, 123 backgrounds, 0 corrupt: 100% 1799/1799
train: Caching images (0.9GB ram): 100% 1799/1799 [00:11<00:00, 161.85it/s]
val: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels.cache... 180 images, 17 backgrounds, 0 corrupt: 100% 180/180 [00:0
val: Caching images (0.1GB ram): 100% 180/180 [00:02<00:00, 84.86it/s]

AutoAnchor: 4.75 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✅
Plotting labels to runs/train/yolov5l_results2/labels.jpg...
Image sizes 416 train, 416 val
Using 2 dataloader workers
Logging results to runs/train/yolov5l_results2
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       0/49      4.77G    0.09032    0.03636    0.03056         40        416: 100% 113/113 [00:39<00:00,  2.88it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:02<00:00,  2.86it/s]
                   all        180        802      0.889     0.0671     0.0629     0.0243

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       1/49      5.49G    0.06823     0.0269    0.01615         52        416: 100% 113/113 [00:34<00:00,  3.30it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  3.39it/s]
                   all        180        802      0.864     0.0751     0.0728     0.0322
```

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      46/49      5.49G    0.01627   0.008554   0.004425         40        416: 100% 113/113 [00:33<00:00,  3.40it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  3.47it/s]
                   all        180        802      0.335      0.259      0.261      0.134

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      47/49      5.49G    0.01575   0.007904   0.004274         35        416: 100% 113/113 [00:33<00:00,  3.40it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  3.54it/s]
                   all        180        802       0.35      0.262      0.268      0.136

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      48/49      5.49G    0.01597   0.008478   0.004064         45        416: 100% 113/113 [00:33<00:00,  3.41it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  3.56it/s]
                   all        180        802      0.376      0.277      0.276      0.141

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      49/49      5.49G    0.01552   0.008144   0.003962         41        416: 100% 113/113 [00:33<00:00,  3.40it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:01<00:00,  3.54it/s]
                   all        180        802      0.361      0.254      0.271      0.142

50 epochs completed in 0.515 hours.
Optimizer stripped from runs/train/yolov5l_results2/weights/last.pt, 92.8MB
Optimizer stripped from runs/train/yolov5l_results2/weights/best.pt, 92.8MB

Validating runs/train/yolov5l_results2/weights/best.pt...
Fusing layers...
Model summary: 267 layers, 46129818 parameters, 0 gradients, 107.7 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:03<00:00,  1.71it/s]
                   all        180        802      0.328      0.263      0.271      0.143
            Full-Faced        180        178      0.432      0.452      0.404      0.162
            Half-Faced        180        158       0.33      0.297       0.25      0.108
               Invalid        180         27     0.0311      0.037       0.02    0.00761
    Not Wearing Helmet        180         29          0          0     0.0283    0.00866
                 Rider        180        410      0.846      0.527      0.651      0.428
Results saved to runs/train/yolov5l_results2
```

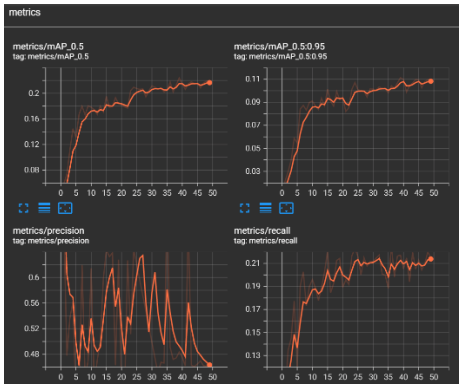
III. TensorBoard

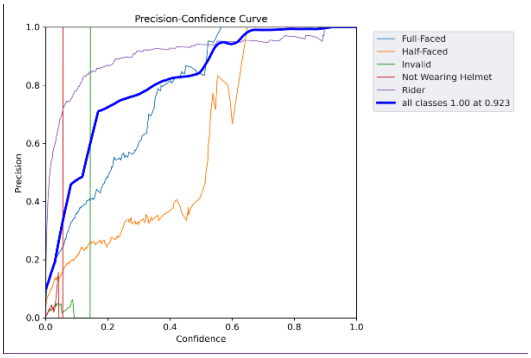
In this project, TensorBoard was used to portray the results of the training process through scalar metrics, images, graphs, and time series. A few examples of well-known scalar metrics are *mAP* (mean average precision), *precision*, and *recall*. These values determine whether the trained model is underfitting or overfitting. Under images and graphs, one will see the performance of the model through graphs that represent the *F1 curve*, *PC curve*, *confusion matrix*, and the model's overall results. The time series shows all runtimes of the model's training process.

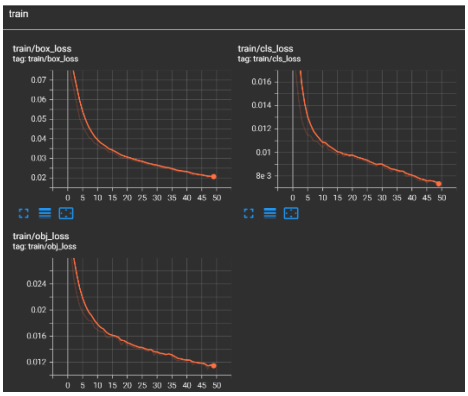
To launch TensorBoard, the following code snippet was run:

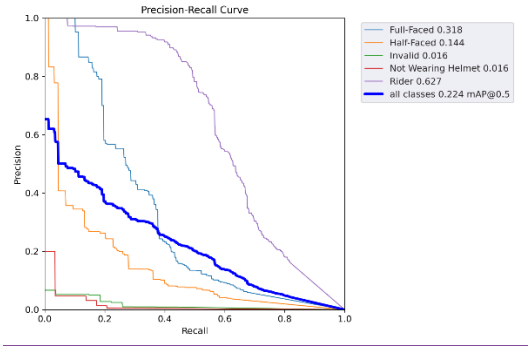
```
%load_ext tensorboard
%tensorboard --logdir runs
```

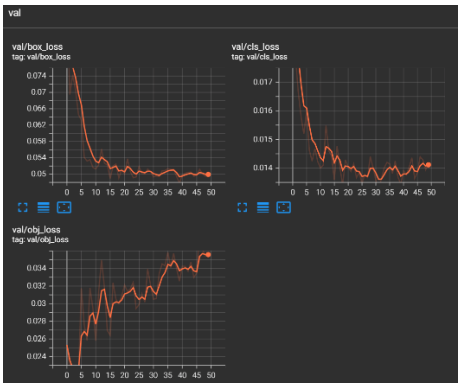
The graphical results of the YOLOv5s (**small**) model are seen in the figures below.

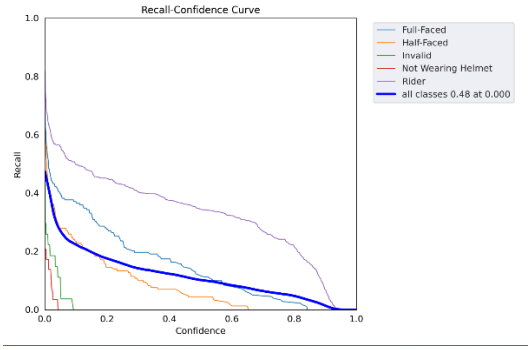


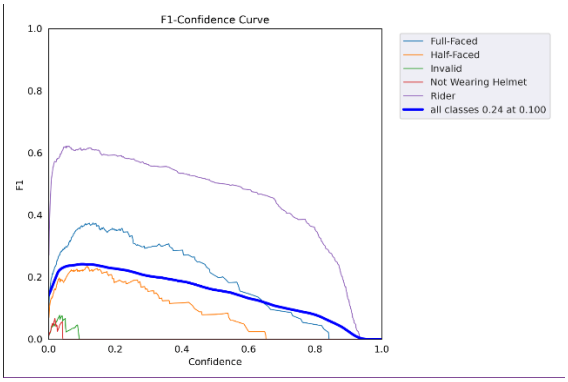


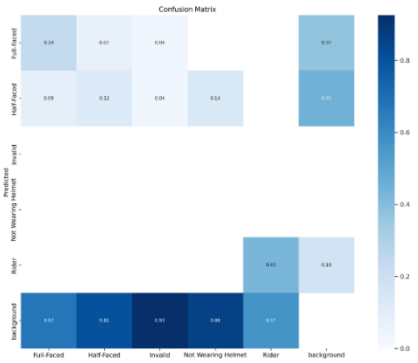


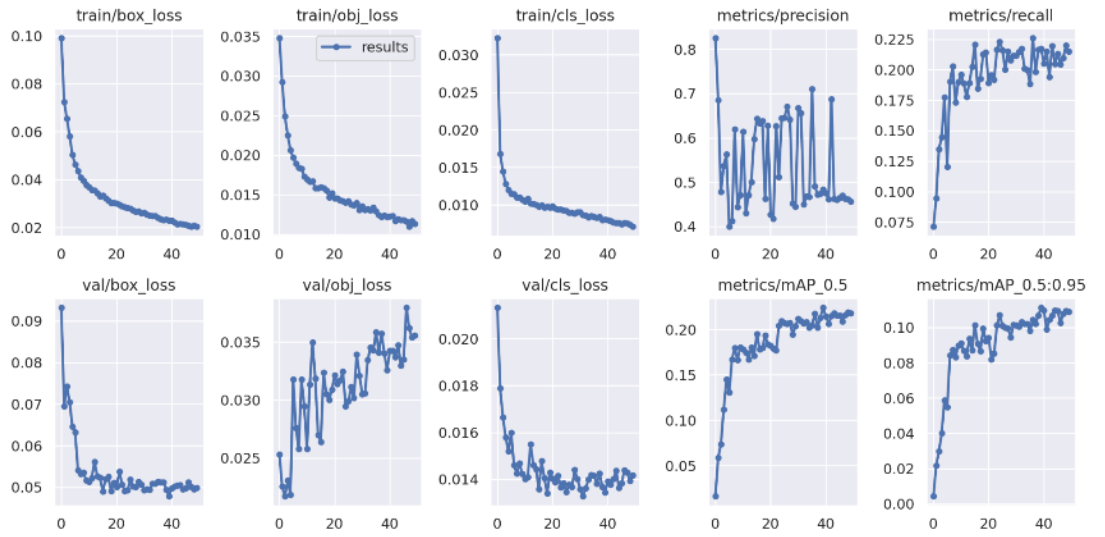




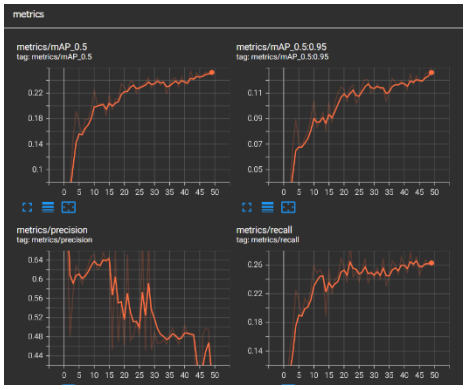


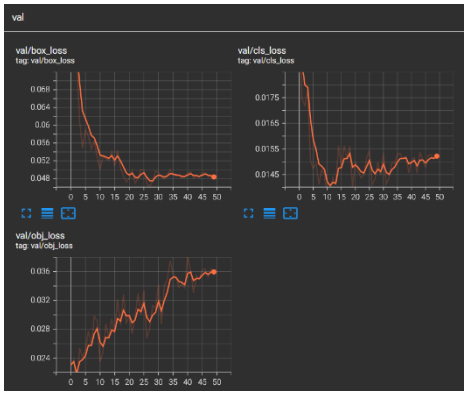


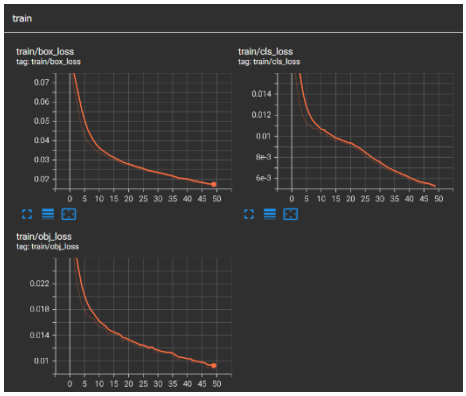


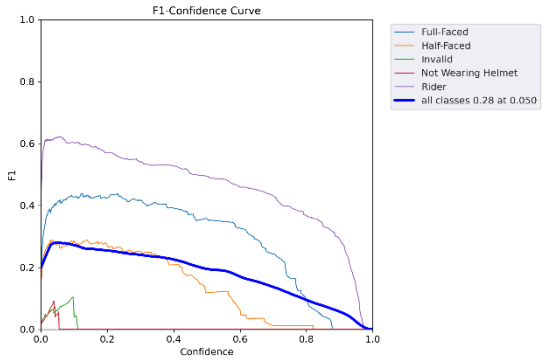


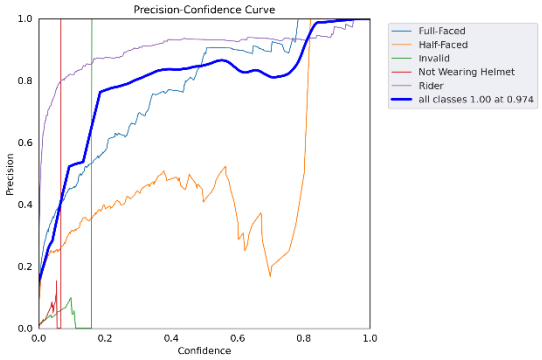
The results for the YOLOv5s (**medium**) model are seen below:

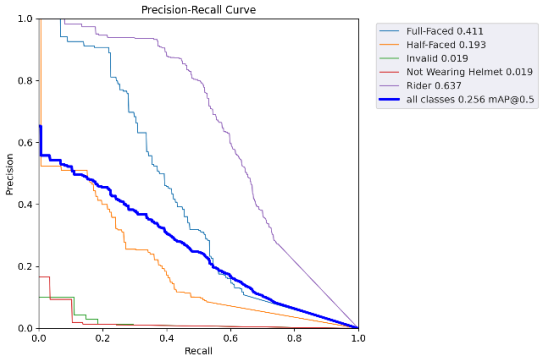


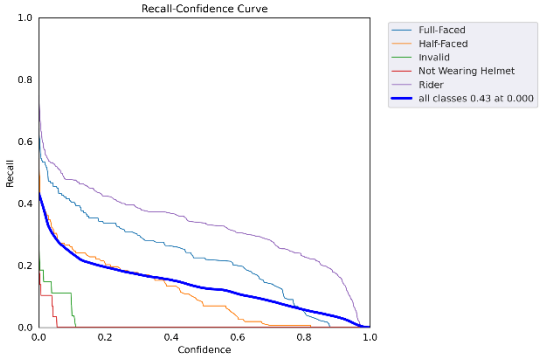


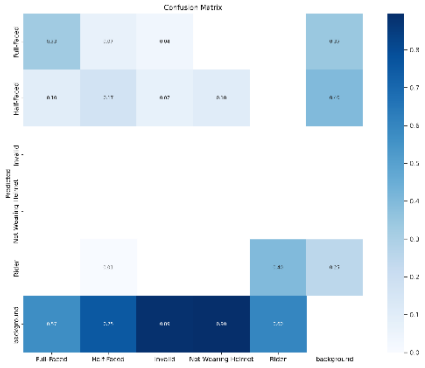


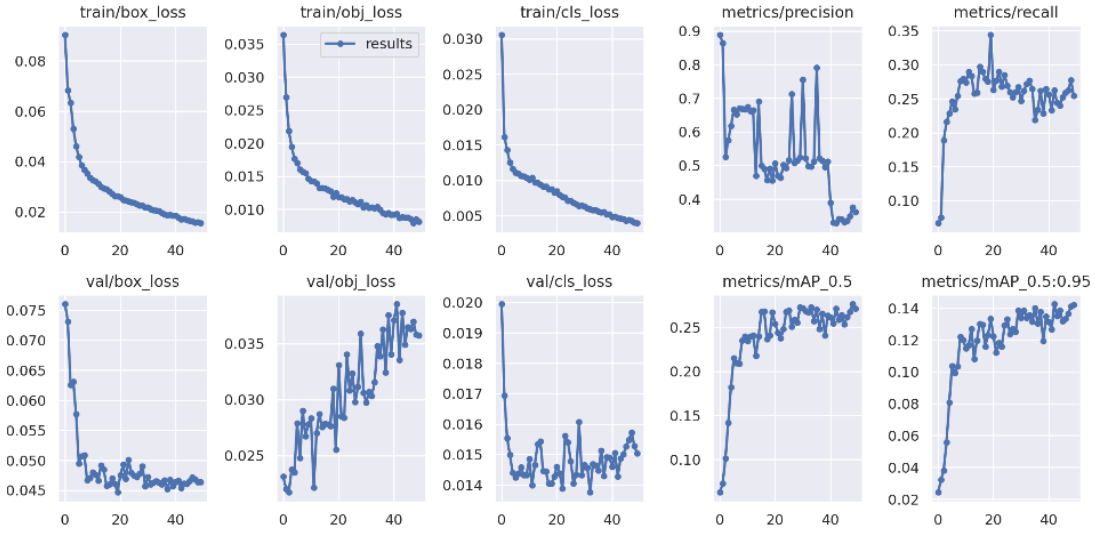



Precision-Confidence Curve
Precision
Confidence
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 1.00 at 0.974

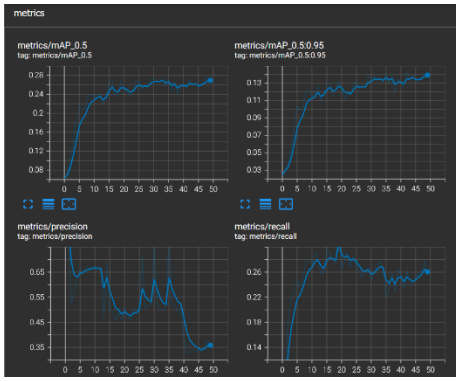

Precision-Recall Curve
Precision
Recall
Full-Faced 0.411
Half-Faced 0.193
Invalid 0.019
Not Wearing Helmet 0.019
Rider 0.637
all classes 0.256 mAP@0.5

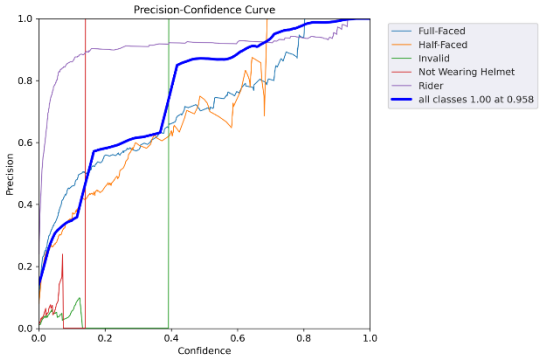

Recall-Confidence Curve
Recall
Confidence
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 0.43 at 0.000

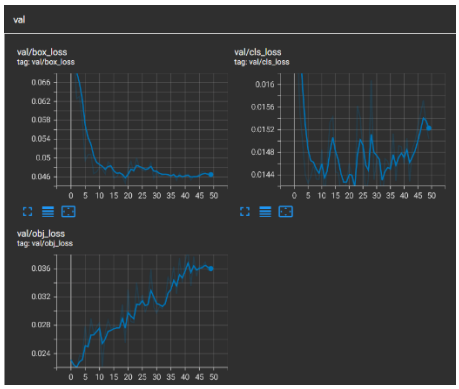

Confusion Matrix

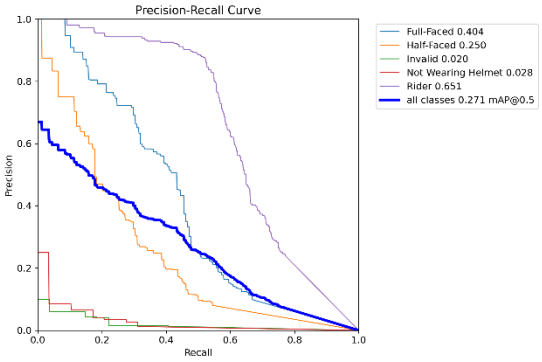

train/box_loss
train/obj_loss
results
train/cls_loss
metrics/precision
metrics/recall
val/box_loss
val/obj_loss
val/cls_loss
metrics/mAP_0.5
metrics/mAP_0.5:0.95

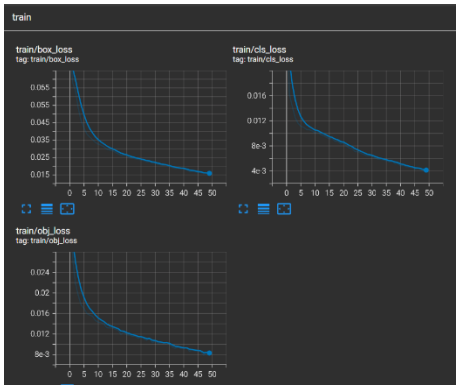
Finally, the results for the YOLOv5s (**large**) model are in the following figures:

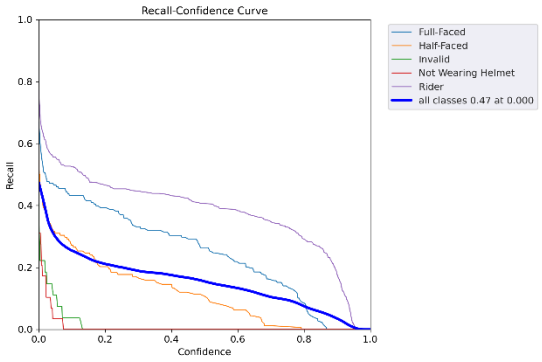


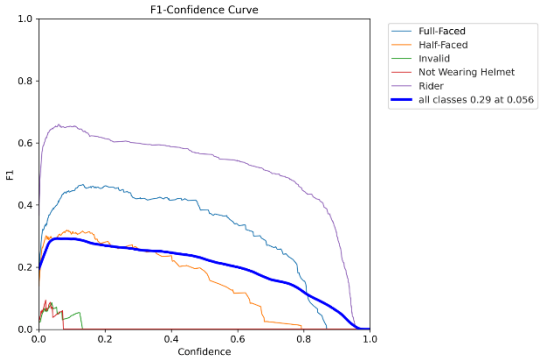


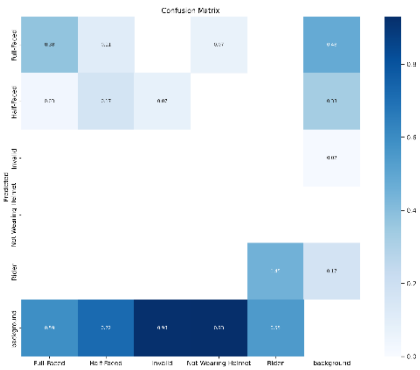


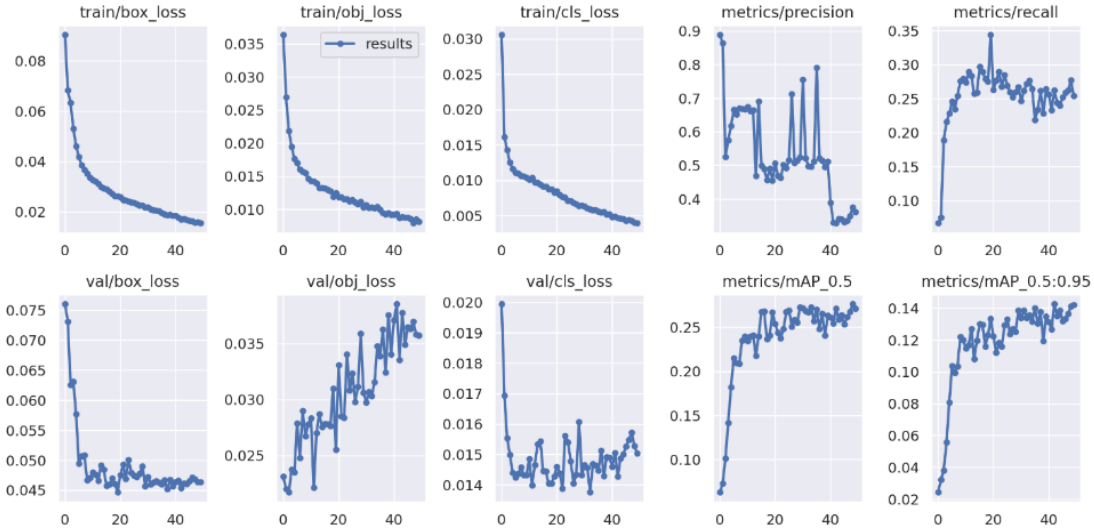










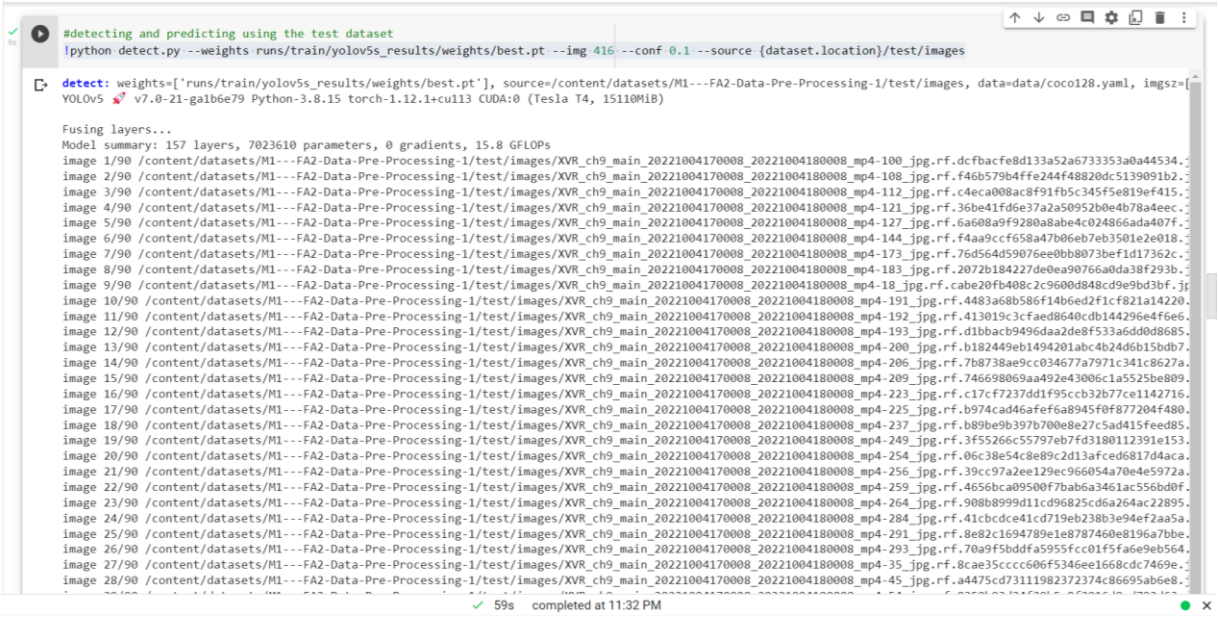


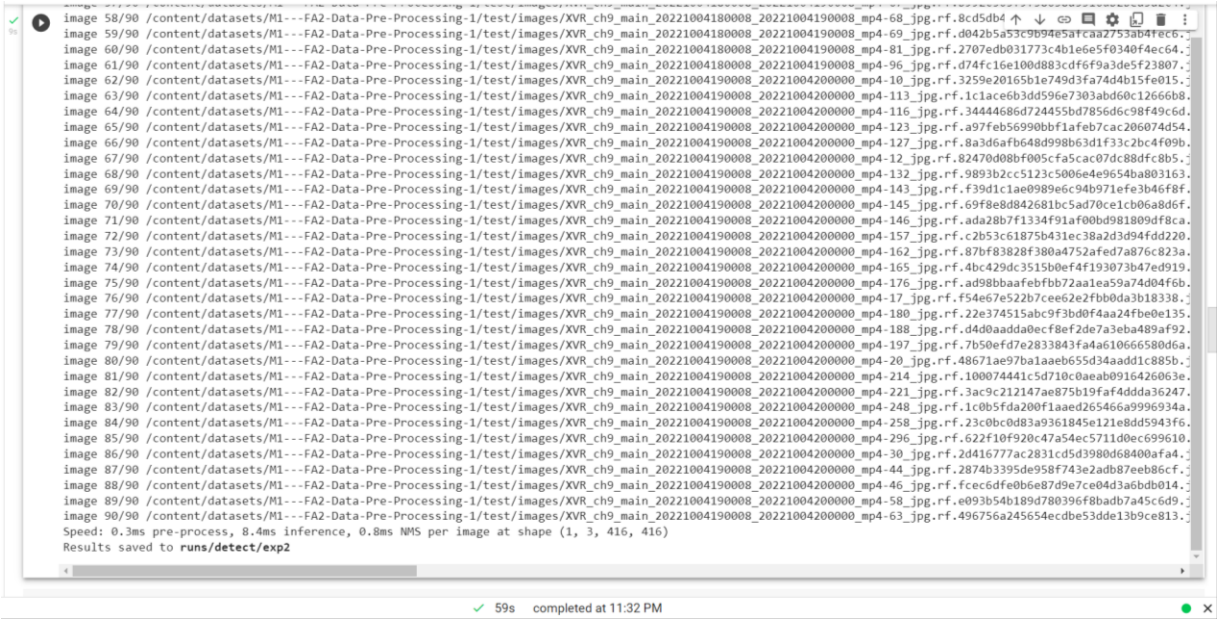
IV. Testing the models (small, medium, and large)

After training the model using the training dataset, the testing dataset, which consists of 4,500 frames from traffic surveillance footage, will now be used to test the model. The purpose of this dataset is to measure the performance metrics and detection capabilities of the model. To fulfill this purpose, *detect.py* from the Ultralytics repository was utilized to detect objects and predict where the bounding boxes from the test images are to be placed.

```
!python detect.py --weights runs/train/yolov5s_results/weights/best.pt --
img 416 --conf 0.1 --source {dataset.location}/test/images
```

Output:





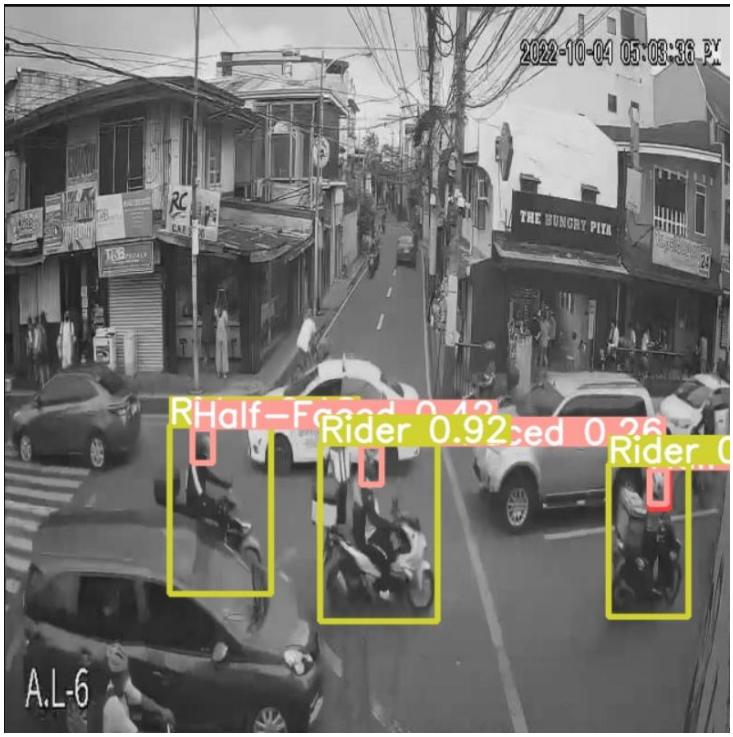
To display all test images with the bounding boxes placed on their predictions, the following libraries were imported:

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

Then, to display the inferenced test images on the next cell by retrieving the images from the specified directory, the code snippet below was run.

```
for imageName in glob.glob('/content/yolov5/runs/detect/exp2/*.jpg'):
#assuming JPG
    display(Image(filename=imageName))
    print("\n")
```

The figures below are a few examples of the inferenced (with bounding boxes) test images.




2022-10-04 05:03:20 PM
THE HUNGRY PITA
Half-Faced 0.32
Half-
Rider 0.94
Rider 0.
A.L-6


2022-10-04 05:03:18 PM
Rider 0.21
Rider 0.91

Next, the following code was run to produce the table output in the figure below. The results will be saved as a file named “best.pt.” The code snippet will produce a summary of the model’s values through a table, as its classes are divided among Full-Faced, Half-Faced, Invalid, Not Wearing Helmet, and Rider and classified according to the number of images, instances, precision (P), recall (R), mAP50, and mAP50-95.

```
!python val.py --weights './runs/train/yolov5s_results/weights/best.pt' --
data {dataset.location}/data.yaml --img 416
```

Output for the small-scaled model:

```
val: data=/content/datasets/M1---FA2-Data-Pre-Processing-1/data.yaml, weights=['./runs/train/yolov5s_results/weights/best.pt'], batch_size=32, imgsz=416,
YOLOv5 🚀 v7.0-21-ga1b6e79 Python-3.8.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
Model summary: 157 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels.cache... 180 images, 17 backgrounds, 0 corrupt: 100% 180/180 [00:00<?, ?it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:04<00:00,  1.33it/s]
                   all        180        802      0.485      0.218      0.226      0.111
            Full-Faced        180        178      0.394      0.369      0.325      0.105
            Half-Faced        180        158      0.229      0.228      0.144     0.0483
               Invalid        180         27          0          0     0.0161    0.00568
    Not Wearing Helmet        180         29          1          0     0.0157    0.00875
                 Rider        180        410      0.801      0.495      0.626      0.389
Speed: 0.3ms pre-process, 3.4ms inference, 2.3ms NMS per image at shape (32, 3, 416, 416)
Results saved to runs/val/exp3
```

Output for the medium-scaled model:

```
val: data=/content/datasets/M1---FA2-Data-Pre-Processing-1/data.yaml, weights=['./runs/train/yolov5m_results/weights/best.pt'], batch_size=32, i
YOLOv5 🚀 v7.0-21-ga1b6e79 Python-3.8.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
Model summary: 212 layers, 20869098 parameters, 0 gradients, 47.9 GFLOPs
val: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels.cache... 180 images, 17 backgrounds, 0 corrupt: 100% 180/180 [00:00<
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:05<00:00,  1.19it/s]
                   all        180        802      0.303      0.264      0.256       0.13
            Full-Faced        180        178      0.395      0.427      0.409      0.158
            Half-Faced        180        158      0.263      0.272      0.194     0.0732
               Invalid        180         27     0.0592      0.111     0.0191      0.01
    Not Wearing Helmet        180         29          0          0       0.02    0.00879
                 Rider        180        410      0.797      0.507      0.637      0.398
Speed: 0.6ms pre-process, 6.9ms inference, 2.2ms NMS per image at shape (32, 3, 416, 416)
Results saved to runs/val/exp2
```

Output for the large-scaled model:

```
val: data=/content/datasets/M1---FA2-Data-Pre-Processing-1/data.yaml, weights=['./runs/train/yolov5l_results2/weights/best.pt'], batch_size=32,
YOLOv5 🚀 v7.0-21-ga1b6e79 Python-3.8.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
Model summary: 267 layers, 46129818 parameters, 0 gradients, 107.7 GFLOPs
val: Scanning /content/datasets/M1---FA2-Data-Pre-Processing-1/valid/labels.cache... 180 images, 17 backgrounds, 0 corrupt: 100% 180/180 [00:00<
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:05<00:00,  1.03it/s]
                   all        180        802      0.328      0.262      0.268      0.142
            Full-Faced        180        178      0.427      0.447       0.39       0.16
            Half-Faced        180        158      0.333      0.297       0.25      0.108
               Invalid        180         27     0.0311      0.037       0.02    0.00737
    Not Wearing Helmet        180         29          0          0      0.028    0.00884
                 Rider        180        410      0.849      0.527      0.651      0.428
Speed: 0.5ms pre-process, 11.6ms inference, 2.3ms NMS per image at shape (32, 3, 416, 416)
Results saved to runs/val/exp
```

V. Discussion of results for each model

As demonstrated above, each model was tested and validated using the test dataset, through *detect.py*, and *val.py*. The small-scaled model has resulted in 48.5% precision rate, 21.8% recall rate, and a 22.6% mean average precision. The medium-scaled had a 30.3% precision rate, 26.4% recall rate, and a 25.6% mean average precision. Lastly, the large-scaled model had a 32.8% precision rate, 26.2% recall rate, and a 26.8% mean average precision.

Models	Precision	Recall	mAP
Small	48.5%	21.8%	22.6%
Medium	30.3%	26.4%	25.6%
Large	32.8%	26.2%	26.8%

As you can see, the small-scaled model scored better in precision but scored the lowest in the mean average precision rate. Note that mAP captures the tradeoff between precision and recall and maximizes the impact of both. Therefore, even with lower precision than the small-scaled model, the large-scaled model performed relatively better than all models as it has the highest mAP.

Furthermore, our trained model was able to detect motorcycle riders fairly, but it does not do a great job of classifying if they are wearing a proper helmet or not, or no helmet at all. As we can see from these graphs, it does not do well on both the training and testing datasets. This could mean that the model is underfitted due to its poor performance on the training data and overfitted because it does not perform well on the evaluation data.

In the future, we recommend increasing the amount of training data examples, enhancing the video quality, and using fewer features for the model.

VI. Testing on a video from the previous assessment

Once the individual video frames (images) have been trained and tested, each video recording will now go through the same process with the final objective of validating the model's performance in real world scenarios (traffic surveillance). To do this, first, we need to install *opencv* to perform image processing on the video.

```
#install opencv-python
%pip install opencv-python
```

Then, we upload a specific video for testing. For this documentation, we will just be using one video.

```
#upload video file
from google.colab import files
video = files.upload()
```

```
Browse... XVR_ch9_main_20221004190008_20221004200000.mp4
XVR_ch9_main_20221004190008_20221004200000.mp4(video/mp4) - 77698947 bytes, last modified: n/a - 100% done
Saving XVR_ch9_main_20221004190008_20221004200000.mp4 to XVR_ch9_main_20221004190008_20221004200000.mp4
```

After uploading the video, we will then use the video to validate the model's performance.

```
!python detect.py --weights runs/train/yolov5m_results/weights/best.pt --
img 416 --conf 0.1 --source '/content/yolov5/
XVR_ch9_main_20221004190008_20221004200000.mp4'
```

```
detect: weights=['runs/train/yolov5m_results/weights/best.pt'], source=/content/yolov5/*.mp4, data=data/coco128.yaml, imgsz=[416, 416], conf_t
YOLOv5 🚀 v7.0-21-ga1b6e79 Python-3.8.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
Model summary: 212 layers, 20869098 parameters, 0 gradients, 47.9 GFLOPs
video 1/1 (1/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 14.7ms
video 1/1 (2/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 14.1ms
video 1/1 (3/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 14.0ms
video 1/1 (4/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 14.0ms
video 1/1 (5/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 14.2ms
video 1/1 (6/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 18.3ms
video 1/1 (7/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 15.4ms
video 1/1 (8/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 13.4ms
video 1/1 (9/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 13.3ms
video 1/1 (10/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 13.4ms
video 1/1 (11/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 13.3ms
video 1/1 (12/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 12.5ms
video 1/1 (13/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 13.0ms
video 1/1 (14/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 11.9ms
video 1/1 (15/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 14.3ms
video 1/1 (16/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 13.1ms
video 1/1 (17/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 15.2ms
video 1/1 (18/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 1 Rider, 12.3ms
```

```
video 1/1 (4480/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.4ms
video 1/1 (4481/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.5ms
video 1/1 (4482/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 14.8ms
video 1/1 (4483/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.9ms
video 1/1 (4484/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.9ms
video 1/1 (4485/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 19.3ms
video 1/1 (4486/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 11.7ms
video 1/1 (4487/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.9ms
video 1/1 (4488/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.4ms
video 1/1 (4489/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 11.7ms
video 1/1 (4490/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.7ms
video 1/1 (4491/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.9ms
video 1/1 (4492/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 11.8ms
video 1/1 (4493/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.4ms
video 1/1 (4494/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.8ms
video 1/1 (4495/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 10.8ms
video 1/1 (4496/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 11.4ms
video 1/1 (4497/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 12.3ms
video 1/1 (4498/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 12.1ms
video 1/1 (4499/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 12.1ms
video 1/1 (4500/4500) /content/yolov5/XVR_ch9_main_20221004190008_20221004200000.mp4: 256x416 (no detections), 11.7ms
Speed: 0.3ms pre-process, 12.0ms inference, 0.4ms NMS per image at shape (1, 3, 416, 416)
Results saved to runs/detect/exp5
```

Then, we download the results using the code snippet below:

```
files.download('/content/yolov5/runs/detect/exp5/XVR_ch9_main_20221004190008_20221004200000.mp4') #download the video after model detection
```

We repeated these steps until all models had the chance to be validated. If you would like to see all three video outputs for each model, the link is here.

References

Roboflow. (2020, June 15). *How to Train YOLO v5 on a Custom Dataset* [Video]. YouTube. https://www.youtube.com/watch?v=MdF6x6ZmLAY

Solawetz, J. (2021, December 14). *How to Train A Custom Object Detection Model with YOLO v5*. Medium. https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-yolo-v5-917e9ce13208