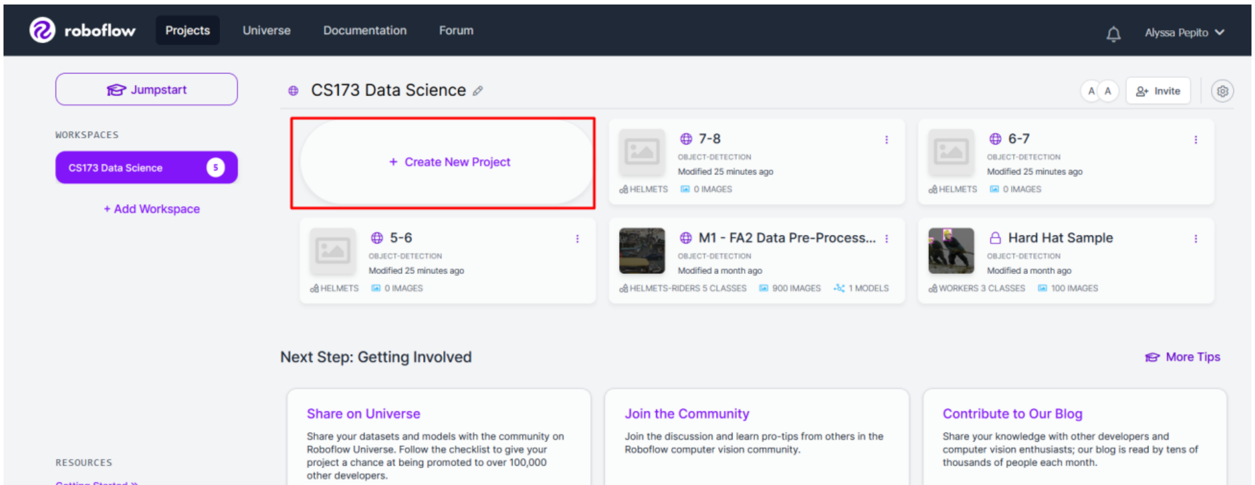
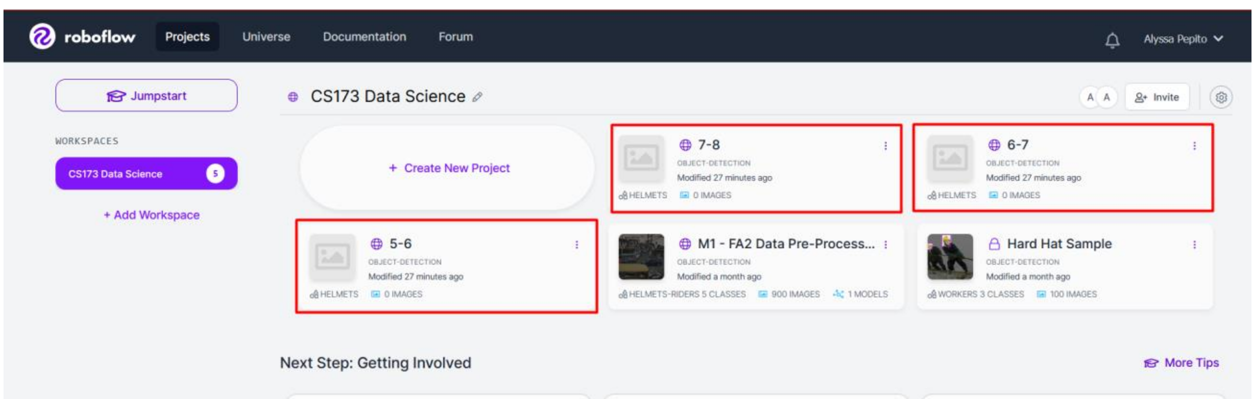
M2 - FA2 YOLOv7

I. Creating Three Separate Time Scenarios

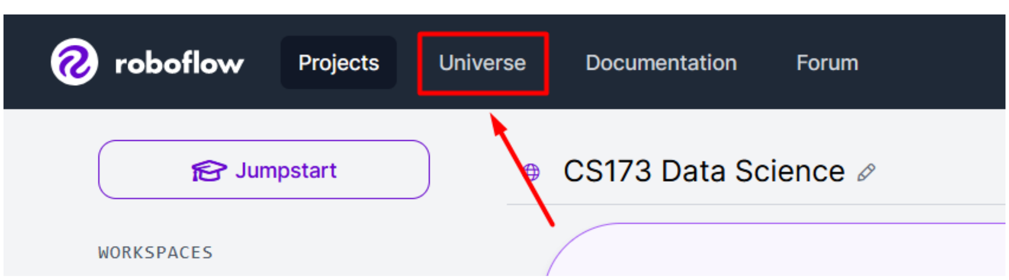


In Roboflow, we must first create three new projects to place our three-time scenarios. Click on "+ Create New Project".

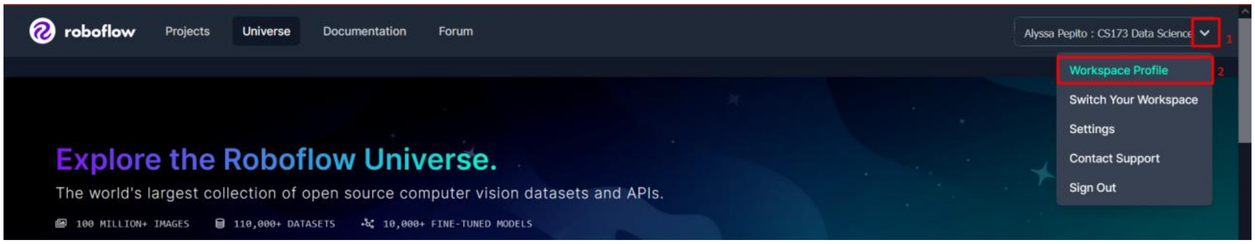


After filling up the details for the new project, repeat this step until you get three projects named specifically for the three-time scenarios (5-6, 6-7, and 7-8).

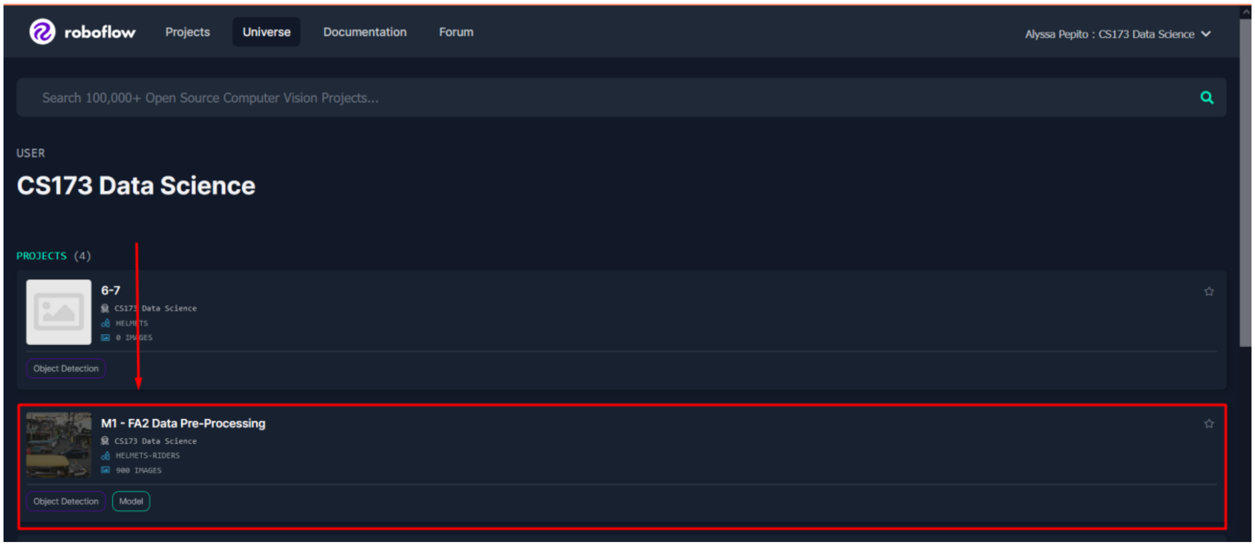
II. Cloning the Images



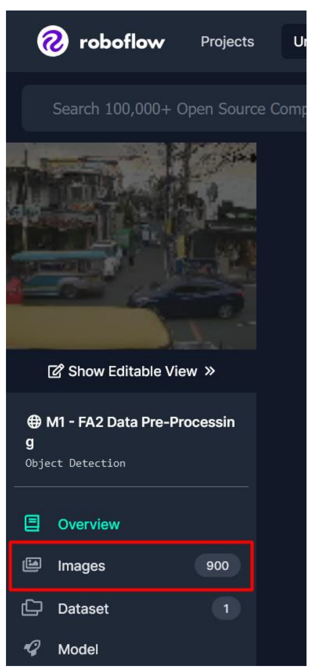
Next, on the top left corner of Roboflow, click on Universe.



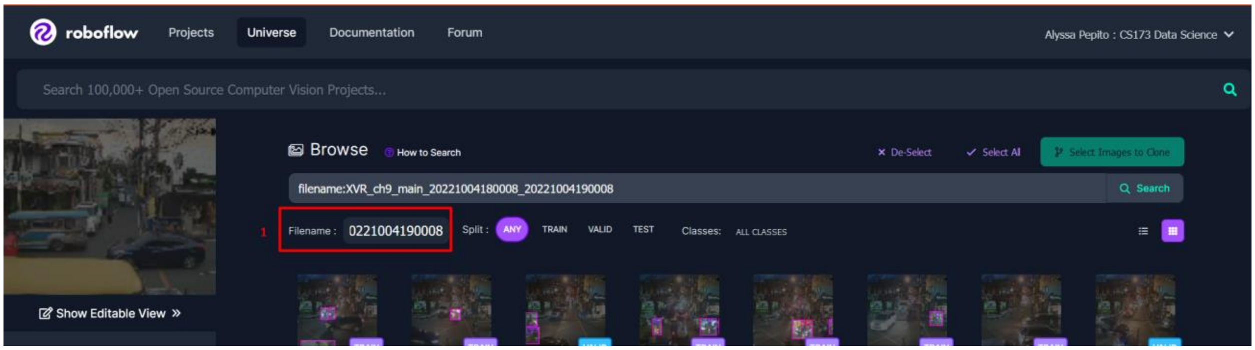
Roboflow Universe will open in a new tab. On the top right corner, click on the dropdown button and then click on "Workspace Profile".



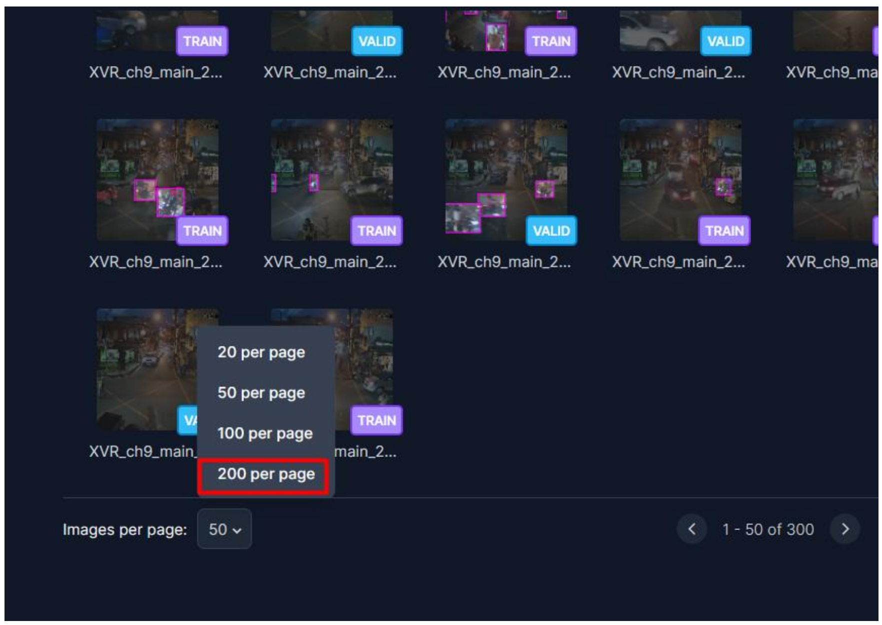
We select the old dataset we used from the previous activity "M1-FA2".



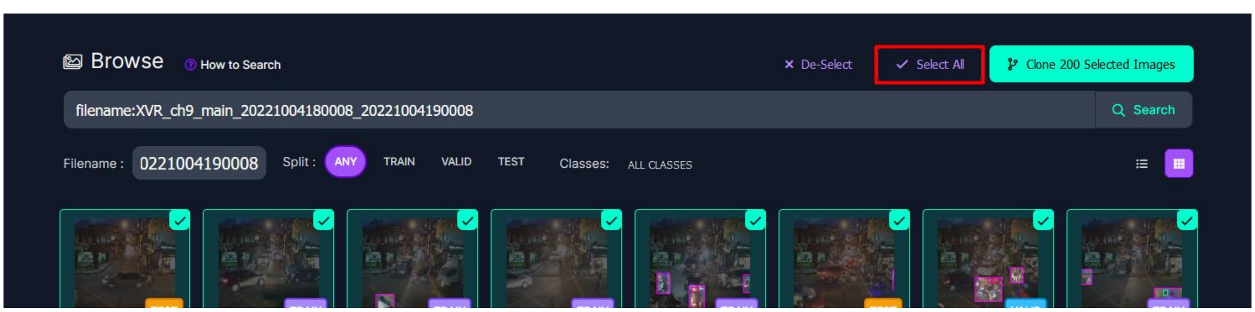
After the dataset has been loaded, we go to the left corner and click on "Images".



Next, we search for files that has the filename we want. As demonstrated above, we searched for "XVR_ch9_main_20221004190008_20221004200000" which are images from the time scenario 7-8 pm.

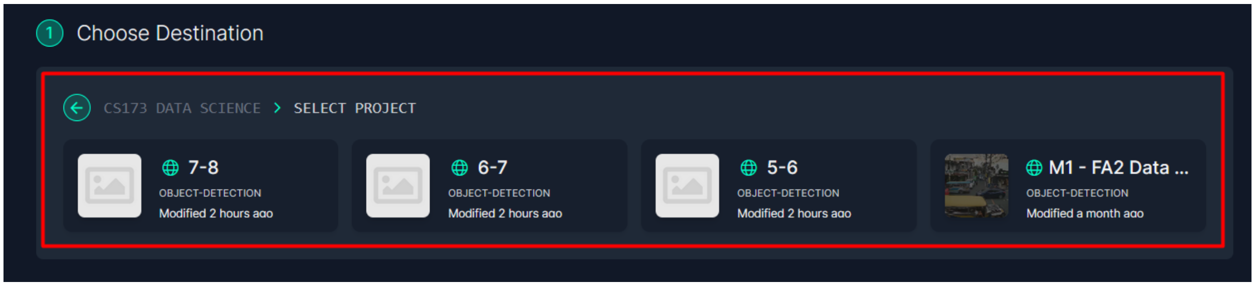


After the images have been loaded, we scrolled down to notice that there are a total of 300 images for that specific time scenario. To see more than 50 images, we clicked on the dropdown button next to the phrase "Images per page" and selected "200 per page".

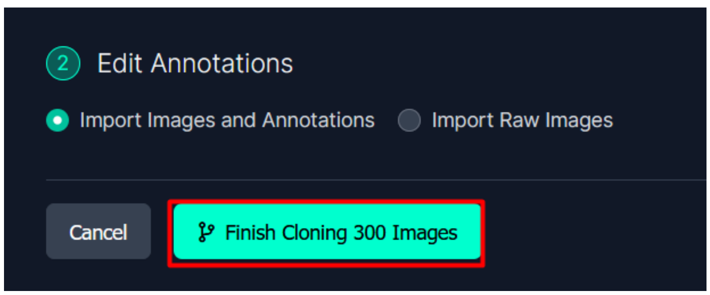


Once that there are 200 images in the first page, scroll up again and click on "Select all". Go to the next page and click on "Select all" again until all 300 images are selected.

Next, click on “Clone 300 Selected Images” to start the cloning process.

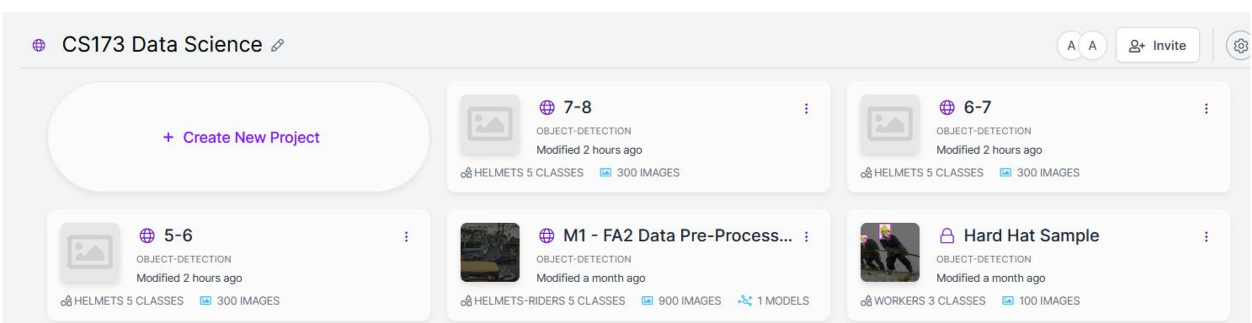


Scroll down to select which project to place the cloned images. In this case, it would be the 7-8 named project.

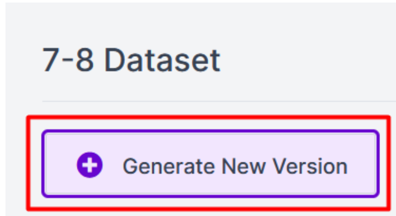


After selecting the project, we continue with the default option of “Import Images and Annotations” and click on “Finish Cloning 300 Images”. Repeat these steps until you have all images for every time scenario.

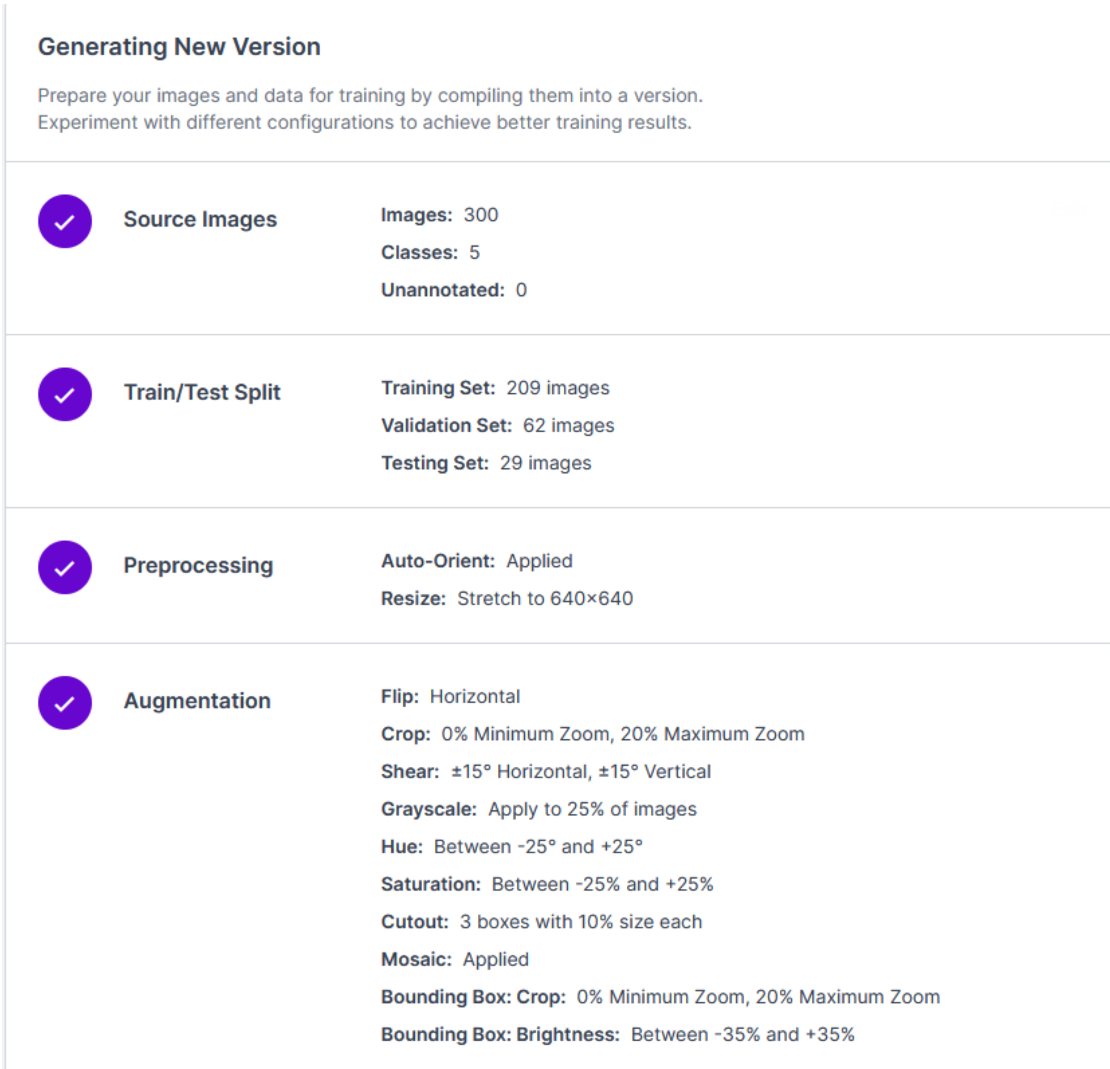
III. Generating New Versions of the Projects



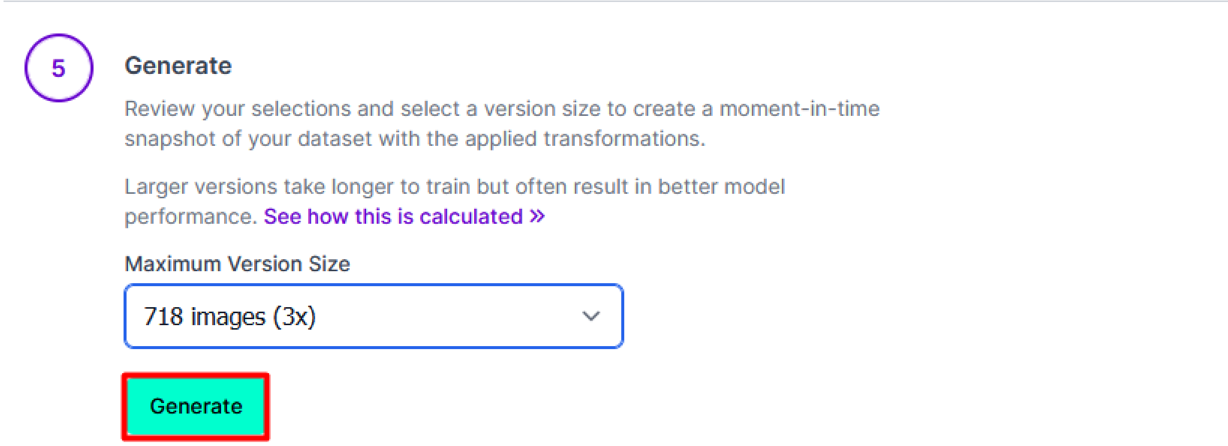
We go back to Roboflow Projects to see all time scenario projects have all their images in place. We will then generate new versions for them. To start, we click on any one of them.



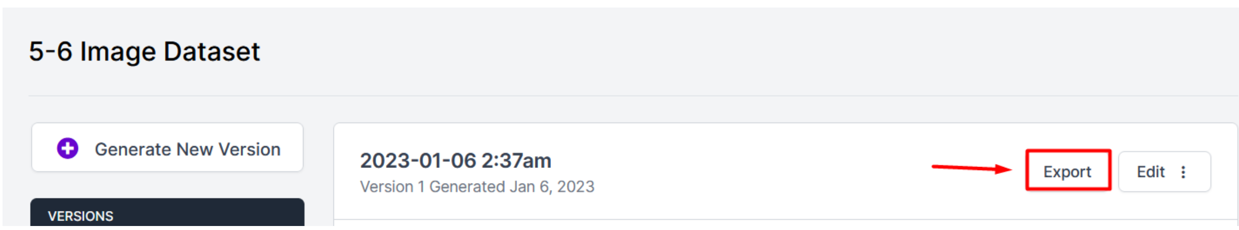
Inside the project, click on "+ Generate New Version" to get started.



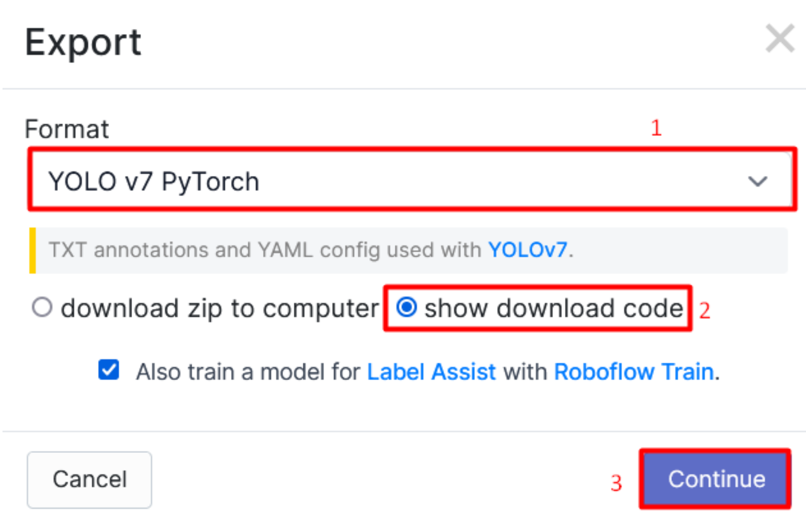
As seen above, we implemented auto-orient and resize for our preprocessing. For our augmentation, we applied flip, crop, shear, grayscale, hue, saturation, cutout, mosaic, bounding box: crop, and bounding box: brightness. Take note that one can experiment with their preprocessing and augmentation options. It is not necessary to follow what we have done here.



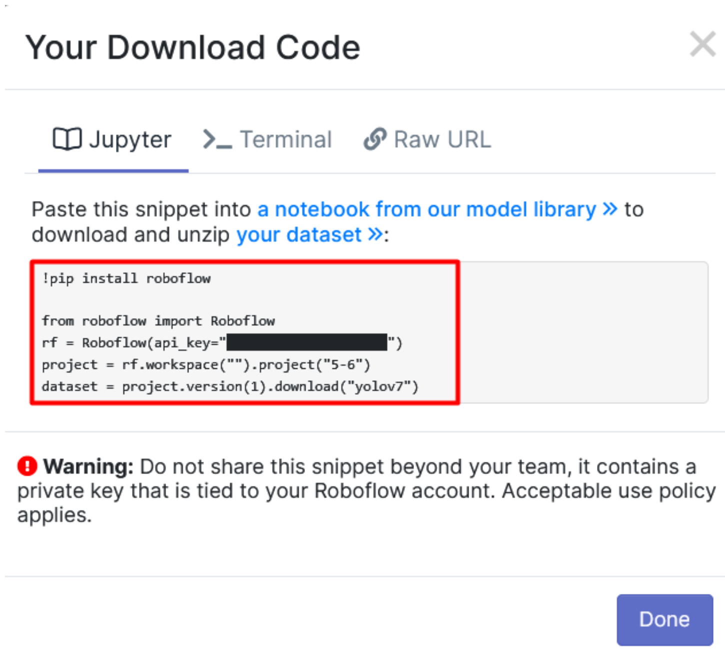
Next, we continue with the default option of 3x for version size and click on Generate.



After generating, the dataset is ready to be used for training a model. To do that, we must click on “Export”.



As seen above, make sure that the format selected should be YOLO v7 PyTorch. Select show download code and click on continue.



The download code will then be generated, make sure to save this snippet for later. Repeat these steps for every time scenario project. By the end of this, you would have a total of three code snippets, each one from the three-time scenarios.

IV. Libraries and Dependencies Installation

Unlike the assessments we accomplished previously, this time we will be creating a YOLOv7 model. First, we cloned and downloaded the YOLOv7 repository and installed its dependencies. As shown in the code sample below, we also installed the Roboflow-specific library because the dataset for this model will originate from Roboflow.

```
# Download YOLOv7 repository and install requirements
!git clone https://github.com/WongKinYiu/yolov7
%cd yolov7
!pip install -r requirements.txt
%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.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

Once the setup above is completed, we imported Roboflow by applying the following code snippet:

```
from roboflow import Roboflow #import roboflow for our datasets
rf = Roboflow(model_format="yolov7", notebook="ultralytics")
```

We then set up the environment.

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

Choose any of the code snippets we have [copied previously](#) from the three-time scenarios for now, as we must repeat this with the other time scenarios later. Then, paste the snippet below the environment set-up and run it. The Roboflow dataset will be downloading.

```
from roboflow import Roboflow
rf = Roboflow(api_key="RU6eGLCNP3Q4UwQqKWha")
project = rf.workspace("cs173-data-science").project("5-6")
dataset = project.version(1).download("yolov7")
```

After the Roboflow dataset has download, we will then download the required weights for our training.

```
# download COCO starting checkpoint
%cd /content/yolov7
!wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7_t
raining.pt
```

V. Training the YOLOv7 model

After installing all required files and dependencies, we used train.py to train the model. The following hyperparameters were chosen for this model:

1. Image size = 640 pixels

2. Batch size = 16

3. Epochs = 50

4. YOLOv7 Version = YOLOv7

The following code snippet was executed to apply the ideal hyperparameters specified above, producing the results from the associated figures:

```
!python train.py --img 640 --batch 16 --epochs 50 --
data {dataset.location}/data.yaml --weights 'yolov7_training.pt' --cache
```

```
!python train.py --img 640 --batch 16 --epochs 50 --data {dataset.location}/data.yaml --weights 'yolov7_training.pt' --cache
```

```
YOLOR 🚀 v0.1-121-g2fdc7f1 torch 1.13.0+cu116 CUDA:0 (Tesla T4, 15109.75MB)

Namespace(adam=False, artifact_alias='latest', batch_size=16, bbox_interval=-1, bucket='', cache_images=True, cfg='', data='/content/datasets/
tensorboard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
hyperparameters: lr0=0.01, lrf=0.1, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05,
wandb: Install Weights & Biases for YOLOR logging with 'pip install wandb' (recommended)
Overriding model.yaml nc=80 with nc=5

                 from  n    params  module                                  arguments
  0                -1  1       928  models.common.Conv                      [3, 32, 3, 1]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1      8320  models.common.Conv                      [128, 64, 1, 1]
  5                -2  1      8320  models.common.Conv                      [128, 64, 1, 1]
  6                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  7                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  8                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  9                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
 10  [-1, -3, -5, -6]  1         0  models.common.Concat                    [1]
 11                -1  1     66048  models.common.Conv                      [256, 256, 1, 1]
 12                -1  1         0  models.common.MP                        []
 13                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 14                -3  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
```

```
Transferred 557/566 items from yolov7_training.pt
Scaled weight_decay = 0.0005
Optimizer groups: 95 .bias, 95 conv.weight, 98 other
train: Scanning '/content/datasets/5-6-1/train/labels' images and labels... 615 found, 0 missing, 2 empty, 0 corrupted: 100% 615/615 [00:00<0
train: New cache created: /content/datasets/5-6-1/train/labels.cache
train: Caching images (0.8GB): 100% 615/615 [00:02<00:00, 238.19it/s]
val: Scanning '/content/datasets/5-6-1/valid/labels' images and labels... 62 found, 0 missing, 0 empty, 0 corrupted: 100% 62/62 [00:00<00:00,
val: New cache created: /content/datasets/5-6-1/valid/labels.cache
val: Caching images (0.1GB): 100% 62/62 [00:00<00:00, 176.90it/s]

autoanchor: Analyzing anchors... anchors/target = 4.86, Best Possible Recall (BPR) = 0.9990
Image sizes 640 train, 640 test
Using 2 dataloader workers
Logging results to runs/train/exp
Starting training for 50 epochs...

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      0/49        8G   0.08582    0.0261    0.0256    0.1375       156       640: 100% 39/39 [01:05<00:00,  1.67s/it]
               Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 2/2 [00:17<00:00,  8.62s/it]
                 all        62       436   0.00184   0.00622  0.000483   7.35e-05

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      1/49     10.8G   0.07599   0.02422   0.01792    0.1181       105       640: 100% 39/39 [00:32<00:00,  1.18it/s]
               Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.55it/s]
                 all        62       436     0.214    0.0653   0.00912   0.00169
```

```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     46/49     10.8G    0.0264   0.01843  0.004139   0.04897       159       640: 100% 39/39 [00:33<00:00,  1.18it/s]
               Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.55it/s]
                 all        62       436     0.556     0.355      0.31     0.149

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     47/49     10.8G    0.0264    0.0187  0.004057   0.04915        91       640: 100% 39/39 [00:34<00:00,  1.15it/s]
               Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.53it/s]
                 all        62       436     0.653     0.361     0.314     0.147

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     48/49     10.8G   0.02592   0.01752  0.004138   0.04758       149       640: 100% 39/39 [00:33<00:00,  1.18it/s]
               Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.54it/s]
                 all        62       436     0.717     0.335     0.314     0.153

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     49/49     10.8G   0.02603   0.01747  0.004216   0.04772        72       640: 100% 39/39 [00:32<00:00,  1.19it/s]
               Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 2/2 [00:02<00:00,  1.10s/it]
                 all        62       436     0.549      0.38     0.322     0.155
           Full-Faced        62       109     0.406     0.596     0.423     0.114
           Half-Faced        62        86     0.256     0.442     0.232    0.0865
              Invalid        62        22     0.328    0.0455    0.0486    0.0167
   Not Wearing Helmet        62        26         1         0    0.0527    0.0152
                Rider        62       193     0.754     0.819     0.855     0.543
50 epochs completed in 0.520 hours.

Optimizer stripped from runs/train/exp/weights/last.pt, 74.8MB
Optimizer stripped from runs/train/exp/weights/best.pt, 74.8MB
```

Since the above was resulted from the 5:00 – 6:00 pm time scenario, we would need to change this snippet to another one from the other time scenarios. Then, execute train.py with the same hyperparameters. The following are the results from the 6:00 – 7:00 pm time scenario.

```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     46/49     10.8G   0.02333   0.01155   0.00246   0.03734        39       640: 100% 41/41 [00:32<00:00,  1.26it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.91it/s]
                 all         56        158      0.767      0.323      0.346      0.147

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     47/49     10.8G   0.02352   0.01185  0.002305   0.03767        47       640: 100% 41/41 [00:32<00:00,  1.26it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.92it/s]
                 all         56        158      0.836      0.288      0.348       0.15

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     48/49     10.8G   0.02275   0.01127  0.002099   0.03613        25       640: 100% 41/41 [00:33<00:00,  1.22it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.96it/s]
                 all         56        158       0.81      0.323      0.364      0.159

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     49/49     10.8G   0.02313   0.01139  0.002152   0.03667        50       640: 100% 41/41 [00:32<00:00,  1.26it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.24it/s]
                 all         56        158      0.828      0.297      0.365      0.159
          Full-Faced         56         32      0.515      0.332        0.4       0.13
          Half-Faced         56         18        0.8      0.444      0.511      0.147
             Invalid         56          1          1          0     0.0138    0.00553
   Not Wearing Helmet         56          1          1          0      0.111     0.0553
               Rider         56        106      0.825      0.708      0.793      0.459
50 epochs completed in 0.512 hours.

Optimizer stripped from runs/train/exp/weights/last.pt, 74.8MB
Optimizer stripped from runs/train/exp/weights/best.pt, 74.8MB
```

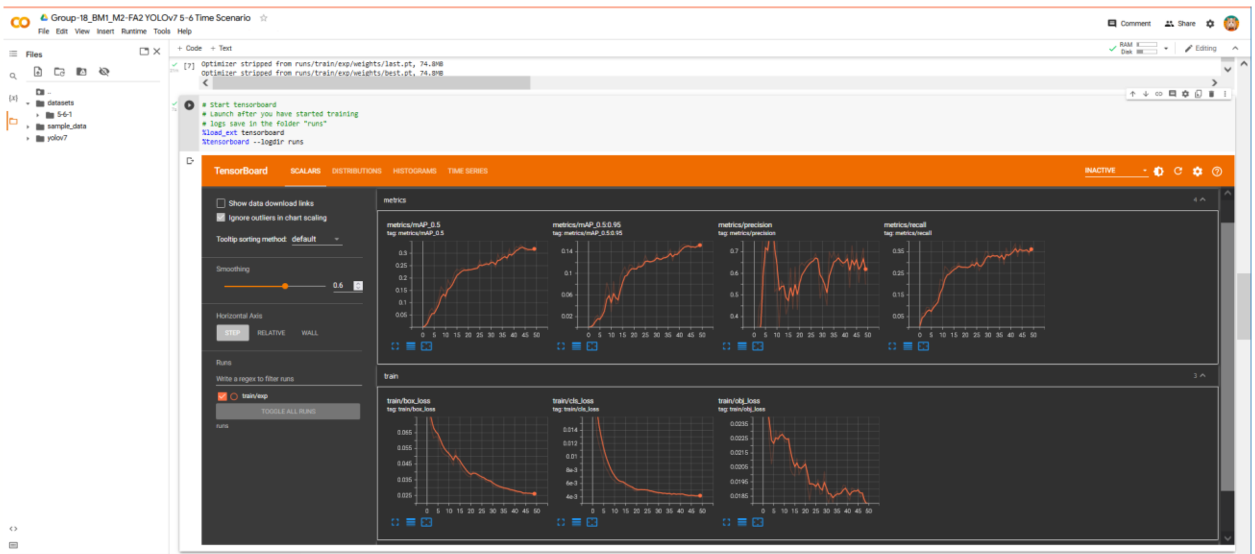
Lastly, these are the results from the 7:00 – 8:00 pm time scenario.

```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     43/49     10.9G   0.02788   0.01207  0.003985   0.04394        28       640: 100% 40/40 [00:33<00:00,  1.21it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.70it/s]
                 all         62        208       0.64      0.303      0.245       0.12

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     44/49     10.9G   0.02654   0.01217  0.003796   0.04251        20       640: 100% 40/40 [00:31<00:00,  1.25it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.71it/s]
                 all         62        208      0.665      0.277      0.248       0.12

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     45/49     10.9G   0.02685   0.01213  0.003628    0.0426        27       640: 100% 40/40 [00:32<00:00,  1.25it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.71it/s]
                 all         62        208       0.64      0.275      0.231      0.117

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     46/49     10.9G    0.0266    0.0119  0.003459   0.04196        22       640: 100% 40/40 [00:31<00:00,  1.25it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.73it/s]
                 all         62        208       0.64      0.321      0.253      0.121

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     47/49     10.9G   0.02702   0.01205  0.003789   0.04285        25       640: 100% 40/40 [00:32<00:00,  1.25it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.71it/s]
                 all         62        208      0.661      0.292      0.254      0.123

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     48/49     10.9G   0.02689   0.01148  0.003695   0.04206        51       640: 100% 40/40 [00:31<00:00,  1.26it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.71it/s]
                 all         62        208      0.663       0.29      0.254      0.121

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     49/49     10.9G   0.02567   0.01163  0.003542   0.04085         3       640: 100% 40/40 [00:31<00:00,  1.25it/s]
               Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.08it/s]
                 all         62        208      0.638      0.311      0.245      0.117
          Full-Faced         62         37      0.174      0.405      0.211     0.0476
          Half-Faced         62         54      0.338      0.368      0.232     0.0512
             Invalid         62          4          1          0    0.00203   0.000862
   Not Wearing Helmet         62          2          1          0          0          0
               Rider         62        111      0.681      0.784       0.78      0.485
50 epochs completed in 0.503 hours.

Optimizer stripped from runs/train/exp/weights/last.pt, 74.8MB
Optimizer stripped from runs/train/exp/weights/best.pt, 74.8MB
```

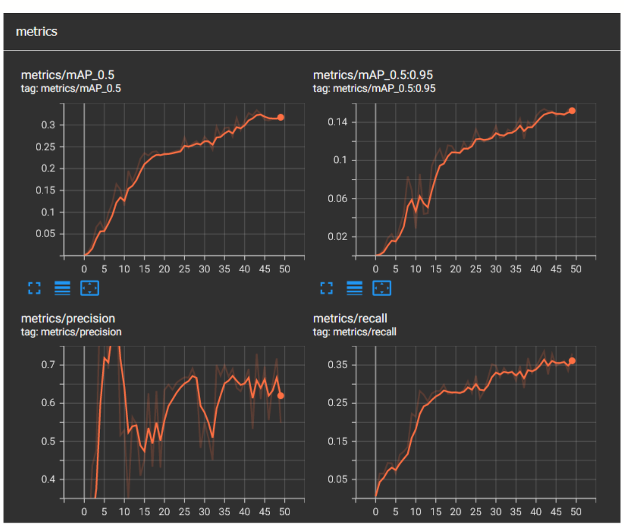
VI. TensorBoard

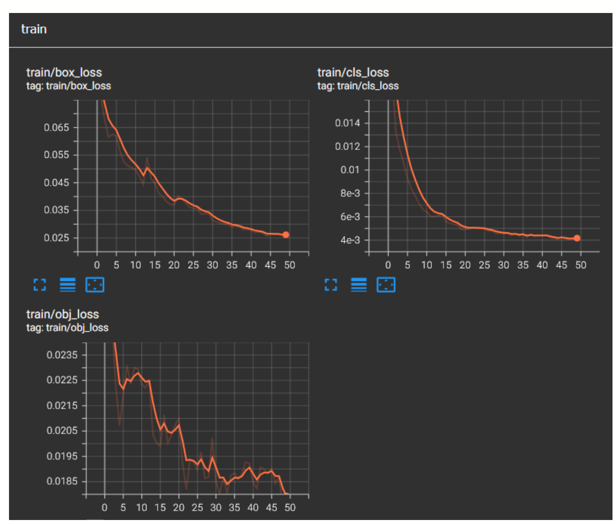
TensorBoard was once again employed to visualize the training process' outcomes using scalar measurements, pictures, graphs, and time series. Scalar measures like mAP (mean average precision), precision, and recall are just a few examples. One can view the model's performance through graphs that show the metrics, train, and the model's overall outcomes under. The time series displays each runtime that the model underwent during training. The TensorBoard was started by running the following line of code:

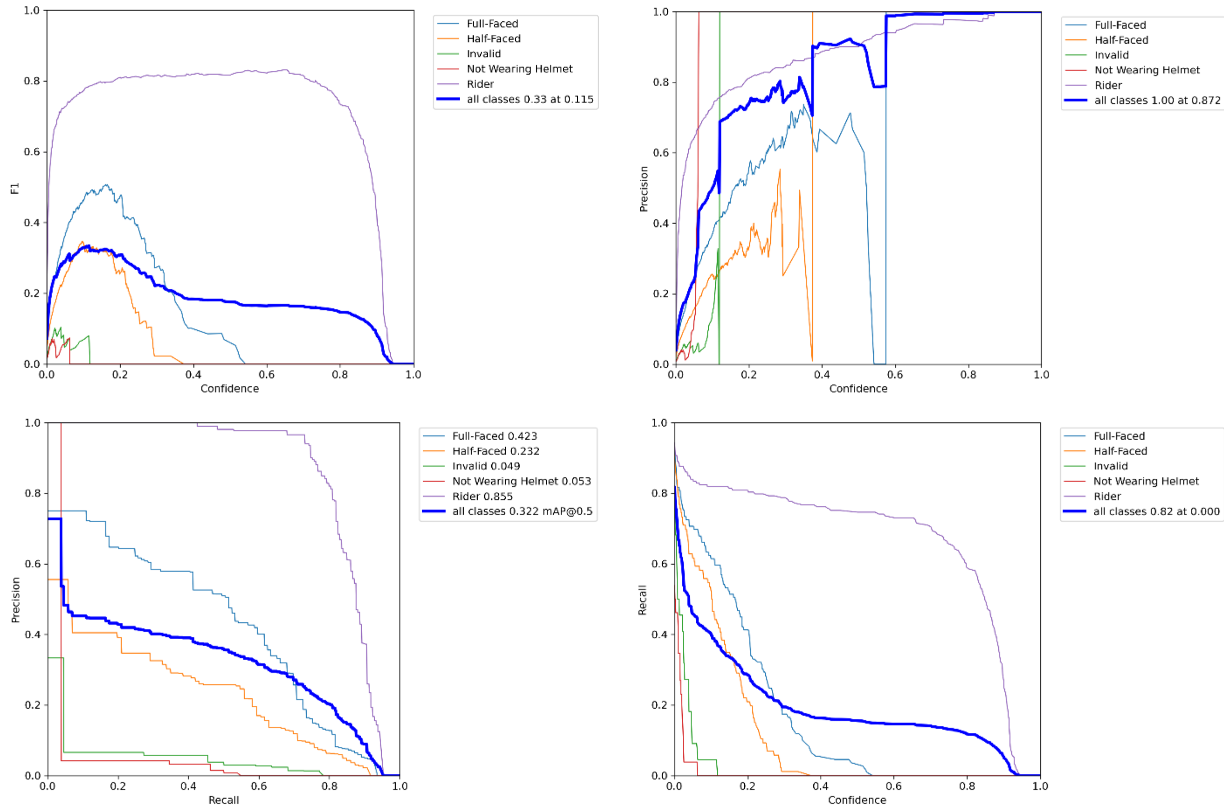
```
%load_ext tensorboard
%tensorboard --logdir runs
```

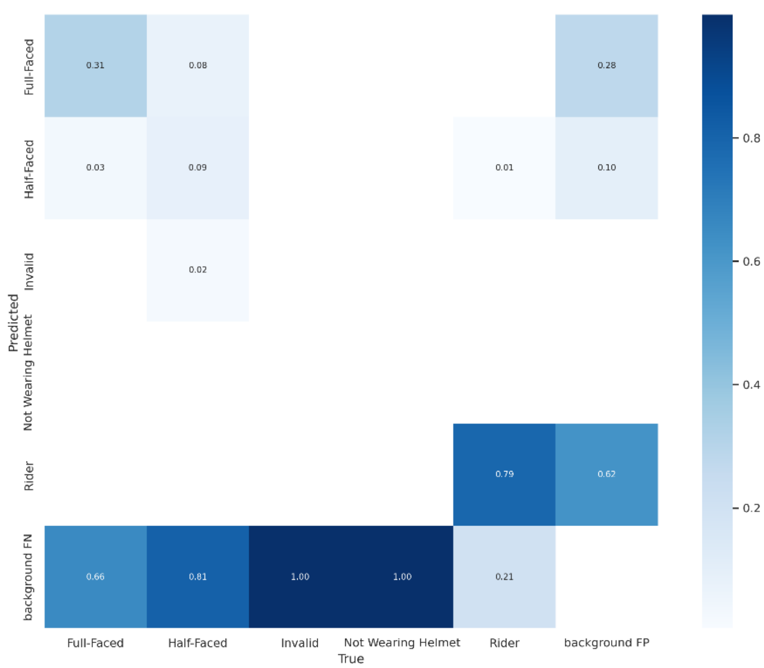


The graphical results for the 5:00 – 6:00 pm time scenario are seen in the figures below:

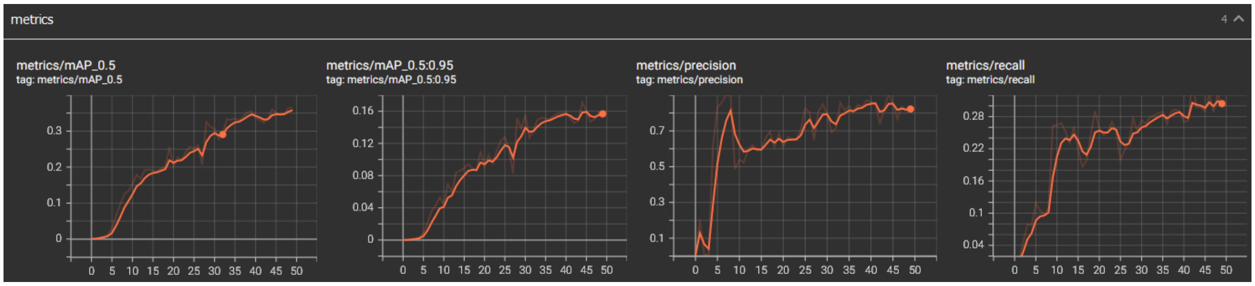


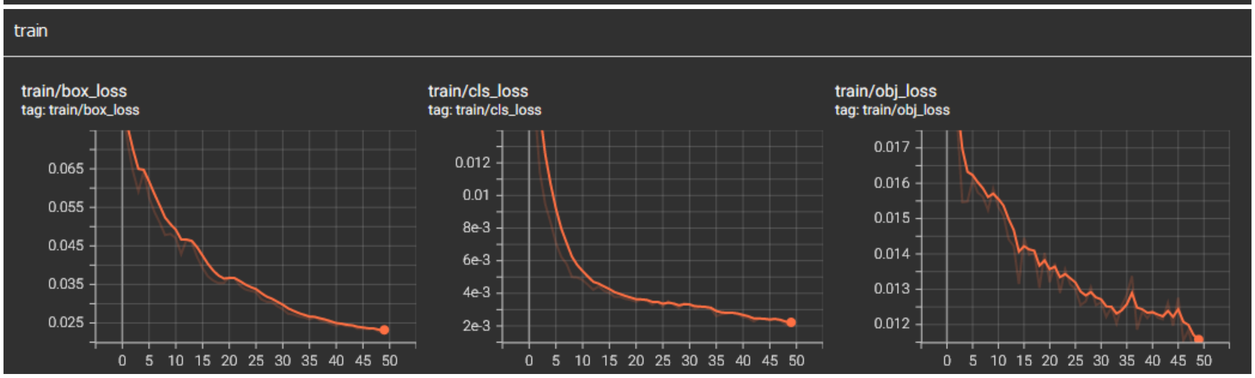


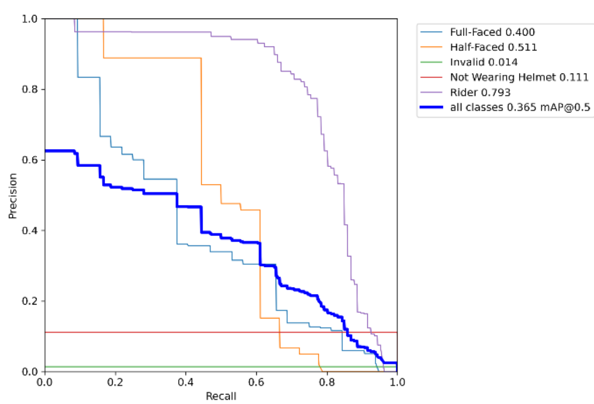


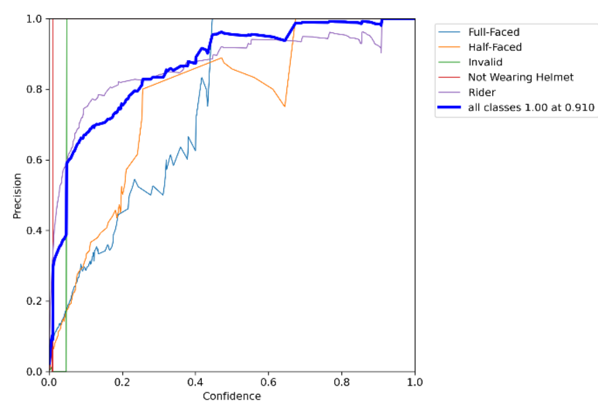


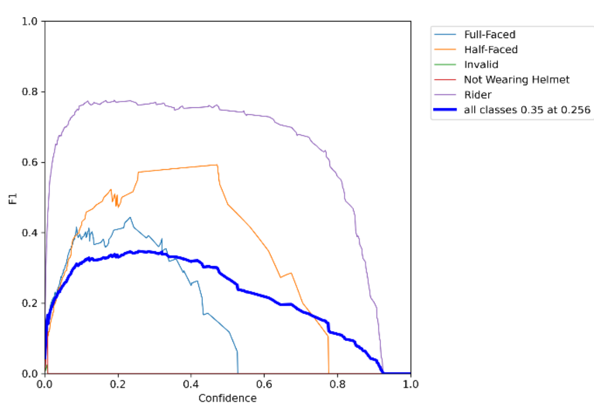
The results for the 6:00 – 7:00 pm time scenario are seen below:

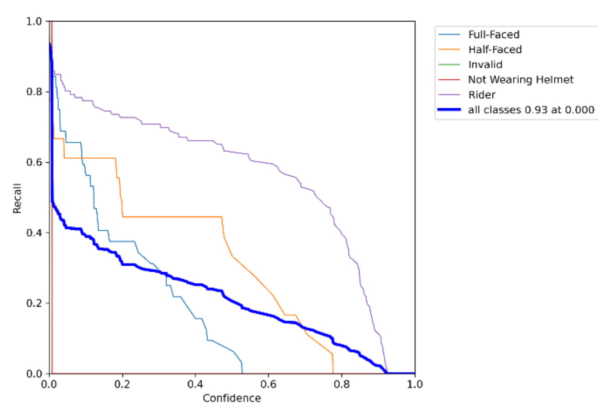

metrics
metrics/mAP_0.5
tag: metrics/mAP_0.5
metrics/mAP_0.5:0.95
tag: metrics/mAP_0.5:0.95
metrics/precision
tag: metrics/precision
metrics/recall
tag: metrics/recall

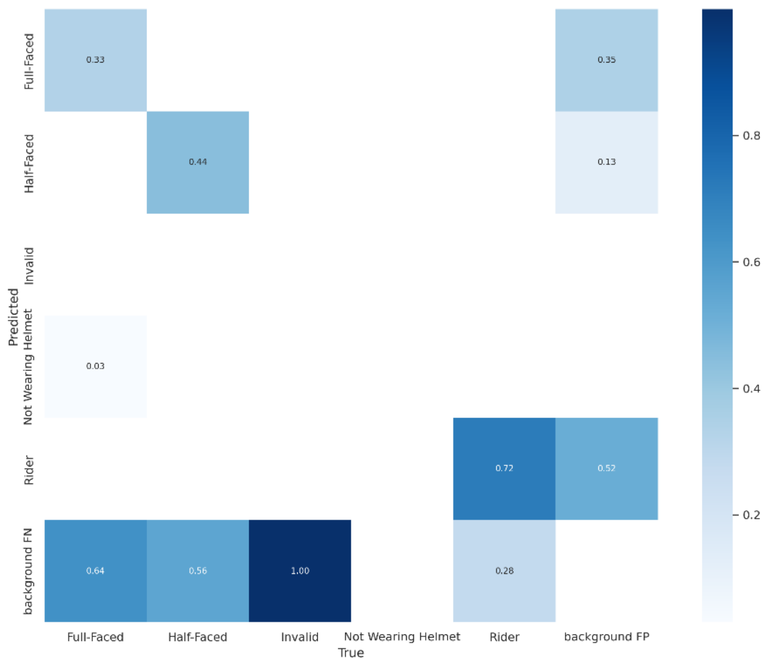

train
train/box_loss
tag: train/box_loss
train/cls_loss
tag: train/cls_loss
train/obj_loss
tag: train/obj_loss


Precision
Recall
Full-Faced 0.400
Half-Faced 0.511
Invalid 0.014
Not Wearing Helmet 0.111
Rider 0.793
all classes 0.365 mAP@0.5

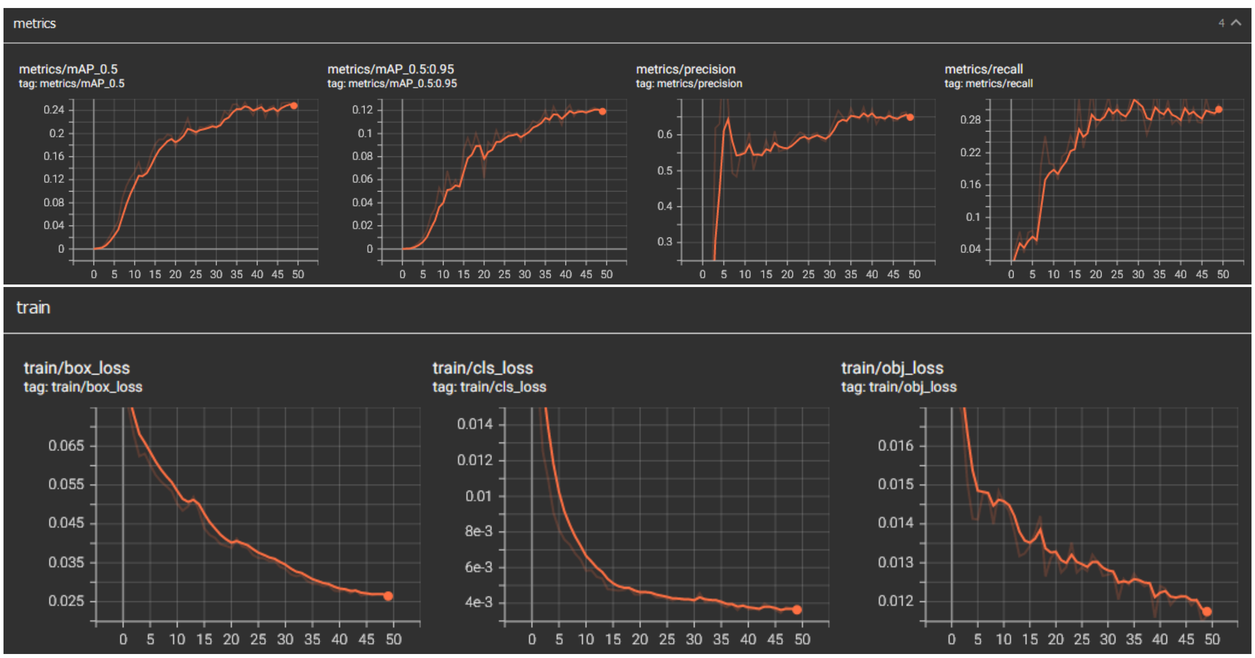

Precision
Confidence
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 1.00 at 0.910

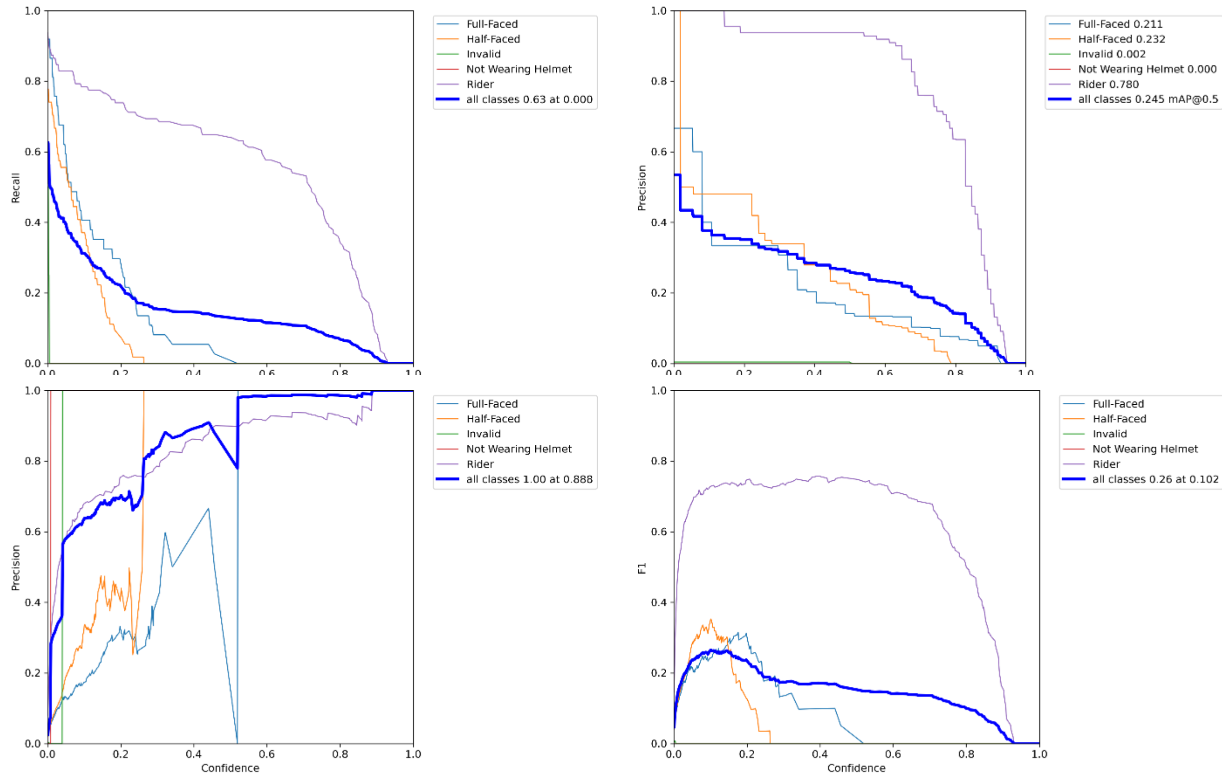

F1
Confidence
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 0.35 at 0.256

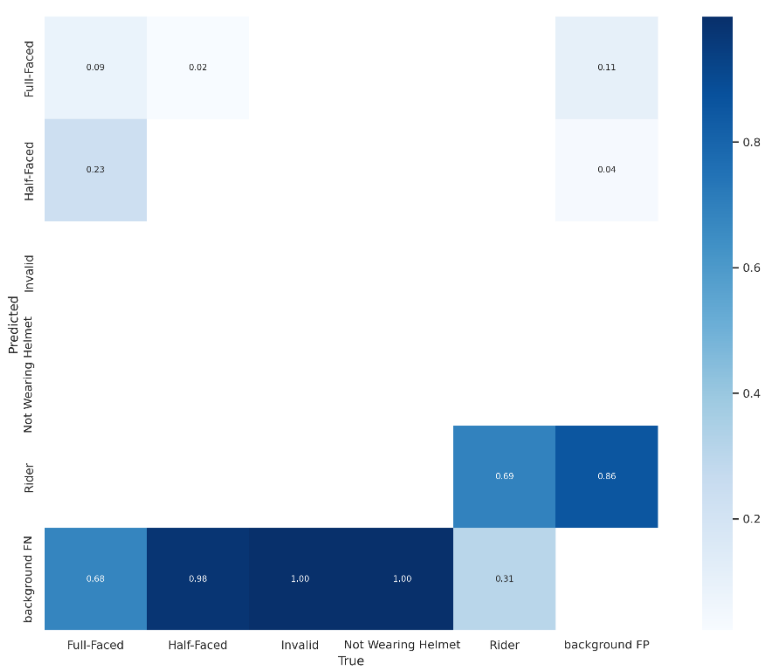

Recall
Confidence
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 0.93 at 0.000



Finally, the results for the 7:00 - 8:00 pm time scenario are in the following figures:




Recall
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 0.63 at 0.000
Precision
Full-Faced 0.211
Half-Faced 0.232
Invalid 0.002
Not Wearing Helmet 0.000
Rider 0.780
all classes 0.245 mAP@0.5
Precision
Confidence
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 1.00 at 0.888
F1
Confidence
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
all classes 0.26 at 0.102


Predicted
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
background FN
0.09
0.02
0.11
0.23
0.04
0.69
0.86
0.68
0.98
1.00
1.00
0.31
Full-Faced
Half-Faced
Invalid
Not Wearing Helmet
Rider
background FP
True

VII. Testing the YOLOv7 model

The testing dataset will be used to test the model after it has been trained using the training dataset. This dataset's goal is to evaluate the model's performance indicators and detection abilities. In order to achieve this goal, *detect.py* was used to identify objects and predict the placement of the bounding boxes from the test photos.

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 640 --
conf 0.1 --source {dataset.location}/test/images
```

```
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.1, device='', exist_ok=False, img_size=640, iou_thres=0.45, name='exp', no_trace=F
YOLOR 🚀 v0.1-121-g2fdc7f1 torch 1.13.0+cu116 CUDA:0 (Tesla T4, 15109.75MB)

Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
IDetect.fuse
/usr/local/lib/python3.8/dist-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the index
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Model Summary: 314 layers, 36503348 parameters, 6194944 gradients, 103.2 GFLOPS
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!

5 Full-Faceds, 3 Riders, Done. (23.1ms) Inference, (1.4ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-100_jpg.rf.daa98d56597398c66bb44194dc0bbc21.jpg
5 Full-Faceds, 2 Riders, Done. (23.1ms) Inference, (1.0ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-108_jpg.rf.ad2191a305b2f2457c8874046e14ba71.jpg
6 Full-Faceds, 2 Half-Faceds, 5 Riders, Done. (23.0ms) Inference, (0.9ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-112_jpg.rf.eac3bb893906371842333a806067ca73.jpg
1 Full-Faced, 1 Half-Faced, 3 Riders, Done. (23.1ms) Inference, (0.9ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-121_jpg.rf.5843c6c85a690eb6e2ef1e2c22b80136.jpg
2 Full-Faceds, 5 Half-Faceds, 5 Riders, Done. (23.0ms) Inference, (0.9ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-127_jpg.rf.b2d9966ebb02b74989ca35a2d22f032b.jpg
1 Half-Faced, 2 Riders, Done. (23.1ms) Inference, (0.9ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-144_jpg.rf.27949ead6bcdec806db94e4d771ba949.jpg


2 Full-Faceds, 2 Half-Faceds, 3 Riders, Done. (12.9ms) Inference, (0.9ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-254_jpg.rf.9c72b8cf1576c5050d150b39f1d697dd.jpg
5 Full-Faceds, 6 Half-Faceds, 5 Riders, Done. (12.9ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-256_jpg.rf.04eadbfa3474776edef5cd3142cbc080.jpg
4 Full-Faceds, 2 Half-Faceds, 5 Riders, Done. (12.9ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-259_jpg.rf.fb659b404c1eb9b815d4f5c74431160c.jpg
4 Full-Faceds, 6 Half-Faceds, 8 Riders, Done. (13.0ms) Inference, (1.3ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-264_jpg.rf.76112cb5a1324ab66f0b9327d2a78cb1.jpg
9 Half-Faceds, 7 Riders, Done. (12.7ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-284_jpg.rf.fa48a91ec68a629599811c07d530893b.jpg
5 Half-Faceds, 7 Riders, Done. (12.7ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-291_jpg.rf.6c01e293d15a444ba92fb3ba21ed7a73.jpg
1 Half-Faced, 4 Riders, Done. (12.6ms) Inference, (0.9ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-293_jpg.rf.200ac70a660e479c97011471499a1401.jpg
2 Half-Faceds, 2 Riders, Done. (12.1ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-35_jpg.rf.5595b60e35b410a207212ec9b7253f42.jpg
2 Full-Faceds, 3 Half-Faceds, 5 Riders, Done. (12.1ms) Inference, (0.9ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-45_jpg.rf.d4be1e46a55b6d49f239277c006b49d0.jpg
1 Half-Faced, 3 Riders, Done. (12.2ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-54_jpg.rf.f0399238194c4f79a01b749a21b9aff7.jpg
1 Full-Faced, 1 Half-Faced, 1 Rider, Done. (12.1ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-62_jpg.rf.c8f3b567ad5bc6d99f9a6dcc8fdda8e1.jpg
3 Riders, Done. (12.1ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-72_jpg.rf.d173c3ca1d4811550092dda660ed6fd4.jpg
6 Full-Faceds, 1 Half-Faced, 5 Riders, Done. (12.2ms) Inference, (0.8ms) NMS
 The image with the result is saved in: runs/detect/exp/XVR_ch9_main_20221004170008_20221004180008_mp4-90_jpg.rf.82c054c354a9cb8a24431454e60a9cd0.jpg
Done. (1.074s)
```

To display all test images with the bounding boxes placed on their predictions, the following code was used:

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

for imageName in glob.glob('/content/yolov7/runs/detect/exp/*.jpg'): #assu
ming JPG
    display(Image(filename=imageName))
```

```python
    print("\n")
```

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

The table output seen in the image below was then created by running the following code. The code snippet will generate a summary of the model's values through a table, as its classes are divided into 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. The results will be saved as a file named "best.pt."

```
!python test.py --weights './runs/train/exp/weights/best.pt' --
data {dataset.location}/data.yaml --img 640
```

Output for the 5:00 – 6:00 pm time scenario:

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95:
all	62	436	0.537	0.382	0.317	0.153
Full-Faced	62	109	0.375	0.596	0.413	0.111
Half-Faced	62	86	0.242	0.449	0.225	0.0825
Invalid	62	22	0.328	0.0448	0.0473	0.0162
Not Wearing Helmet	62	26	1	0	0.0494	0.0144
Rider	62	193	0.74	0.819	0.852	0.541

Output for the 6:00 – 7:00 pm time scenario:

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95:
all	56	158	0.897	0.298	0.356	0.168
Full-Faced	56	32	0.611	0.344	0.402	0.172
Half-Faced	56	18	1	0.441	0.536	0.2
Invalid	56	1	1	0	0.0105	0.00629
Not Wearing Helmet	56	1	1	0	0.0356	0.0213
Rider	56	106	0.872	0.708	0.797	0.442

Output for the 7:00 – 8:00 pm time scenario:

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	62	208	0.649	0.274	0.244	0.123
Full-Faced	62	37	0.174	0.297	0.179	0.0494
Half-Faced	62	54	0.374	0.278	0.248	0.0703
Invalid	62	4	1	0	0.000769	0.000385
Not Wearing Helmet	62	2	1	0	0	0
Rider	62	111	0.697	0.793	0.791	0.495

VIII. Ground Truth of Dataset and Confusion Matrix

We would be using the same method for manually counting all classes within the test dataset. Using excel to accelerate the process, the following tables were created:

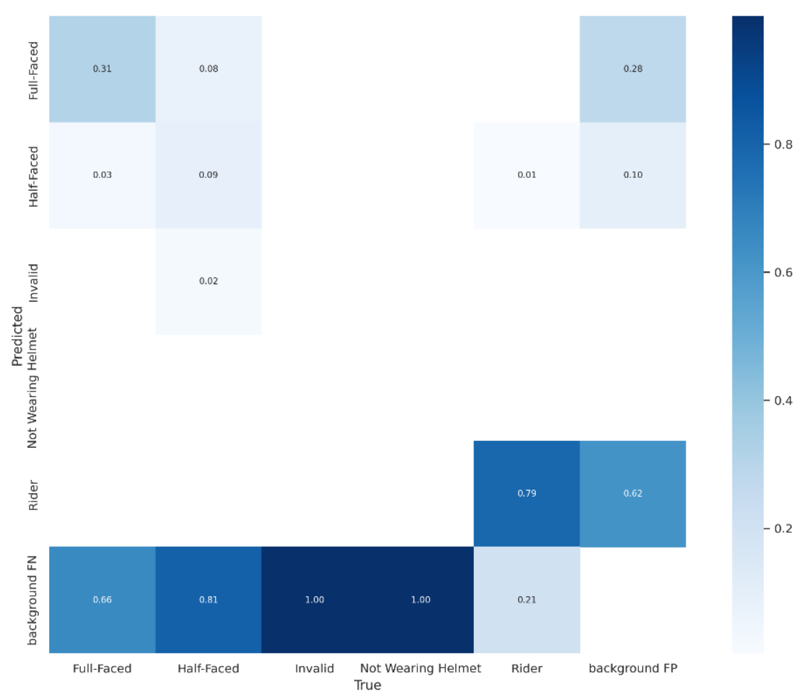
5:00PM - 6:00PM	
Total	
Rider	99
Full-Faced	53
Half-Faced	41
Invalid	14
Not Wearing Helmet	6

6:00PM - 7:00PM	
Total	
Rider	69
Full-Faced	24
Half-Faced	13
Invalid	0
Not Wearing Helmet	1

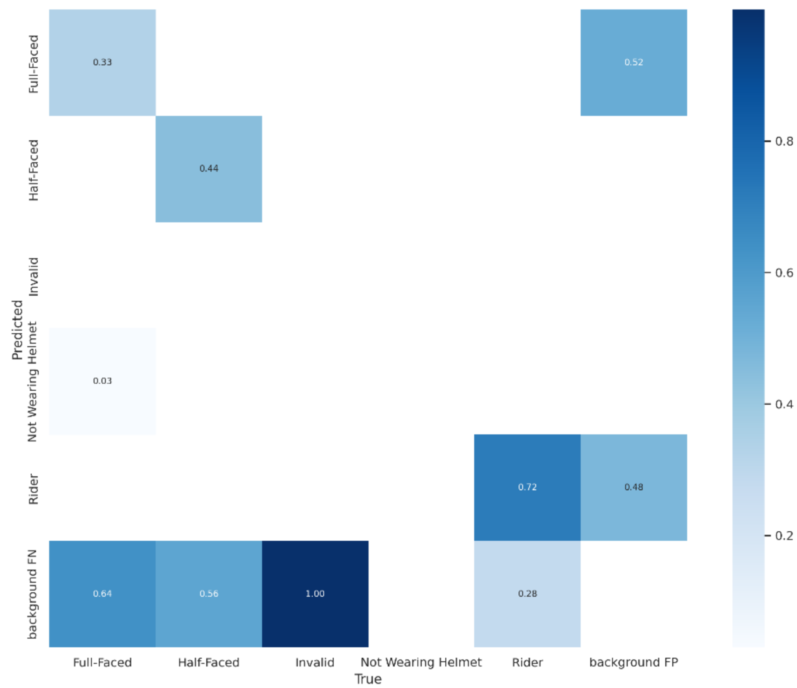
7:00PM-8:00PM	
Total	
Rider	66
Full-Faced	35
Half-Faced	23
Invalid	2
Not Wearing Helmet	0

After testing the YOLOv7 model, we were able to produce these confusion matrices.

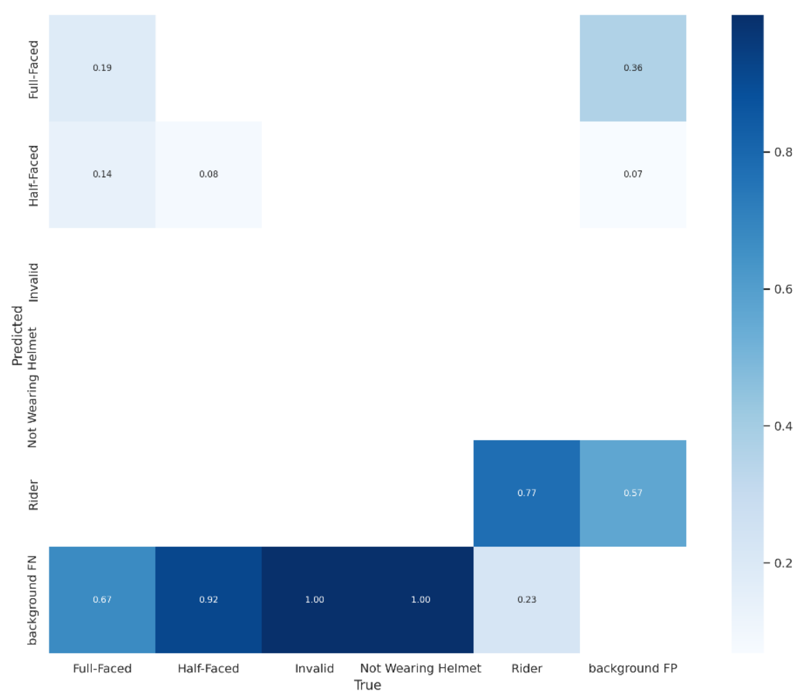
5:00 – 6:00 PM



6:00 – 7:00 PM



7:00 – 8:00 PM



IX. Collab Notebooks

Below are the links to our collab notebooks:

1. 5:00 – 6:00 pm scenario

   https://colab.research.google.com/drive/1rtCEkPe7VSbuuEtunktTeXTIKOmLNiao?usp=sharing

2. 6:00 – 7:00 pm scenario

   https://colab.research.google.com/drive/1tYhQjceTyuyJ1Mg4mHfg-Otl-gLM1UdW?usp=sharing

3. 7:00 – 8:00 pm scenario
   https://colab.research.google.com/drive/1S7k35CjCrOshuz2DWnFThMaX7edaEBOc?usp=sharing

References

Dwyer, B. (2022, December 29). *How to Train YOLOv7 on a Custom Dataset*. Roboflow Blog. https://blog.roboflow.com/yolov7-custom-dataset-training-tutorial/

Skelton, J. (2022, August 17). *How to train and use a custom YOLOv7 model*. Paperspace Blog. https://blog.paperspace.com/yolov7/