IP Project Members:
Vidhi Sharma 2019286
Harshita Gupta 2019467
Udit Bhati 2019281

IP REPORT
Surveillance Pruning for Traffic Detection

Dataset Description:

We worked on the UA-DETRAC dataset. The dataset contains description, video sample with annotations, benchmark, and annotation categories

- Multi-object tracking and detection dataset
- 10 hours of video at 24 locations in Beijing and Tianjin in China
- Recorded at 25 frames per second
- Resolution is 960 x 540 pixels
- 8250 vehicles that are manually annotated
- Vehicle categories are Car, Bus, Van, and Other

Dataset download:

```
wget -O DETRAC-train-data.zip http://detrac-db.rit.albany.edu/Data/DETRAC-train-data.zip
#wget -O DETRAC-test-data.zip http://detrac-db.rit.albany.edu/Data/DETRAC-test-data.zip
wget -O DETRAC-Train-Annotations-XML.zip http://detrac-db.rit.albany.edu/Data/DETRAC-Train-Annotations-XML.zip
```

Annotations for the dataset were not readily available online.

Annotated XML file for test:

wget -O DETRAC-Test-Annotations-XML.zip http://detrac-db.rit.albany.edu/Data/DETRAC-Test-Annotations-XML.zip

Preparing custom dataset for Yolov5:

- Converted the dataset into yolov5 format by using https://github.com/wy17646051/UA-DETRAC-Format-Converter/blob/main/process.py. Run '$ python process.py'
- Folders created:
    - Images
        - Train
        - Val
    - labels(.txt files generated mentioning the objects and their locations)
        - Train

- val
- Reduced the size of the dataset.
- We added the data configuration file, dataset.yaml, to the yolov5 folder.

SparseMLPruning Model:

SparseML recipes only work with [Neural Magic's YOLOv5 fork](#) and NOT WORK with the original YOLOv5 by Ultralytics.

There are 3 methods to sparsify models with SparseML:

- 1️⃣ Post-training (One-shot).
- 2️⃣ Sparse Transfer Learning.
- 3️⃣ Training Aware.

NOTE: 1️⃣ does not require re-training but only supports dynamic quantization. 2️⃣ and 3️⃣ requires re-training and supports pruning and quantization, which may give better results.

Methodology:

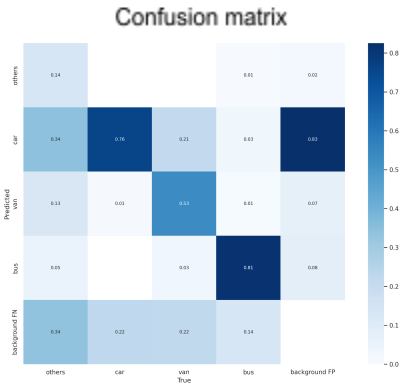
We used the following sparseML commands:

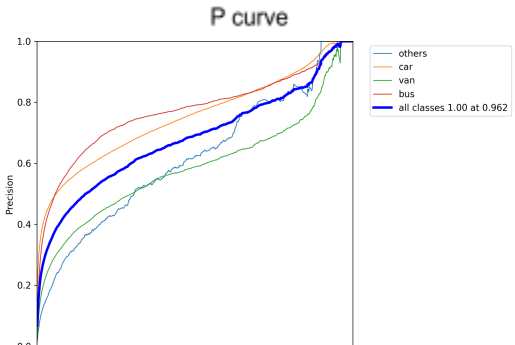
1. **For training dataset:**
   a. **Baseline YOLOv5s model:** let's start by training a baseline model with no optimization.

```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
     24/24     1.91G    0.0206   0.02486  0.002224       169       416: 100%|██████████| 276/276 [01:52<00:00,  2.46it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 17/17 [00:39<00:00,  2.32s/i
                  all       2100      24919      0.651      0.545      0.572      0.413

26 epochs completed in 1.052 hours.
Optimizer stripped from yolov5-pruned/yolov5s-sgd3/weights/last.pt, 14.4MB
Optimizer stripped from yolov5-pruned/yolov5s-sgd3/weights/best.pt, 14.4MB

Validating yolov5-pruned/yolov5s-sgd3/weights/best.pt...
Fusing layers...
YOLOv5s summary: 224 layers, 7062001 parameters, 0 gradients
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 17/17 [01:13<00:00,  4.34s/i
                  all       2100      24919       0.71      0.552      0.595      0.428
               others       2100        600      0.649      0.168      0.236      0.153
                  car       2100      20383      0.784      0.702      0.772      0.547
                  van       2100       1444      0.606      0.533      0.535      0.402
                  bus       2100       2492      0.801      0.803      0.835      0.611
Results saved to yolov5-pruned/yolov5s-sgd3
```

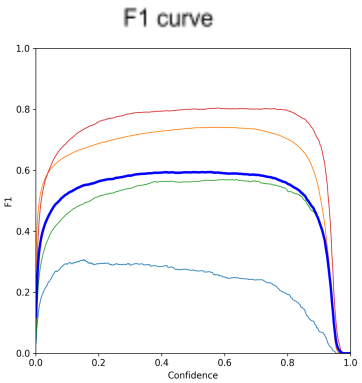
Command:

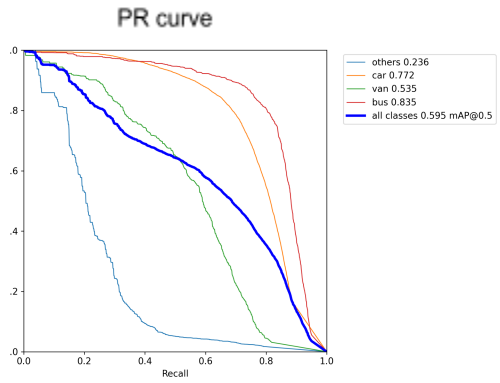
python train.py --cfg ./models_v5.0/yolov5s.yaml --data dataset.yaml --hyp data/hyps/hyp.scratch.yaml --weights yolov5s.pt --img 416 --batch-size 64 --optimizer SGD --epochs 25 --project yolov5-pruned

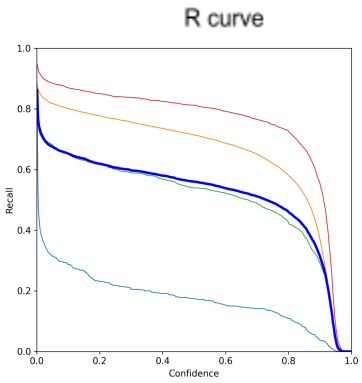


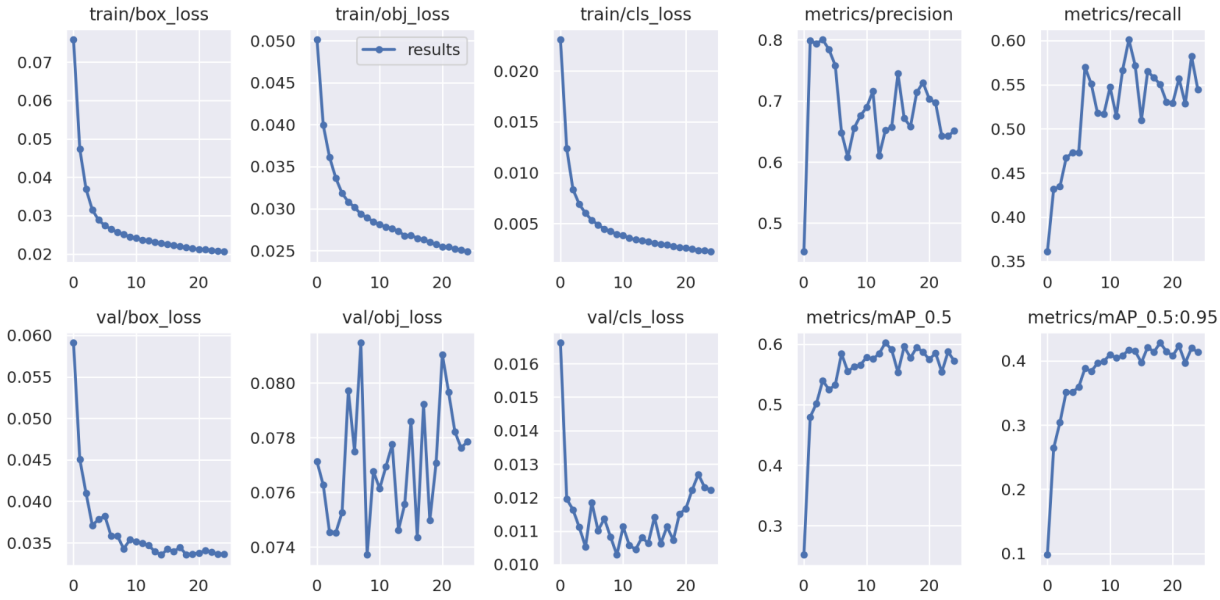


--name yolov5s-sgd3





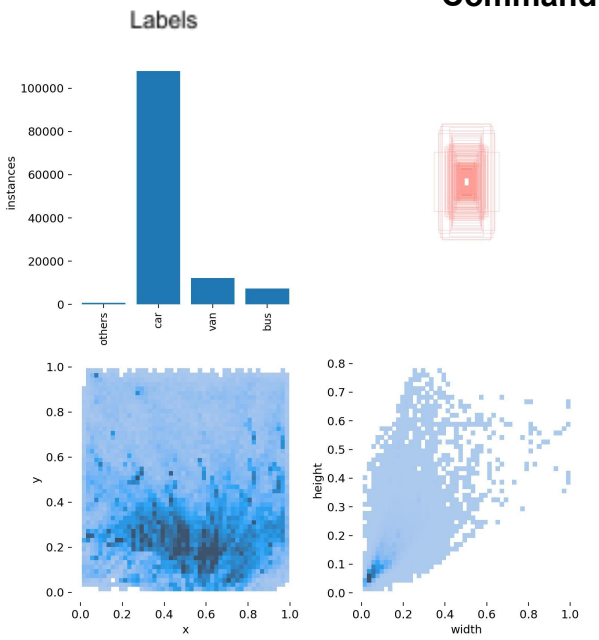




Results

b. **One-shot sparseML pruned YOLOv5s model:** The one-shot method is the easiest way to sparsify an existing model as it doesn't require re-training.

Command:

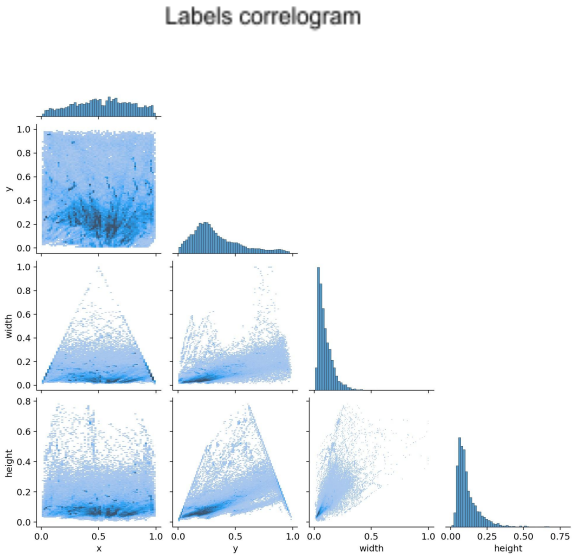


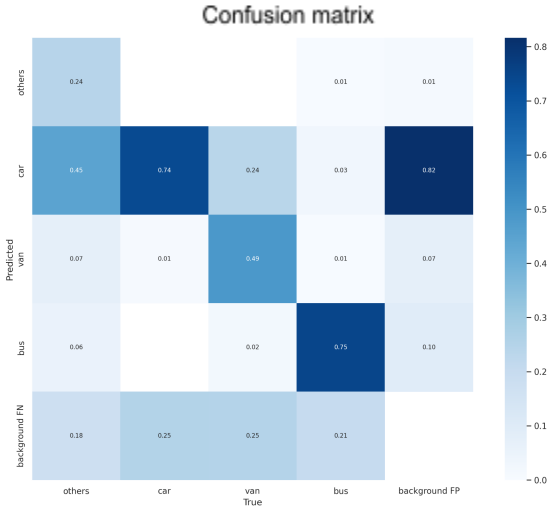
python train.py --cfg ./models_v5.0/yolov5s.yaml --recipe ../recipes/yolov5s.pruned.md --data dataset.yaml --hyp data/hyps/hyp.scratch.yaml --weights yolov5-pruned/yolov5s-sgd3/weights/best.pt --img 416 --batch-size 64 --optimizer SGD --epochs 25 --project yolov5-pruned --name yolov5s-sgd-one-shot --one-shot

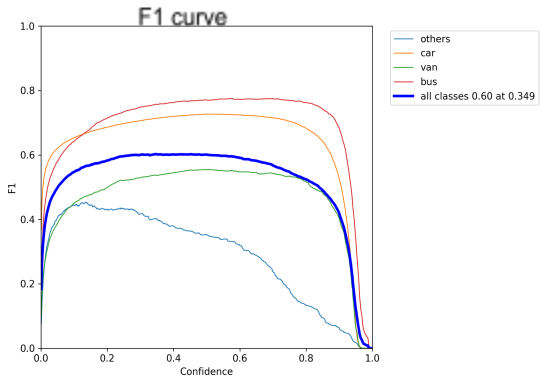
c. **Pruned YOLOv5s:** Here, instead of taking an already sparsified model, we are going to sparsify our model by pruning it ourselves.

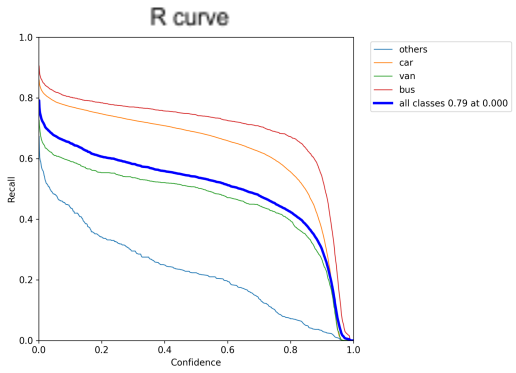
Command:

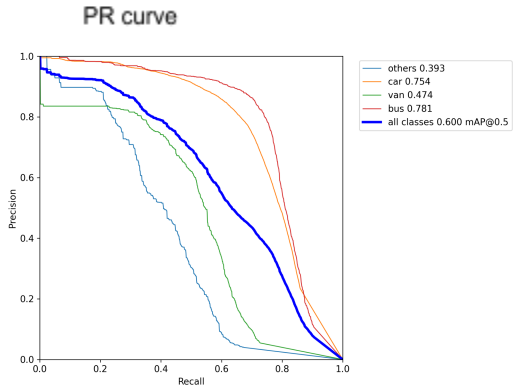
python train.py --cfg ./models_v5.0/yolov5s.yaml --recipe ../recipes/yolov5s.pruned.md --data dataset.yaml --hyp data/hyps/hyp.scratch.yaml --weights yolov5s.pt --img 416 --batch-size 64 --optimizer SGD --project yolov5-pruned --name yolov5s-sgd-pruned

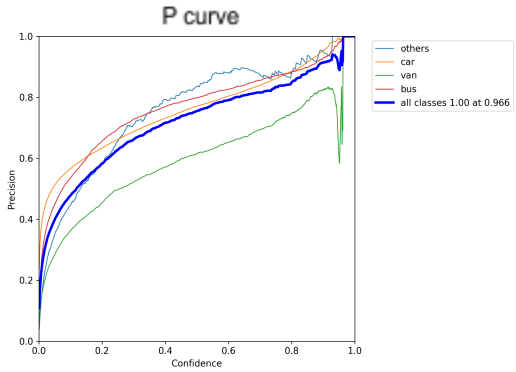




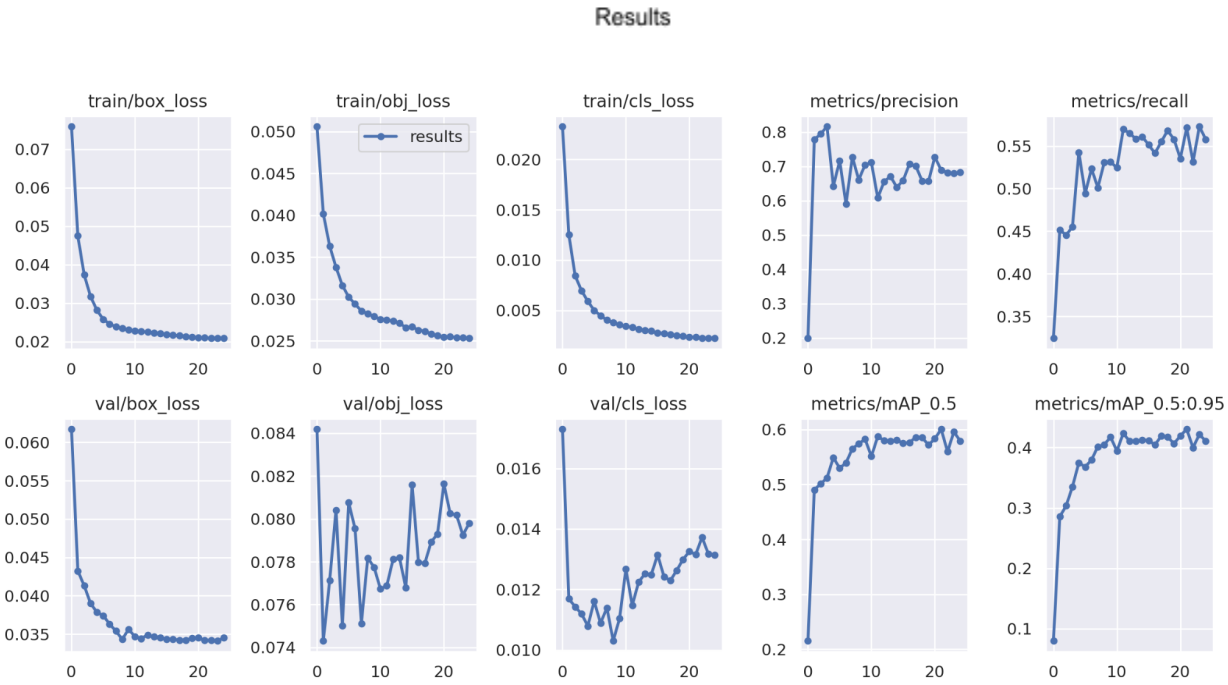

F1 curve
F1
Confidence
others
car
van
bus
all classes 0.60 at 0.349


R curve
Recall
Confidence
others
car
van
bus
all classes 0.79 at 0.000


PR curve
Precision
Recall
others 0.393
car 0.754
van 0.474
bus 0.781
all classes 0.600 mAP@0.5


P curve
Precision
Confidence
others
car
van
bus
all classes 1.00 at 0.966


MVI_40871_img00047.jpg
MVI_40751_img01406.jpg
MVI_40191_img00333.jpg
MVI_20052_img00313.jpg
MVI_40141_img00148.jpg
MVI_63525_img00683.jpg
MVI_63553_img00616.jpg
MVI_40992_img01840.jpg

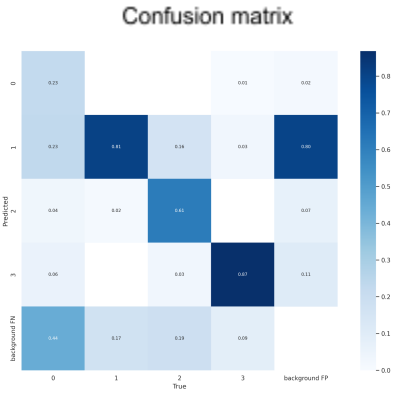


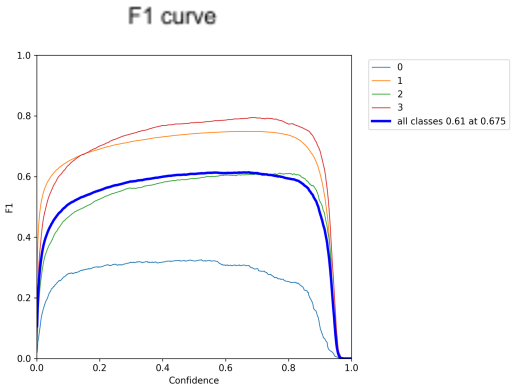
2. For validation dataset:

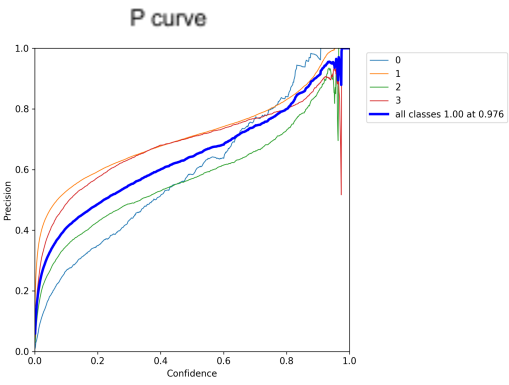
a. Baseline YOLOv5s model:

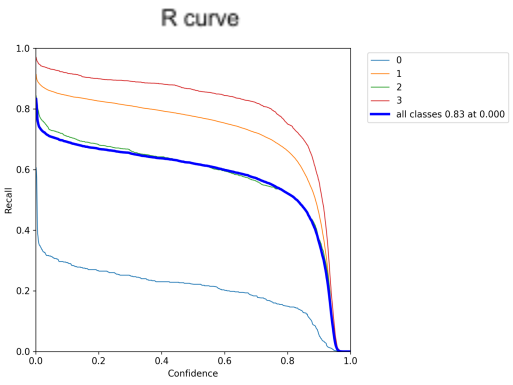
Command:

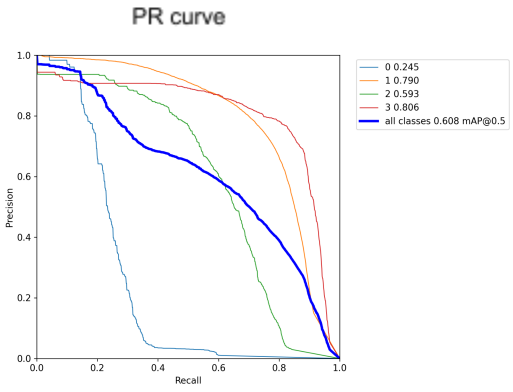
python val.py --weights yolov5-pruned/yolov5s-sgd3/weights/best.pt --data dataset.yaml

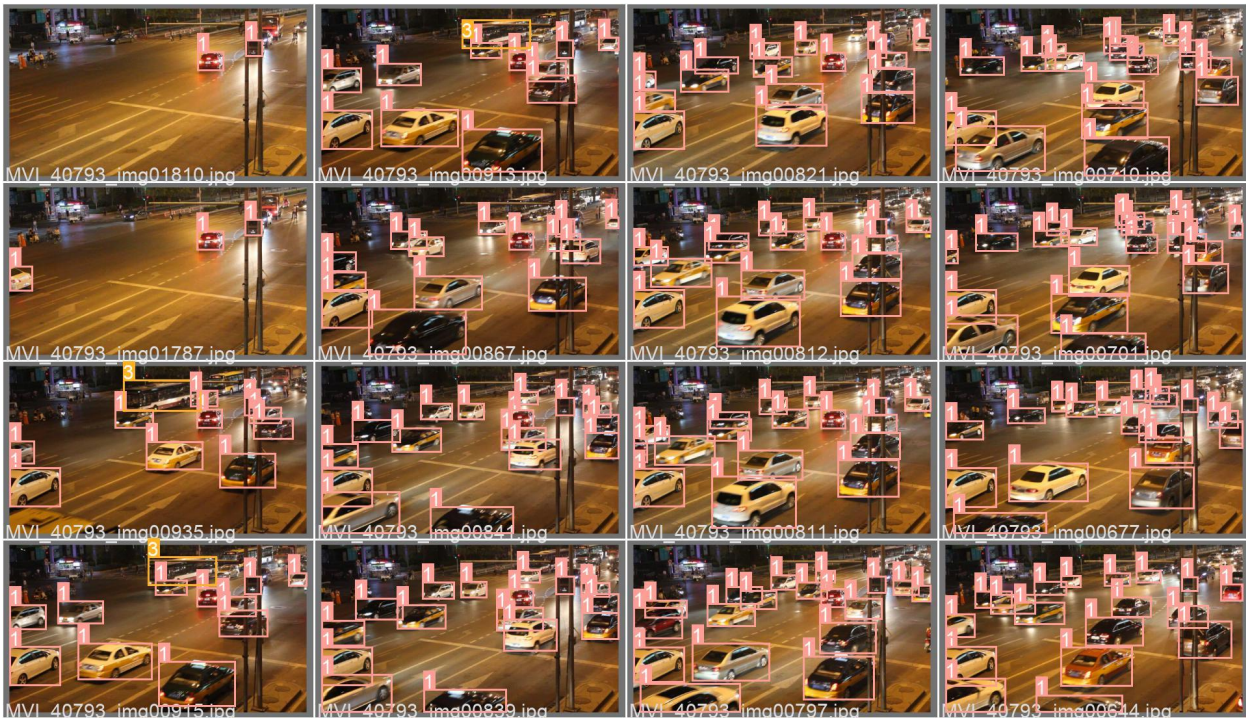


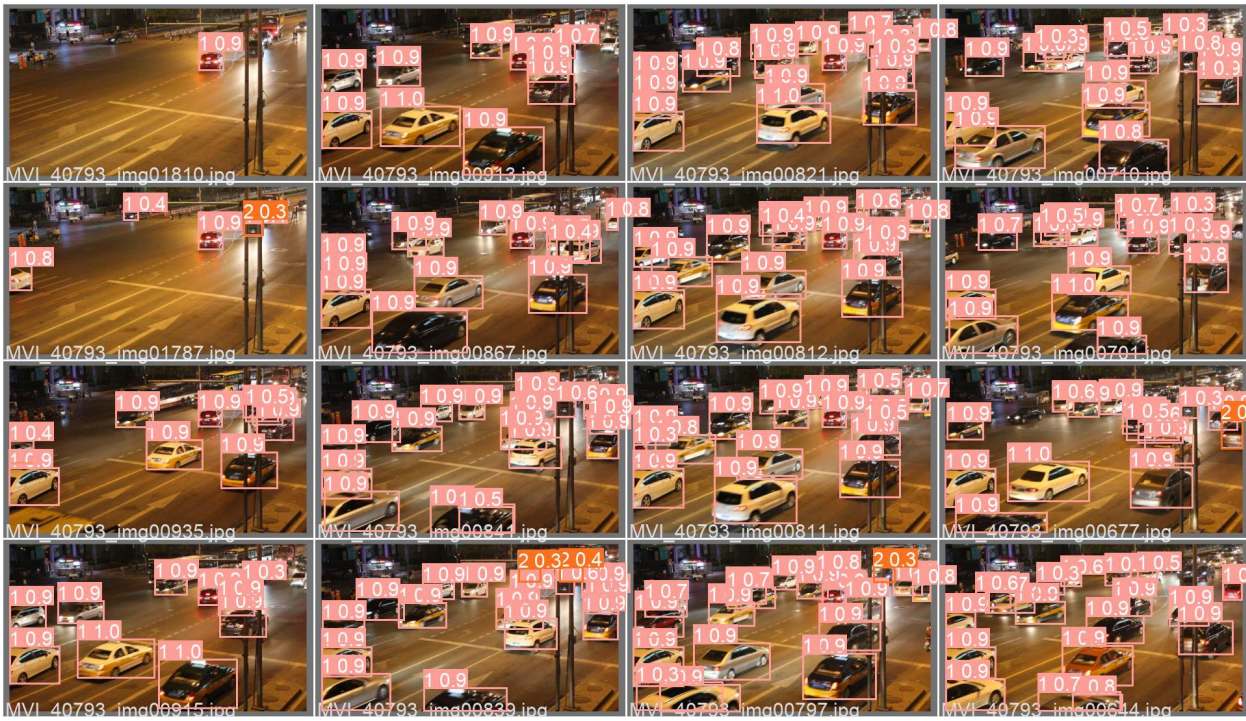







PR curve
Precision
Recall
0 0.245
1 0.790
2 0.593
3 0.806
all classes 0.608 mAP@0.5

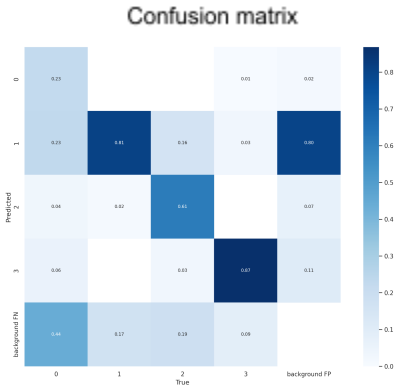

MVI_40793_img01810.jpg
MVI_40793_img00913.jpg
MVI_40793_img00821.jpg
MVI_40793_img00710.jpg
MVI_40793_img01787.jpg
MVI_40793_img00867.jpg
MVI_40793_img00812.jpg
MVI_40793_img00701.jpg
MVI_40793_img00935.jpg
MVI_40793_img00841.jpg
MVI_40793_img00811.jpg
MVI_40793_img00677.jpg
MVI_40793_img00915.jpg
MVI_40793_img00839.jpg
MVI_40793_img00797.jpg
MVI_40793_img00644.jpg

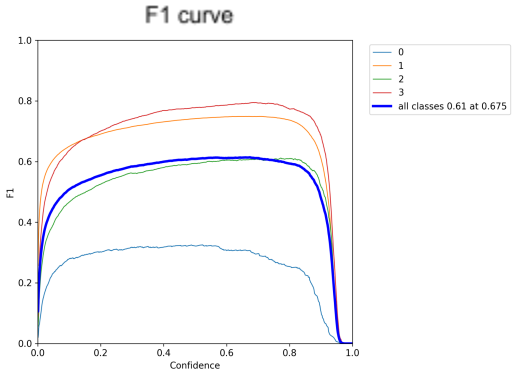


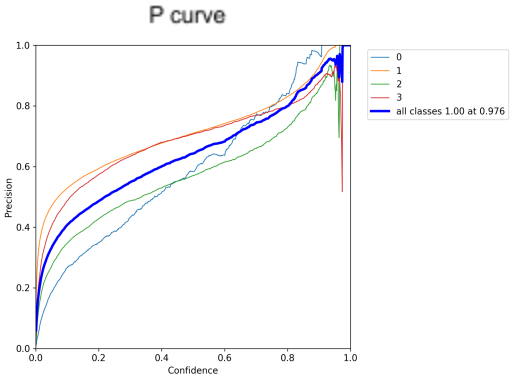
b. One shot sparseML pruned YOLOv5s model:

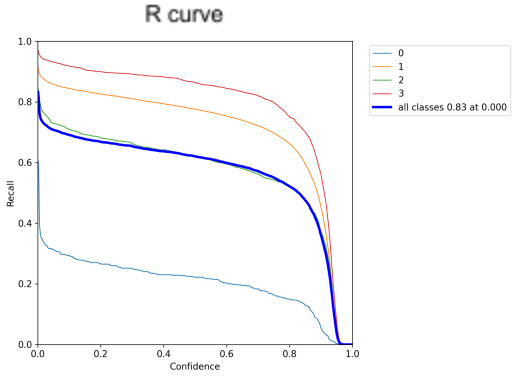
Command:

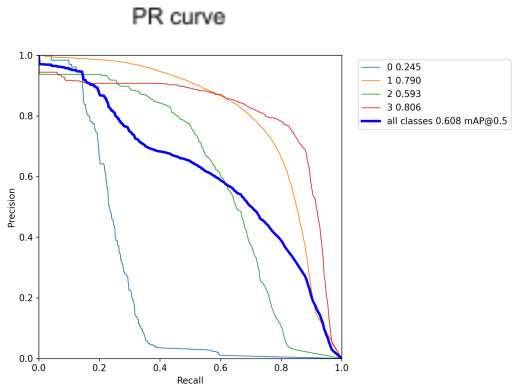
python val.py --weights yolov5-pruned/yolov5s-sgd-one-shot/weights/checkpoint-one-shot.pt --data dataset.yaml

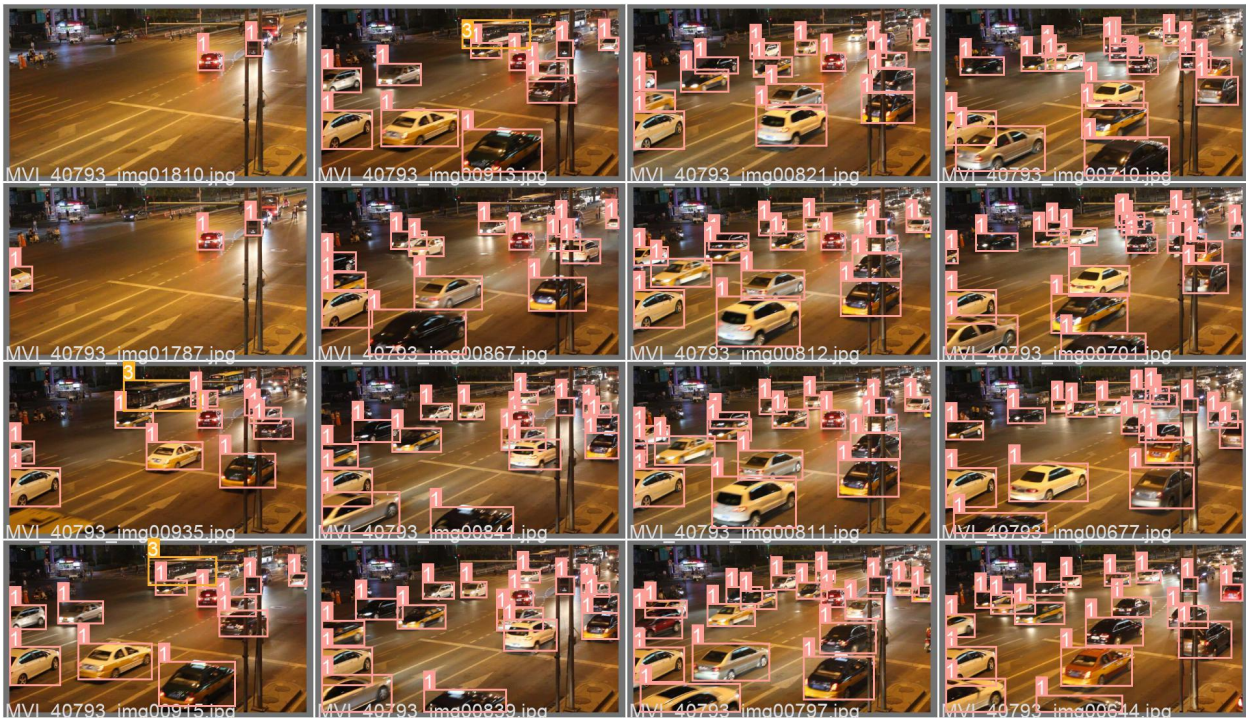


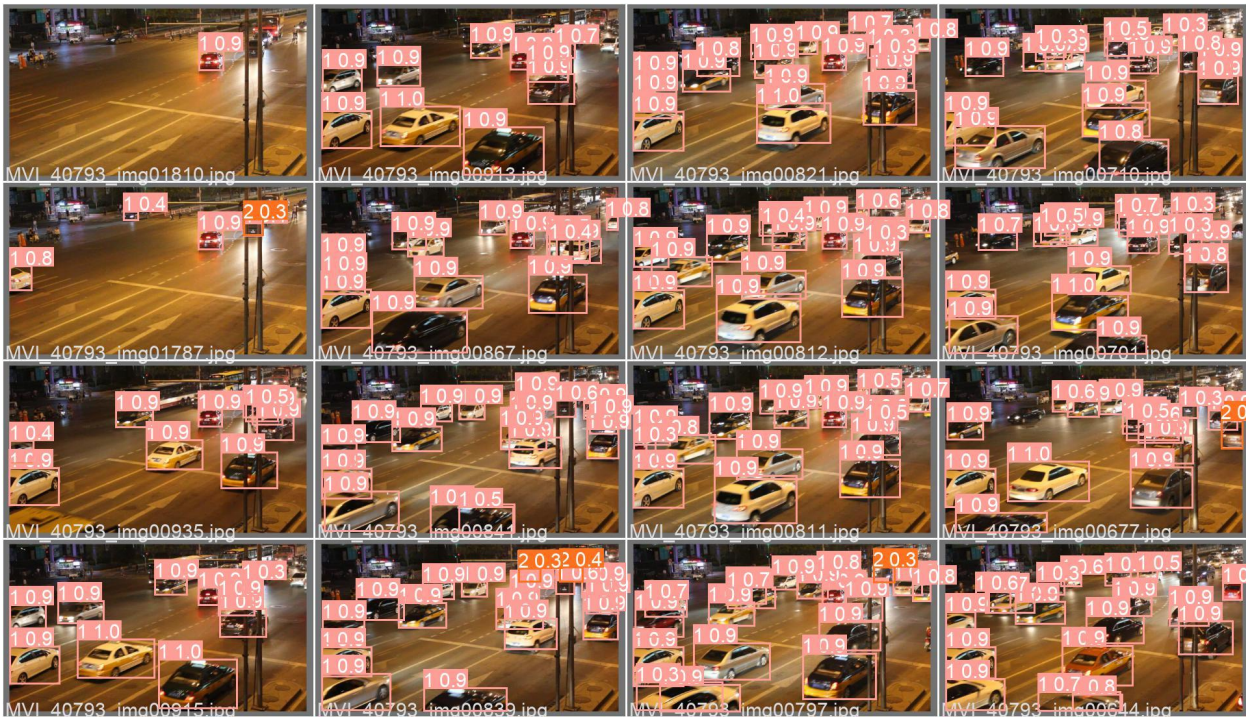







PR curve
Precision
Recall
0 0.245
1 0.790
2 0.593
3 0.806
all classes 0.608 mAP@0.5

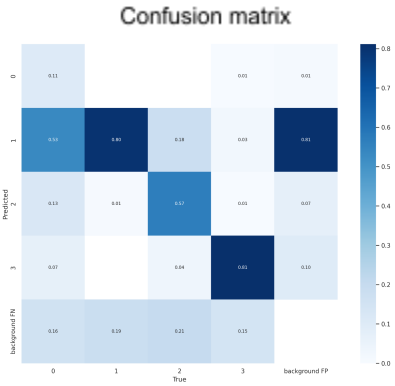

MVI_40793_img01810.jpg
MVI_40793_img00913.jpg
MVI_40793_img00821.jpg
MVI_40793_img00710.jpg
MVI_40793_img01787.jpg
MVI_40793_img00867.jpg
MVI_40793_img00812.jpg
MVI_40793_img00701.jpg
MVI_40793_img00935.jpg
MVI_40793_img00841.jpg
MVI_40793_img00811.jpg
MVI_40793_img00677.jpg
MVI_40793_img00915.jpg
MVI_40793_img00839.jpg
MVI_40793_img00797.jpg
MVI_40793_img00644.jpg

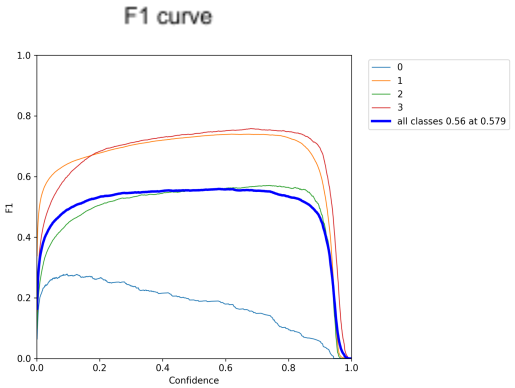


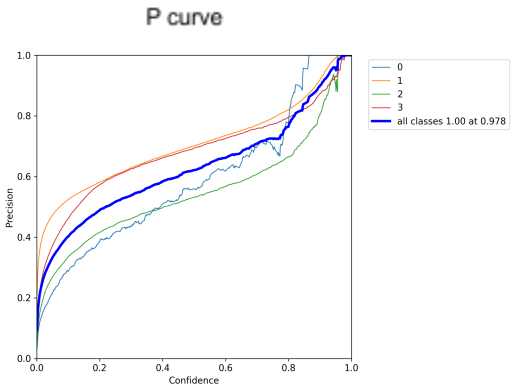
c. Pruned YOLOv5s:

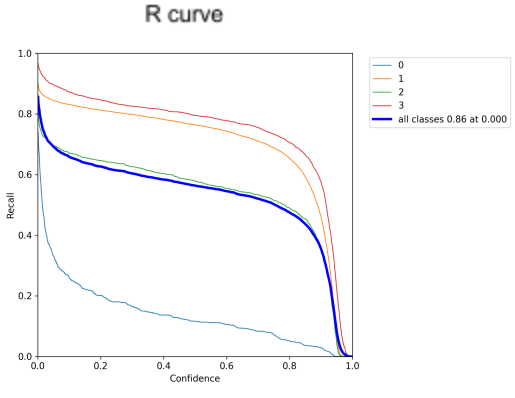
Command:

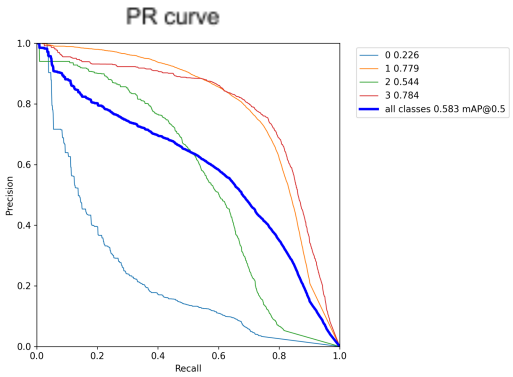
python val.py --weights yolov5-pruned/yolov5s-sgd-pruned/weights/best.pt --data dataset.yaml

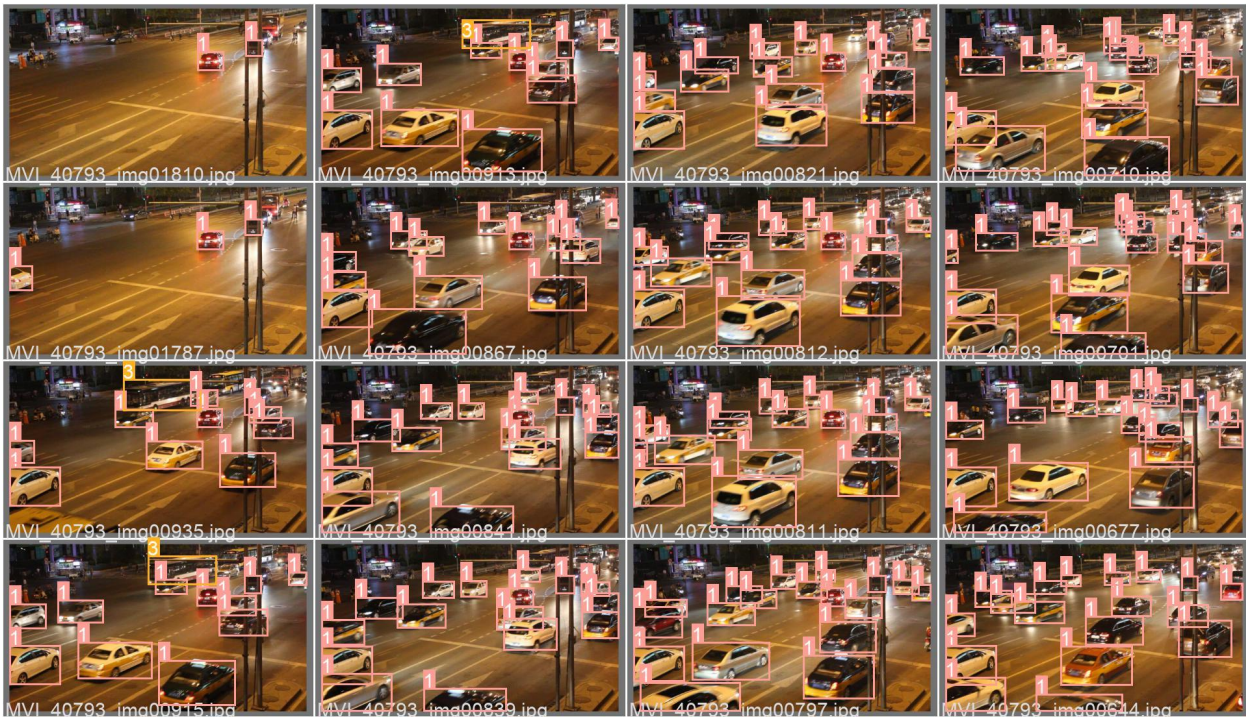


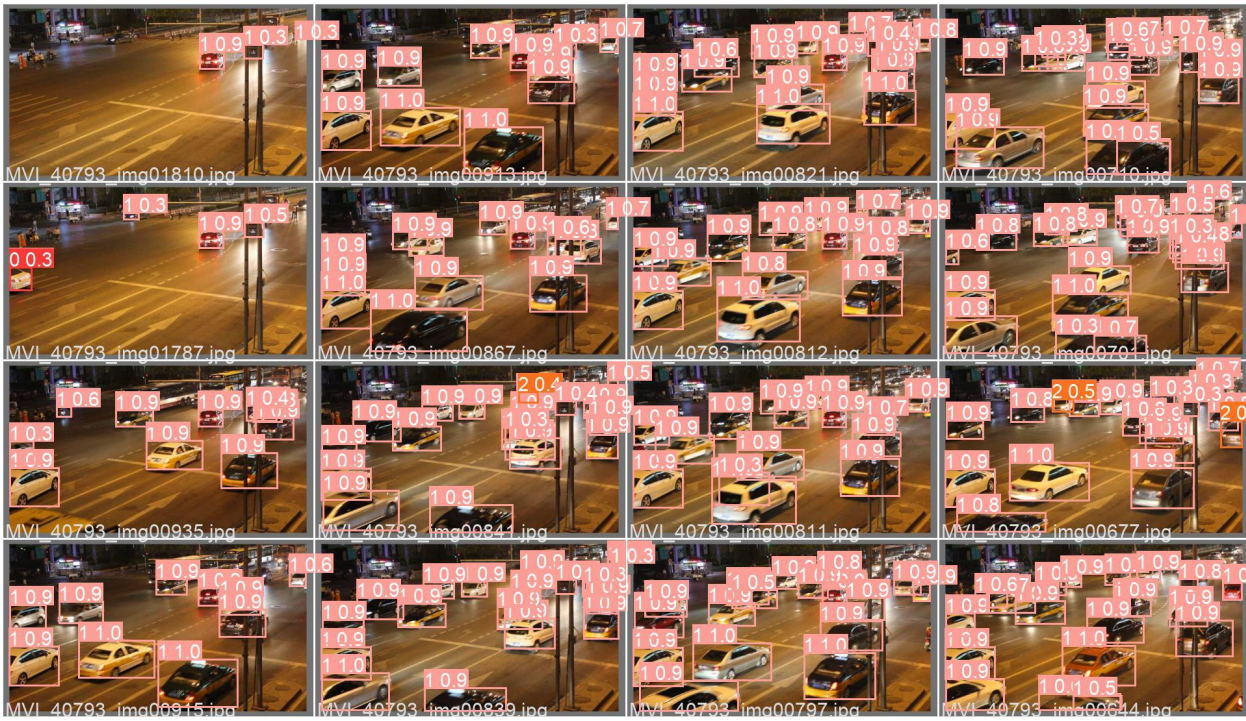







PR curve
Precision
Recall
0 0.226
1 0.779
2 0.544
3 0.784
all classes 0.583 mAP@0.5


MVI_40793_img01810.jpg
MVI_40793_img00913.jpg
MVI_40793_img00821.jpg
MVI_40793_img00710.jpg
MVI_40793_img01787.jpg
MVI_40793_img00867.jpg
MVI_40793_img00812.jpg
MVI_40793_img00701.jpg
MVI_40793_img00935.jpg
MVI_40793_img00841.jpg
MVI_40793_img00811.jpg
MVI_40793_img00677.jpg
MVI_40793_img00915.jpg
MVI_40793_img00839.jpg
MVI_40793_img00797.jpg
MVI_40793_img00644.jpg

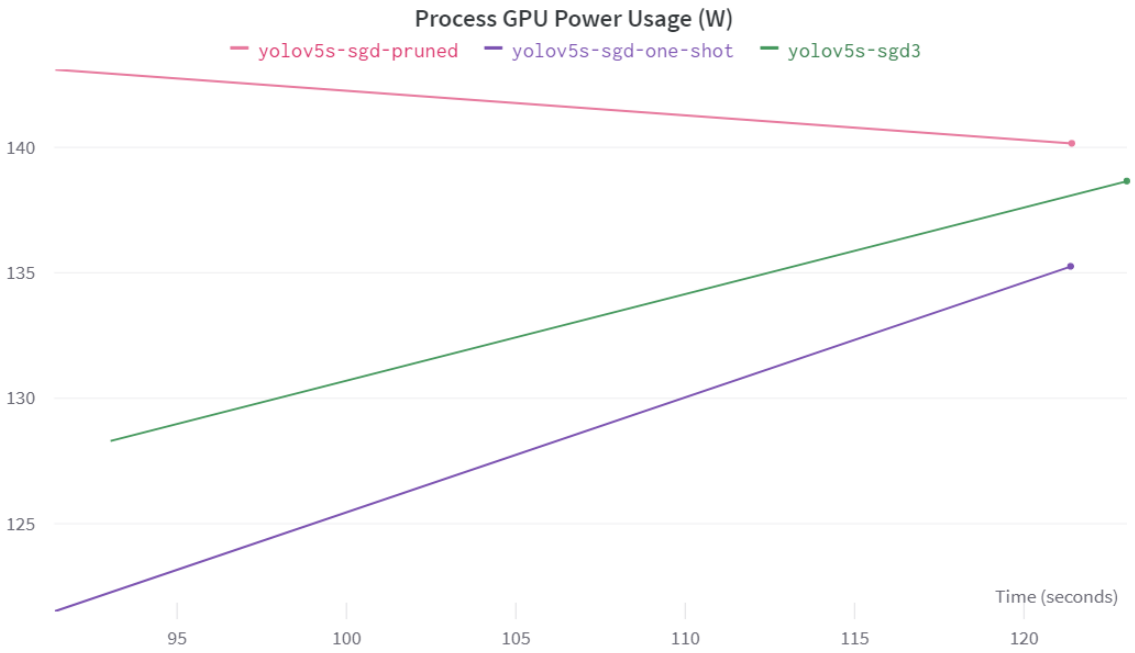


Results:

The graphs for GPU power usage, GPU memory allocation, GPU utilization are attached below.

Mean average precision or mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

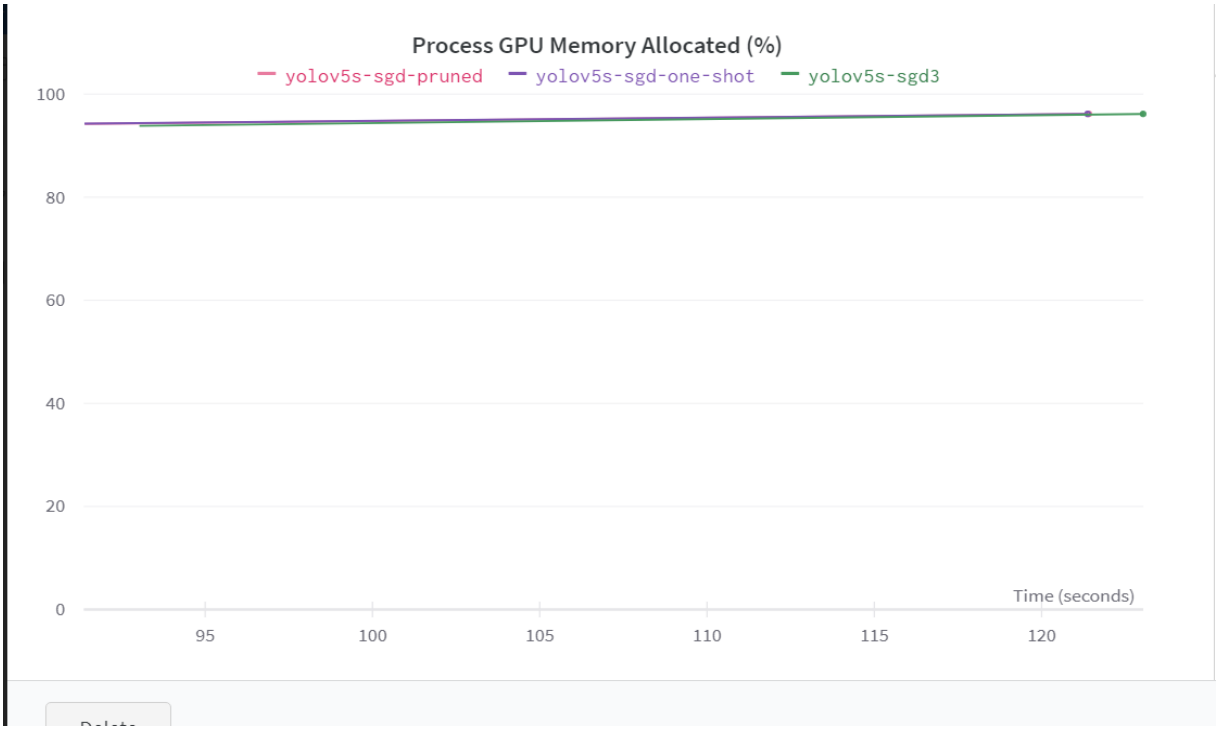
1. On validation:



a)

Command: python val.py --weights yolov5-pruned/yolov5s-sgd3/weights/best.pt --data dataset.yaml

```
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 66/66 [01:15<00:00,  1.15s/it]
                   all       2100      24919       0.73      0.581      0.608       0.45
                     0       2100        600      0.743      0.193      0.245      0.178
                     1       2100      20383      0.768      0.731       0.79      0.581
                     2       2100       1444      0.646      0.572      0.593       0.46
                     3       2100       2492      0.762      0.828      0.806      0.582
Speed: 0.1ms pre-process, 1.4ms inference, 4.7ms NMS per image at shape (32, 3, 640, 640)
Results saved to [1mruns/val/exp
```

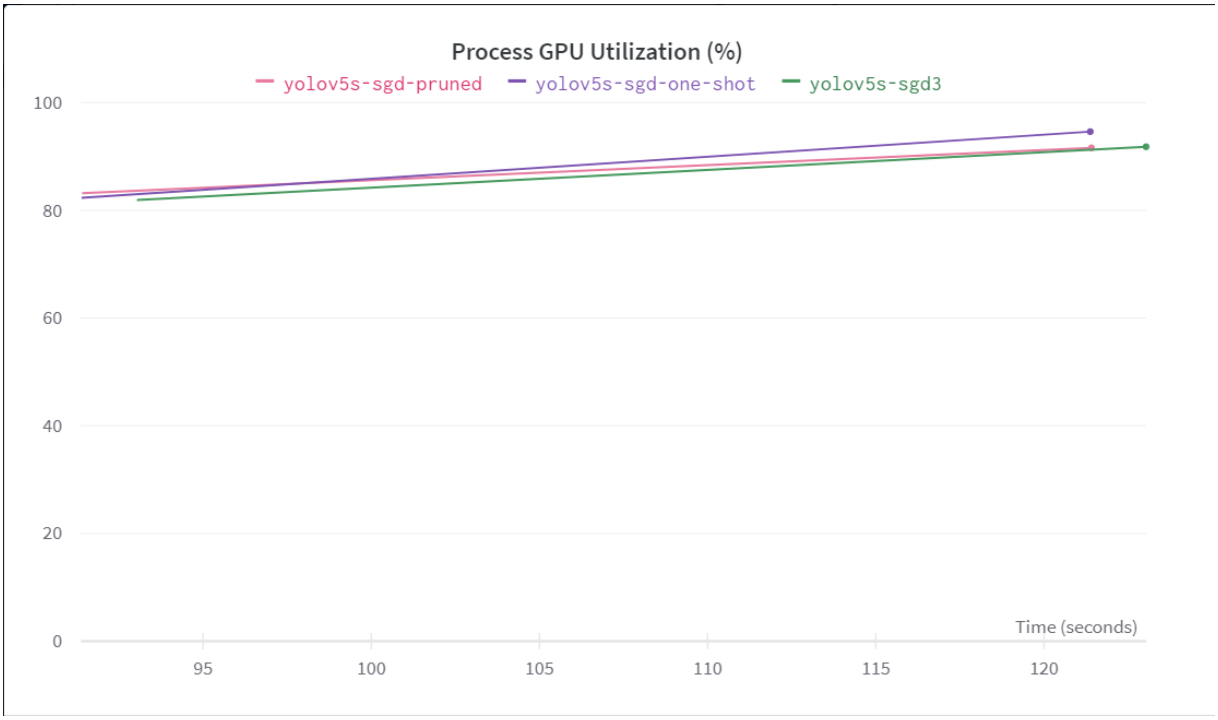


b)

Command: python val.py --weights yolov5-pruned/yolov5s-sgd-one-shot/weights/checkpoint-one-shot.pt --data dataset.yaml

Class	Images	Labels	P	R	mAP@.5	
all	2100	24919	0.73	0.581	0.608	0.45
0	2100	600	0.743	0.193	0.245	0.178
1	2100	20383	0.768	0.731	0.79	0.581
2	2100	1444	0.646	0.572	0.593	0.46
3	2100	2492	0.762	0.828	0.806	0.582

```
Speed: 0.2ms pre-process, 1.8ms inference, 4.6ms NMS per image at shape (32, 3, 640, 640)
Results saved to [1mruns/val/exp2
```



c)

Command: python val.py --weights yolov5-pruned/yolov5s-sgd-pruned/weights/best.pt --data dataset.yaml

```
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 66/66 [01:20<00:00,  1.23s/it]
                 all       2100      24919      0.658      0.551      0.583      0.429
                   0       2100        600      0.625      0.111      0.226      0.141
                   1       2100      20383      0.728      0.748      0.779      0.573
                   2       2100       1444      0.559       0.56      0.544      0.423
                   3       2100       2492       0.72      0.783      0.784      0.581
Speed: 0.2ms pre-process, 1.9ms inference, 4.1ms NMS per image at shape (32, 3, 640, 640)
Results saved to [1mruns/val/exp3
```