به نام خدا

گزارش پروژه درس مکاترونیک بخش پردازش تصویر

دانشجو: امیرحسین اسلامی

بهار ۱۴۰۳

فهرست

راه اندازی و نصب Ultralytics

در گام اول یک نوت بوک درون گوگل کولب باز می کنیم.

سپس این کد هارا جهت نصب Ultralytics می‌نویسیم.

```
%pip install ultralytics
import ultralytics
ultralytics.checks()
```

پس از ران کردن مشاهده می کنیم که نصب با موفقیت انجام می شود.

```
Ultralytics YOLOv8.1.43 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 28.9/78.2 GB disk)
```

حالآ به سراغ ست کردن دیتا ست مورد نظر می رویم.

دیتاستی که ما نیاز داریم تا بتوانیم با استفاده از آن ماشین ها را از بالا تشخیص دهیم دیتاستی خواهد بود که توسط تصاویر هوایی گرفته شده اند و بین دیتاست های آماده ای که درون سایت Ultralytics وجود دارد دیتاست Visdrone می تواند خواسته های مارا برطرف نماید.

آشنایی اولیه با دیتاست Visdrone

جهت آشنایی بیشتر با این دیتاست به سایت Ultralytics می رویم و بخش مربوط به دیتاست مورد نظر را باز می کنیم.

"مجموعه داده VisDrone یک معیار در مقیاس بزرگ است که توسط تیم AISKYEYE در آزمایشگاه یادگیری ماشین و داده کاوی، دانشگاه تیانجین، چین ایجاد شده است. این شامل داده های حقیقت زمینی با دقت مشروح شده برای وظایف مختلف بینایی کامپیوتری مربوط به تجزیه و تحلیل تصویر و ویدئو مبتنی بر هواپیماهای بدون سرنشین است.

VisDrone از ۲۸۸ کلیپ ویدیویی با ۲۶۱۹۰۸ فریم و ۱۰۲۰۹ تصویر ثابت تشکیل شده است که توسط دوربین های مختلف نصب شده بر روی پهپاد گرفته شده است. مجموعه داده طیف گسترده‌ای از جنبه‌ها، از جمله موقعیت مکانی (۱۴ شهر مختلف در سراسر چین)، محیط (شهری و روستایی)، اشیاء (عابران پیاده، وسایل نقلیه، دوچرخه‌ها و غیره) و تراکم (صحنه‌های پراکنده و شلوغ) را پوشش می‌دهد. مجموعه داده با استفاده از سکوهای مختلف پهپاد تحت سناریوهای مختلف و شرایط آب و هوایی و نور جمع آوری شد. این قاب ها به صورت دستی با بیش از ۲.۶

میلیون جعبه محدود کننده اهداف مانند عابران پیاده، اتومبیل ها، دوچرخه ها و سه چرخه ها حاشیه نویسی می شوند. ویژگی هایی مانند دید صحنه، کلاس شی و انسداد نیز برای استفاده بهتر از داده ها ارائه شده است."

ساختار دیتاست Visdrone

مجموعه داده VisDrone در پنج زیرمجموعه اصلی سازماندهی شده است که هر کدام بر روی یک کار خاص تمرکز دارند:

وظیفه ۱: تشخیص اشیاء در تصاویر

وظیفه ۲: تشخیص اشیاء در ویدیوها

وظیفه ۳: ردیابی تک شی

وظیفه ۴: ردیابی چند شی

وظیفه ۵: شمارش جمعیت

کاربردها

مجموعه داده VisDrone به طور گسترده ای برای آموزش و ارزیابی مدل های یادگیری عمیق در وظایف بینایی کامپیوتری مبتنی بر پهپاد مانند تشخیص اشیا، ردیابی اشیا و شمارش جمعیت استفاده می شود. مجموعه متنوع داده‌های حسگر، حاشیه‌نویسی‌ها و ویژگی‌های مجموعه داده، آن را به منبعی ارزشمند برای محققان و متخصصان در زمینه بینایی رایانه مبتنی بر هواپیماهای بدون سرنشین تبدیل می‌کند.

دیتاست YAML

یک فایل YAML (Yet Another Markup Language) برای تعریف پیکربندی مجموعه داده استفاده می شود. این شامل اطلاعاتی در مورد مسیرهای مجموعه داده، کلاس ها و سایر اطلاعات مرتبط است.

اطلاعات صفحهٔ گیت هاب دیتاست Visdrone

```
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: ../datasets/VisDrone # dataset root dir
train: VisDrone2019-DET-train/images # train images (relative to 'path')  6471 images
val: VisDrone2019-DET-val/images # val images (relative to 'path')  548 images
test: VisDrone2019-DET-test-dev/images # test images (optional)  1610 images
```

در این بخش اطلاعات پایه ای که برای دسترسی و استفاده از دیتاست نیاز داریم را مشاهده می کنید مانند آدرس دیتاست یا تصاویری که به واسطهٔ آن دیتاست آموزش داده شده است همچنین حدود ۱۶۱۰ تصویر در اختیار ما گذاشته شده است تا بتوانیم این دیتاست را تست کنیم.

```
# Classes
names:
  0: pedestrian
  1: people
  2: bicycle
  3: car
  4: van
  5: truck
  6: tricycle
  7: awning-tricycle
  8: bus
  9: motor
```

در این تصویر لیست کلاس های دیتاست را مشاهده می کنید همانطور که واضح است این دیتاست شامل ۱۰ کلاس از اشیاء و موجوداتی هست که می تواند آن ها را ردیابی و تشخیص بدهد.

در ادامه هم کد های دانلود این دیتاست در اختیار توسعه دهندگان قرار داده شده است.

Train مدل در گوگل کولب

```
!yolo train model = yolov8l.pt data = VisDrone.yaml epochs=100
#from ultralytics import YOLO

# Load a model
#model = YOLO('yolov8n.pt')  # load a pretrained model (recommended for training)

# Train the model
#results = model.train(data='VisDrone.yaml', epochs=100, imgsz=640)
```

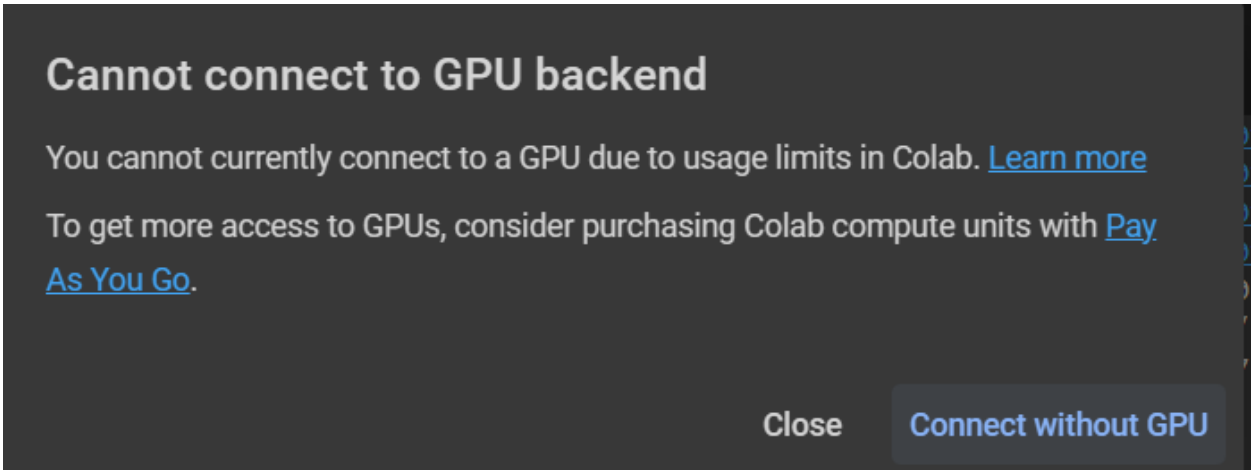
```
Downloading https://github.com/ultralytics/assets/releases/download/v8.1.0/yolov8l.pt to 'yolov8l.pt'...
100% 83.7M/83.7M [00:00<00:00, 282MB/s]
Ultralytics YOLOv8.1.43 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=yolov8l.pt, data=VisDrone.yaml, epochs=100, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False,

Dataset 'VisDrone.yaml' images not found ⚠️, missing path '/content/datasets/VisDrone/VisDrone2019-DET-val/images'
Downloading https://github.com/ultralytics/yolov5/releases/download/v1.0/VisDrone2019-DET-val.zip to '/content/datasets/VisDrone/VisDrone2019-DET-val.zip'...
Downloading https://github.com/ultralytics/yolov5/releases/download/v1.0/VisDrone2019-DET-test-dev.zip to '/content/datasets/VisDrone/VisDrone2019-DET-test-dev.zip'...
Downloading https://github.com/ultralytics/yolov5/releases/download/v1.0/VisDrone2019-DET-test-challenge.zip to '/content/datasets/VisDrone/VisDrone2019-DET-test-challenge.zip'...
Downloading https://github.com/ultralytics/yolov5/releases/download/v1.0/VisDrone2019-DET-train.zip to '/content/datasets/VisDrone/VisDrone2019-DET-train.zip'...
Converting /content/datasets/VisDrone/VisDrone2019-DET-train: 6471it [00:40, 159.06it/s]
Converting /content/datasets/VisDrone/VisDrone2019-DET-val: 548it [00:05, 104.34it/s]
Converting /content/datasets/VisDrone/VisDrone2019-DET-test-dev: 1610it [00:08, 179.31it/s]
Dataset download success ✅ (87.2s), saved to /content/datasets
```

تعداد epochs انتخابی بسیار زیاد است و فرصت نمی شود بنابراین آن را ده برابر کوچک تر کردم.

در نهایت پاسخی که گرفتم نشان از Train شدن مدل می دهد:

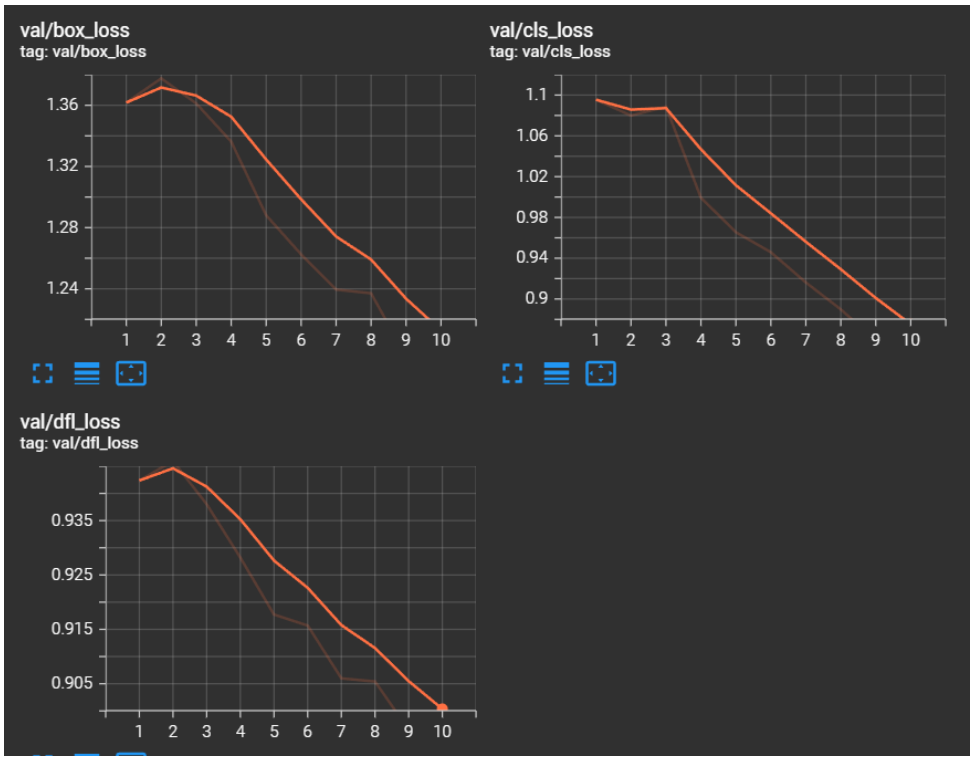
```
10 epochs completed in 1.002 hours.
Optimizer stripped from runs/detect/train3/weights/last.pt, 87.6MB
Optimizer stripped from runs/detect/train3/weights/best.pt, 87.6MB

Validating runs/detect/train3/weights/best.pt...
Ultralytics YOLOv8.1.43 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 268 layers, 43614318 parameters, 0 gradients, 164.9 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):  22% 4/18 [00:37<02:34, 11.04s/it]
```

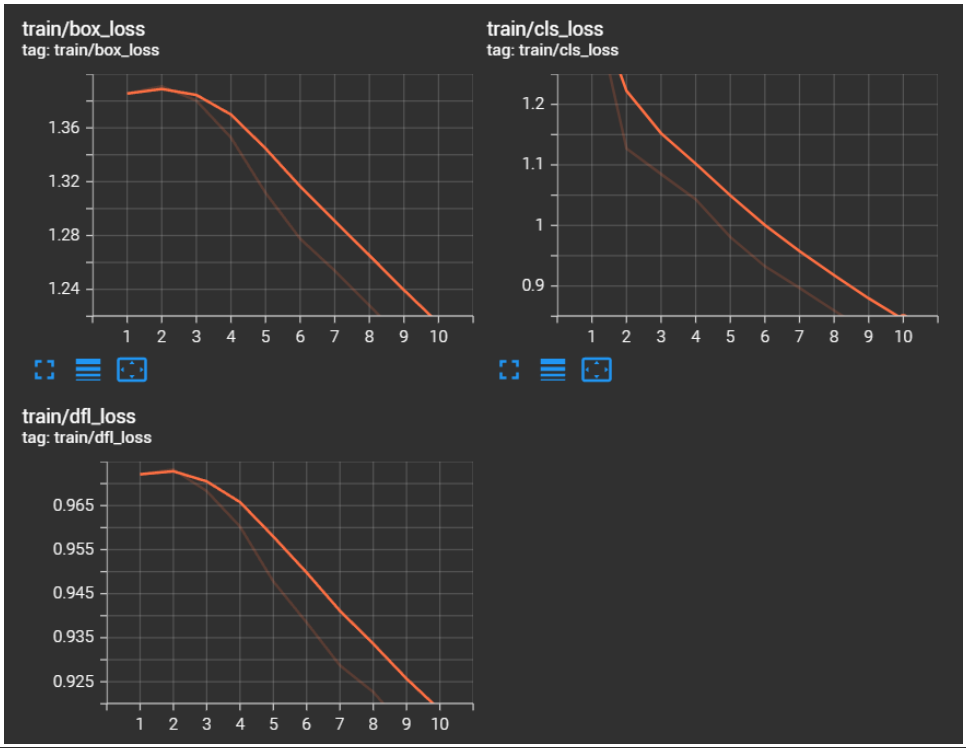
متاسفانه به دلیل تلاش های زیاد برای Train کردن بالاخره حجم رایگان به پایان رسید.

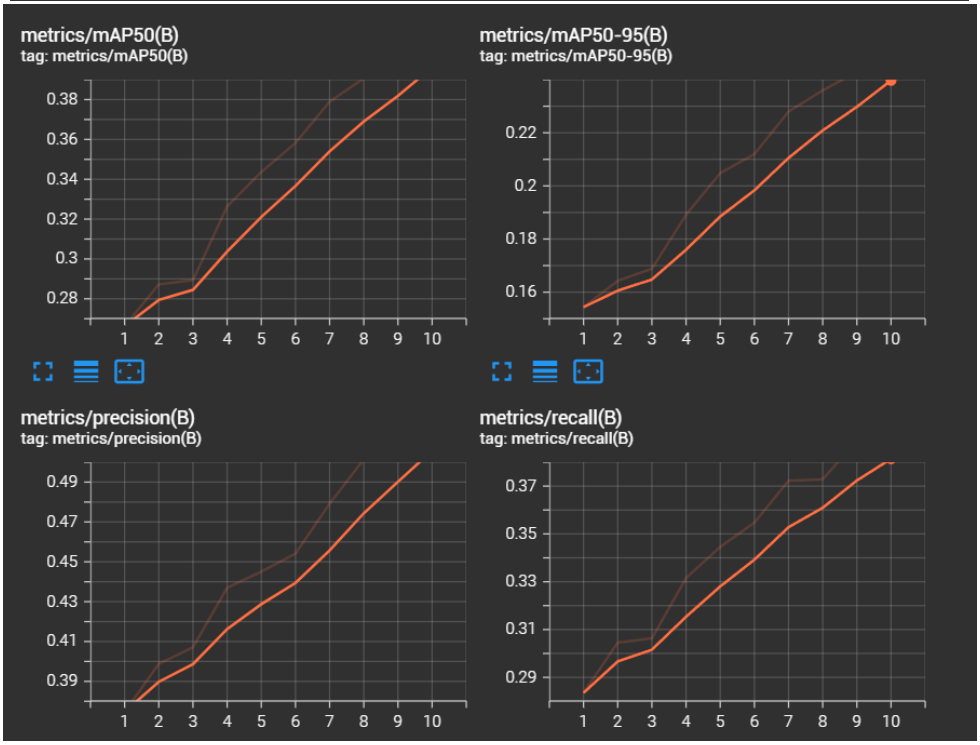


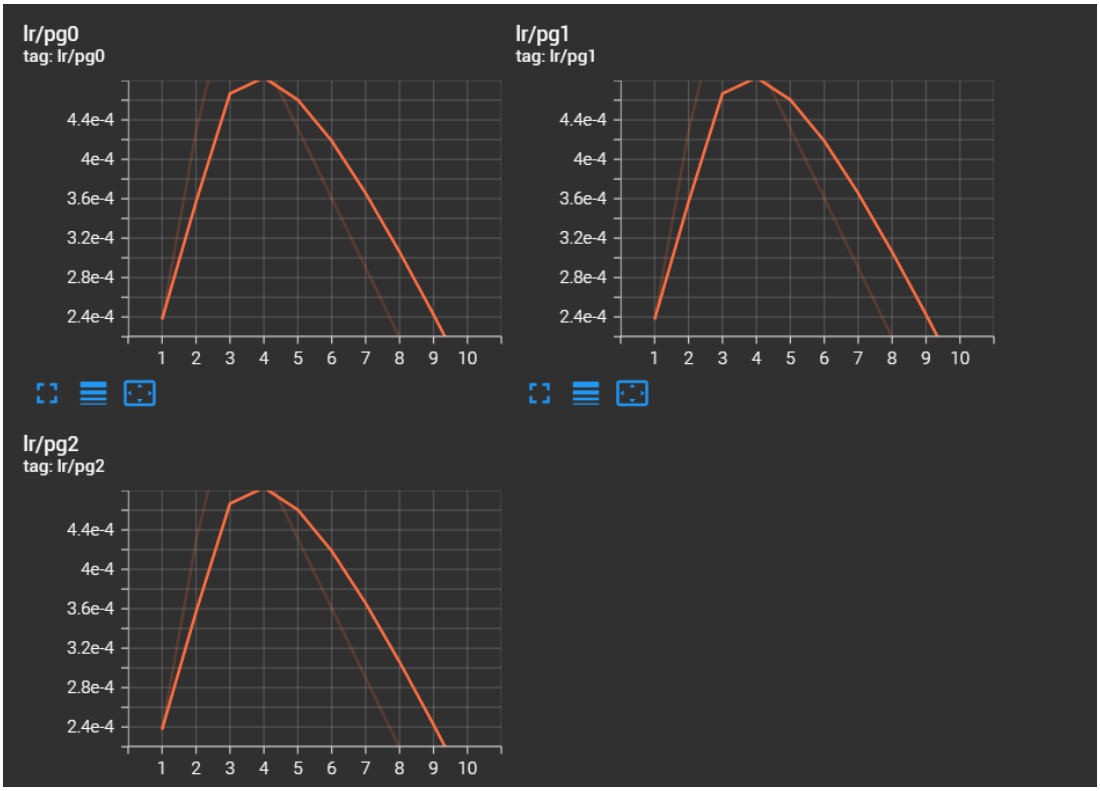
در نتیجه مجبور هستیم تا برای ادامهٔ همه ی مراحل را دوباره در یک اکانت دیگر انجام دهیم.

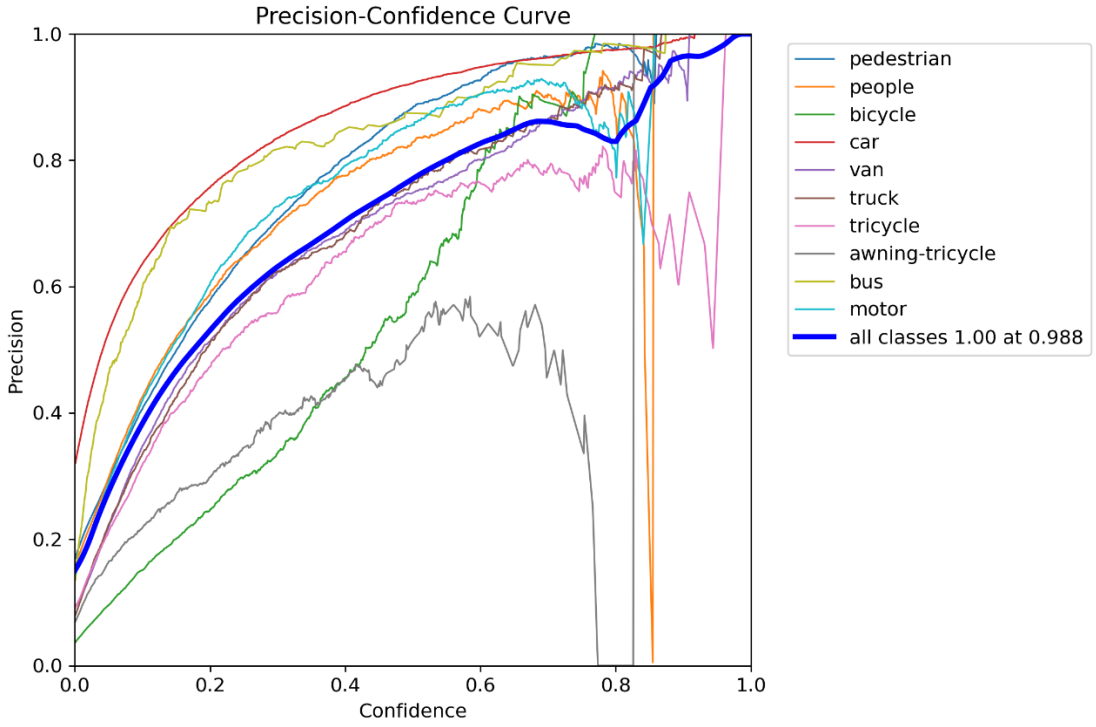
نمایش در TensorBoard

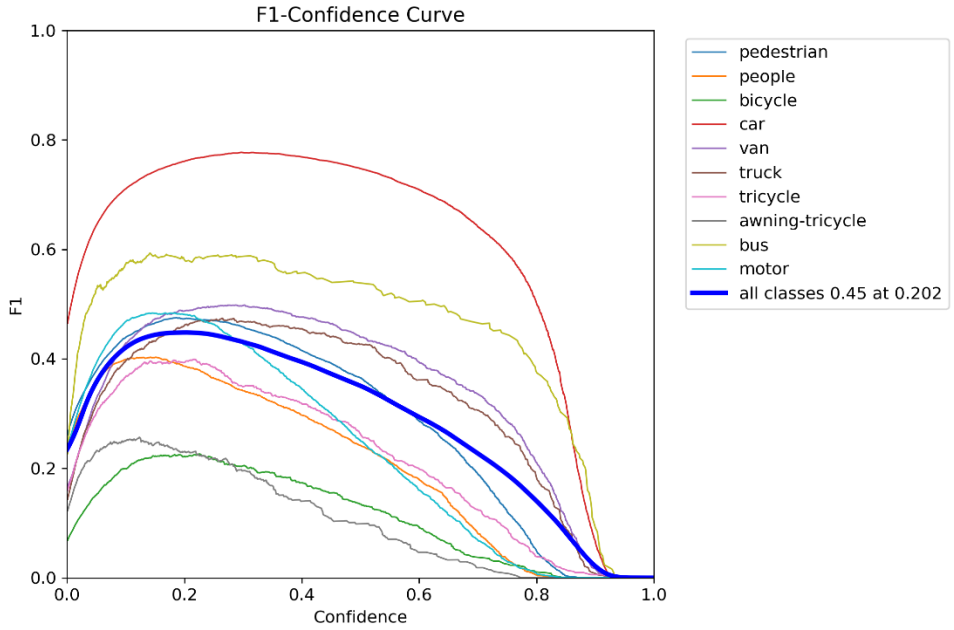


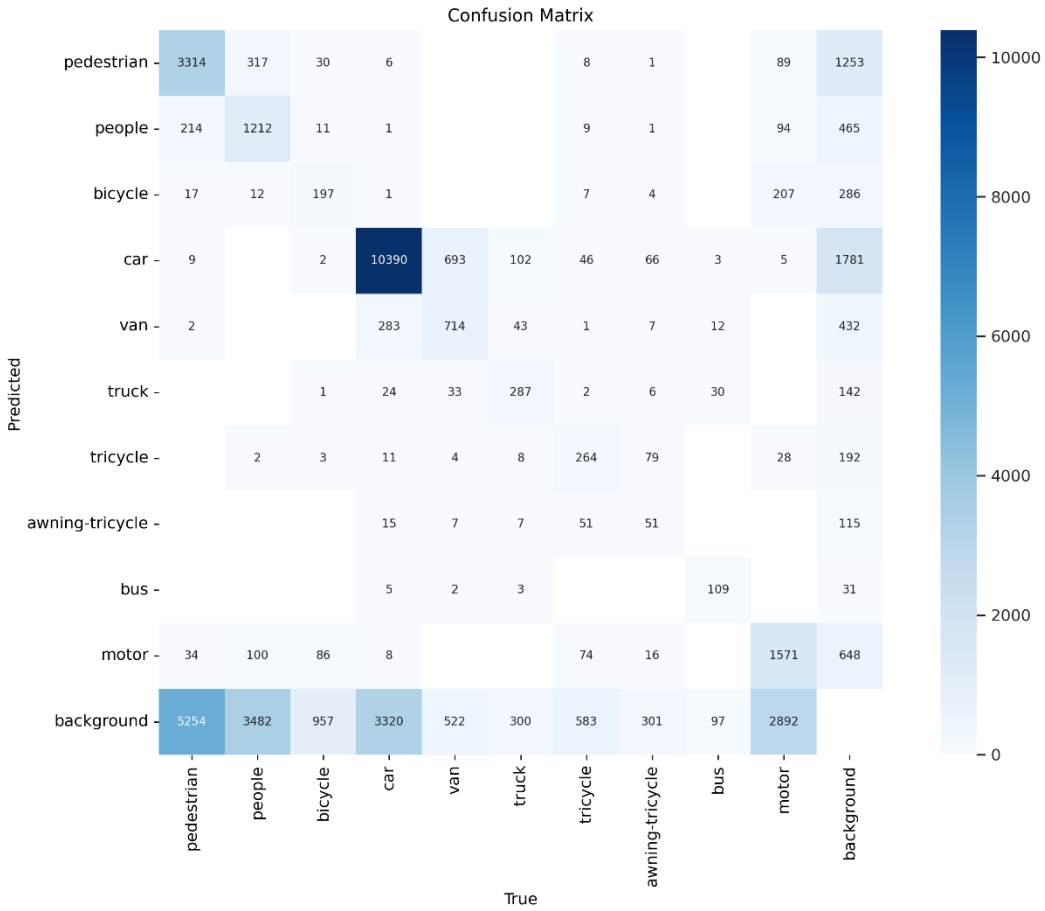

train/box_loss
tag: train/box_loss
1.36
1.32
1.28
1.24
1 2 3 4 5 6 7 8 9 10
train/cls_loss
tag: train/cls_loss
1.2
1.1
1
0.9
1 2 3 4 5 6 7 8 9 10
train/dfl_loss
tag: train/dfl_loss
0.965
0.955
0.945
0.935
0.925
1 2 3 4 5 6 7 8 9 10

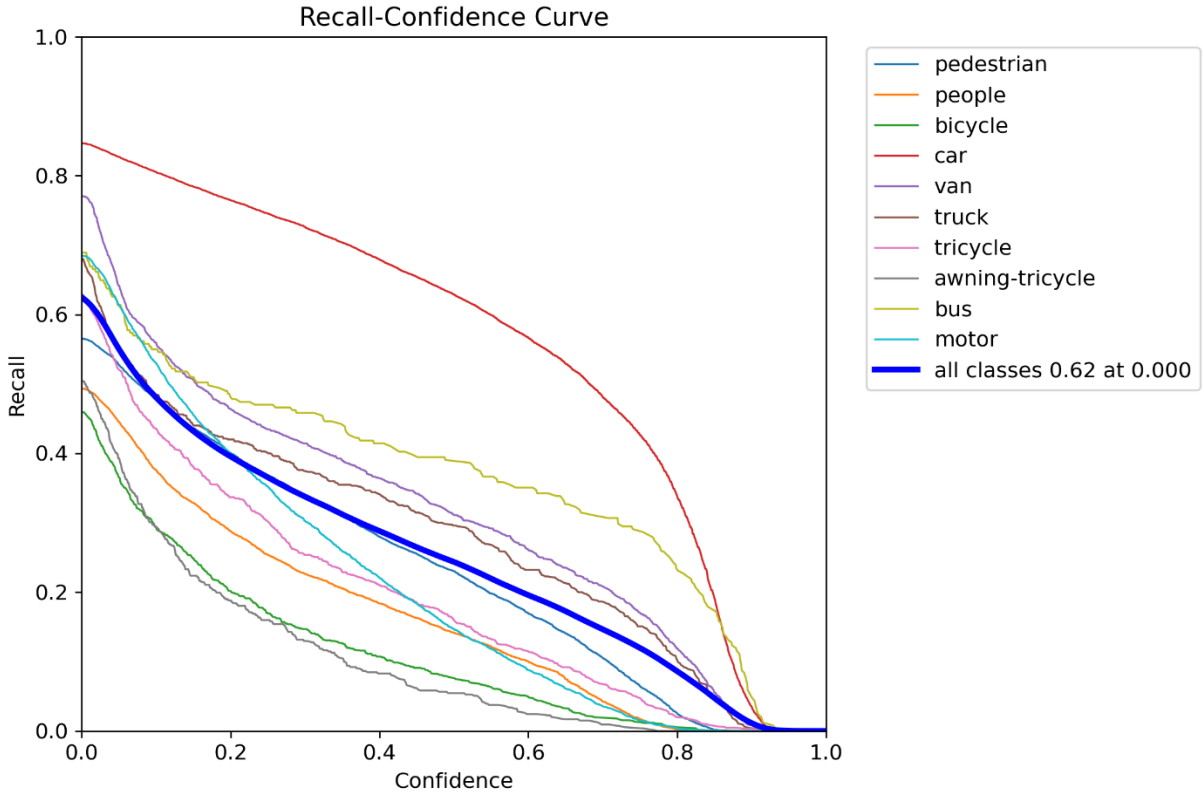

metrics/mAP50(B)
tag: metrics/mAP50(B)
0.38
0.36
0.34
0.32
0.3
0.28
1 2 3 4 5 6 7 8 9 10
metrics/mAP50-95(B)
tag: metrics/mAP50-95(B)
0.22
0.2
0.18
0.16
1 2 3 4 5 6 7 8 9 10
metrics/precision(B)
tag: metrics/precision(B)
0.49
0.47
0.45
0.43
0.41
0.39
1 2 3 4 5 6 7 8 9 10
metrics/recall(B)
tag: metrics/recall(B)
0.37
0.35
0.33
0.31
0.29
1 2 3 4 5 6 7 8 9 10

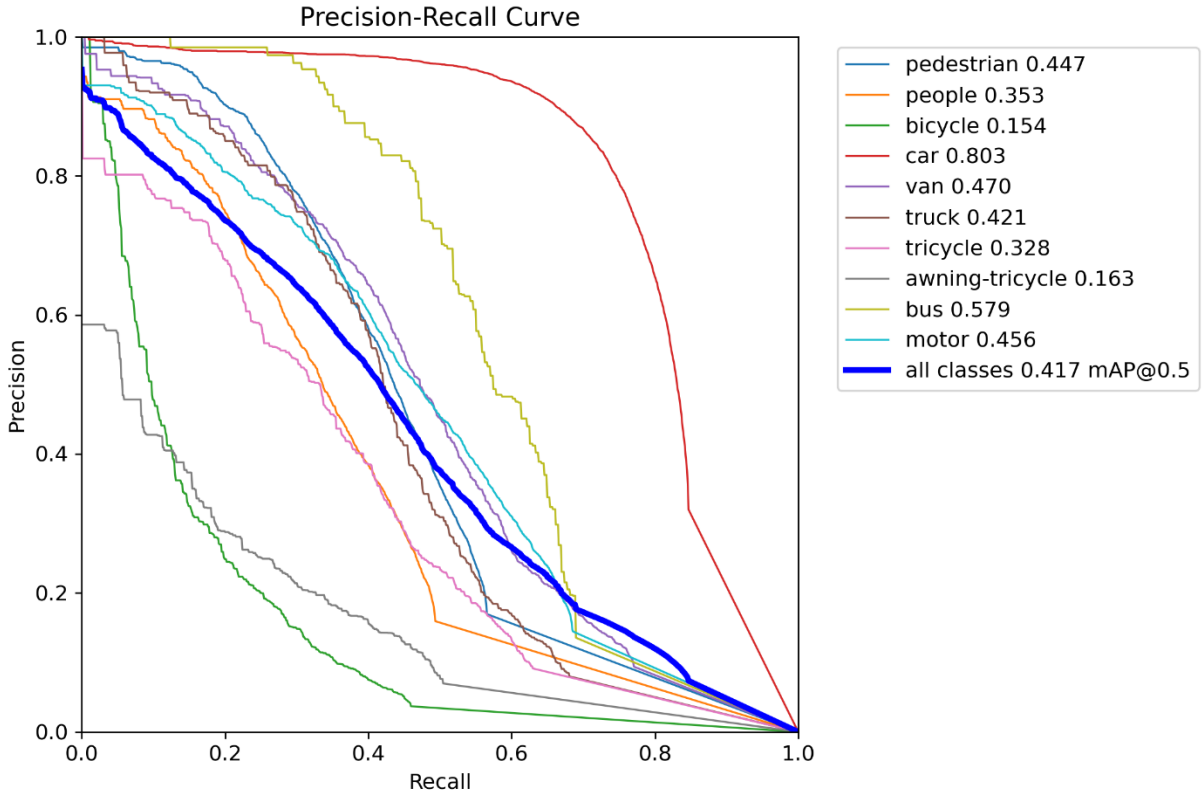

lr/pg0
tag: lr/pg0
4.4e-4
4e-4
3.6e-4
3.2e-4
2.8e-4
2.4e-4
1 2 3 4 5 6 7 8 9 10
lr/pg1
tag: lr/pg1
4.4e-4
4e-4
3.6e-4
3.2e-4
2.8e-4
2.4e-4
1 2 3 4 5 6 7 8 9 10
lr/pg2
tag: lr/pg2
4.4e-4
4e-4
3.6e-4
3.2e-4
2.8e-4
2.4e-4
1 2 3 4 5 6 7 8 9 10


Precision-Confidence Curve
Precision
Confidence
0.0
0.2
0.4
0.6
0.8
1.0
pedestrian
people
bicycle
car
van
truck
tricycle
awning-tricycle
bus
motor
all classes 1.00 at 0.988


F1-Confidence Curve
F1
Confidence
pedestrian
people
bicycle
car
van
truck
tricycle
awning-tricycle
bus
motor
all classes 0.45 at 0.202


Confusion Matrix
Predicted
True
pedestrian
people
bicycle
car
van
truck
tricycle
awning-tricycle
bus
motor
background


Recall-Confidence Curve
Recall
Confidence
0.0
0.2
0.4
0.6
0.8
1.0
pedestrian
people
bicycle
car
van
truck
tricycle
awning-tricycle
bus
motor
all classes 0.62 at 0.000


Precision-Recall Curve
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
pedestrian 0.447
people 0.353
bicycle 0.154
car 0.803
van 0.470
truck 0.421
tricycle 0.328
awning-tricycle 0.163
bus 0.579
motor 0.456
all classes 0.417 mAP@0.5

بررسی پیش بینی مدل

خب مدل کامل Train شد حالا برای Predict کردن کاملا آماده هستیم کافیست تا ویدیو مورد نظر از تصاویر هوایی پیداکرده تا بتوان از آن جهت نمونه ای برای پیش بینی استفاده کرد.

```
https://checker.in/go/8213681
```

حالا کافیست از دستور زیر استفاده کنیم تا پیش بینی را انجام دهیم:

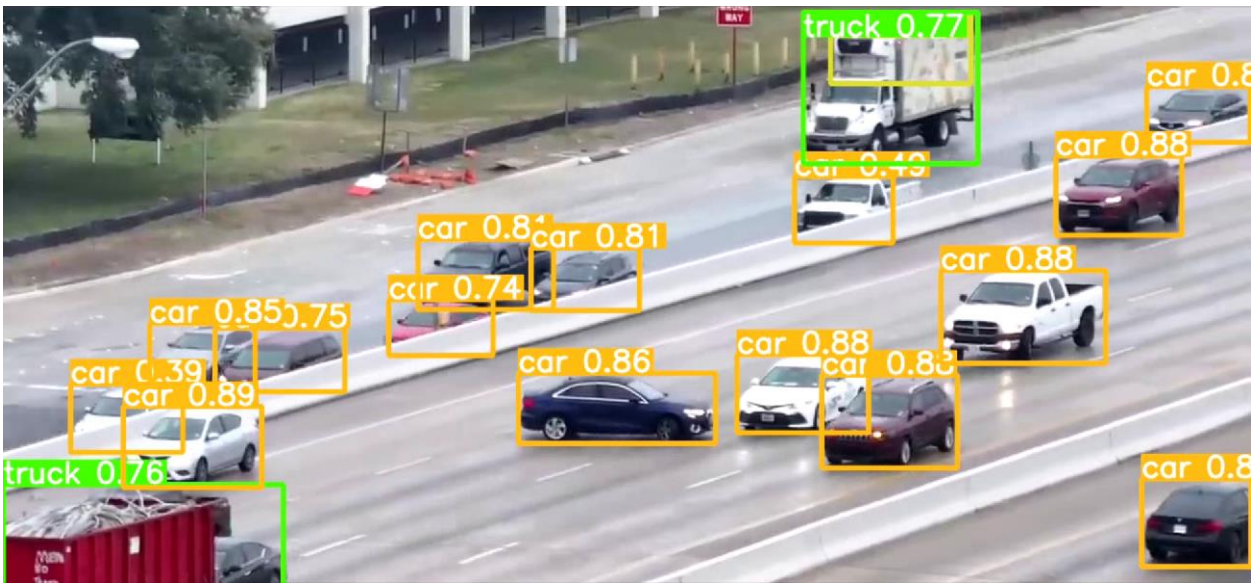
```
0: 384x640 1 pedestrian, 24 cars, 1 van, 4 trucks, 39.6ms
0: 384x640 1 pedestrian, 24 cars, 1 van, 3 trucks, 39.5ms
0: 384x640 1 pedestrian, 26 cars, 1 van, 3 trucks, 39.5ms
0: 384x640 2 pedestrians, 31 cars, 1 van, 6 trucks, 39.8ms
0: 384x640 2 pedestrians, 31 cars, 1 van, 3 trucks, 39.5ms
0: 384x640 2 pedestrians, 29 cars, 1 van, 3 trucks, 39.5ms
0: 384x640 1 pedestrian, 29 cars, 1 van, 4 trucks, 48.9ms
0: 384x640 1 pedestrian, 31 cars, 1 van, 3 trucks, 39.5ms
```

همانطور که مشاهده می شود پیش بینی شروع شده است و با توجه به اینکه تعداد فریم های ویدیو زیاد است در نتیجه مدت زمان زیادی را باید صبر کنیم تا این تعداد فریم پردازش شود و پیش بینی در هر فریم انجام شود.

```
0: 384x640 1 pedestrian, 31 cars, 1 van, 3 trucks, 39.5ms
Speed: 2.8ms preprocess, 39.3ms inference, 10.0ms postprocess per image at shape
Results saved to runs/detect/predict3
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

در نهایت نتیجه پیش بینی بدست آمد و مشاهده می شود که پیش بینی به صورت بسیار خوبی انجام شده در نتیجه مدل ما به خوبی برای تشخیص اشیاء Train شده بود.

بخشی از تصویر بدست آمده به عنوان نتیجهٔ پیش بینی:


truck 0.77
car 0.88
car 0.88
car 0.81
car 0.74
car 0.85
car 0.89
car 0.86
car 0.88
truck 0.76