



دانشکده مهندسی برق

استخراج و آنالیز هوشمند داده

گزارش تمرین دوم

نویسنده ها:

املین غازاریان 9923056

انوشا شریعتی 9923041

استاد درس:

دکتر شریفیان

بهار 1403

فهرست:

-1 لينك گوگل کولب برای سه بخش تمرین

-2 بخش اول

-3 بخش دوم

-4 بخش سوم

لینک گوگل کولب برای سه بخش تمرین

بخش اول:

<https://colab.research.google.com/drive/1tJX2JwKp53PJmfLNFxq2bWSak001Gmuh?usp=sharing>

بخش دوم:

<https://colab.research.google.com/drive/1l8JlobOLGg2PYKx0vLKnWgFFQxfWS4yS?usp=sharing>

بخش سوم:

قسمت اول و دوم: yolov5

https://colab.research.google.com/drive/1HQVUW06gLaQKVmomz-CHwV_-tqW4TGbi?usp=sharing

قسمت سوم: YOLOV8 IMPROVED

<https://colab.research.google.com/drive/1lRs5p7t5lKpvogjw-pfCbiYuJPhyA3Ji?usp=sharing>

بخش اول:

<https://colab.research.google.com/drive/1tJX2JwKp53PJmfLNFXq2bWSak001Gmuh?usp=sharing>

با استفاده از کد گیت داده شده در فایل بروزه 7 yolov را یکبار به تنهایی آموزش می دهیم.

```
!python train.py --batch 2 --cfg  
/content/drive/MyDrive/yolov7/cfg/training/yolov7.yaml --epochs 40 --data  
/content/drive/MyDrive/yolov7/sat-1/data.yaml --weights 'yolov7.pt' --  
device 0
```

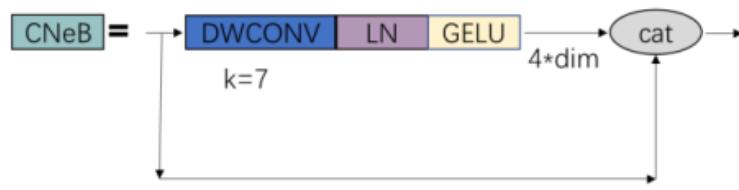
سپس در مراحل بعدی مدل های بهبود یافته را به آن اضافه می کنیم.

مدل اول:

1_yolov7-transBoT3-ConvNext-attention.yaml

Modules: CNeB, BoT3, CBAM

: **cneb** ماژول



CNeB از هسته بزرگ 7×7 DWCONV برای جایگزینی CNN معمولی استفاده می کند. اگر چه لایه BN می تواند بهبود همگرایی و کاهش بیش از برازش شود، هنوز هم دارد پیچیدگی های بسیاری است که در عملکرد مدل روی تأثیر منفی می گذارد، بنابراین استفاده می شود به جای BN. در نهایت، جایگزینی ReLU با GELU باعث می شود flop ها در حین train به G2.66 کاهش می یابند و در عین حال دقت بهبود می یابند.

add the codes for the CNeB module:

```
class CNeB(nn.Module):
```

```
# CSP ConvNextBlock with 3 convolutions by iscyy/yoloair
```

```

def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5): # ch_in, ch_out, number,
shortcut, groups, expansion

    super().__init__()

    c_ = int(c2 * e) # hidden channels

    self.cv1 = Conv(c1, c_, 1, 1)

    self.cv2 = Conv(c1, c_, 1, 1)

    self.cv3 = Conv(2 * c_, c2, 1)

    self.m = nn.Sequential(*(ConvNextBlock(c_) for _ in range(n)))

def forward(self, x):

    return self.cv3(torch.cat((self.m(self.cv1(x)), self.cv2(x)), dim=1))

```

: منبع

<https://alife-robotics.co.jp/members2023/icarob/data/html/data/OS/OS31/OS31-3.pdf>

An improved network for pedestrian-vehicle detection based on YOLOv7

ماژول BoT3

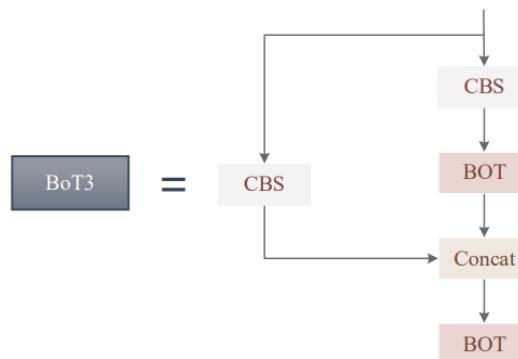


Fig.4. Structural diagram of BoT3

BoT3 هم رمزگذار و هم رمزگشنا را ساده می کند اجزای مدل قابلیت نگهداری مدل و مقیاس پذیری با ترکیب تکنیک های تطبیقی افزایش می یابد. توابع مدولار ماژول BoT3 که استفاده می شود عمدتاً شامل بلوک های Convolutional و Bottleneck transforms (BoT) همانطور که در شکل بالا می توان مشاهده کرد.

object classification و Bottleneck Transformer مخفف BoT است که برای detection شده است. توجه به خود چند سر یک pivotal component مدل BoT شده است. در شکل زیر نشان داده شده است. BoT شامل سه جزء اساسی است: contraction، expansion و MHSA.

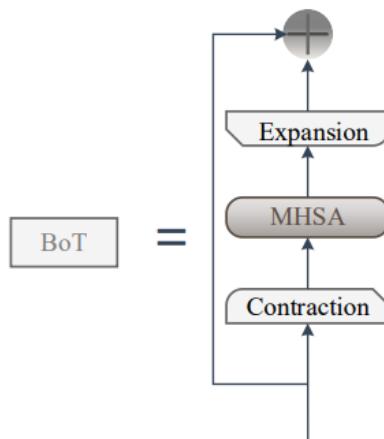


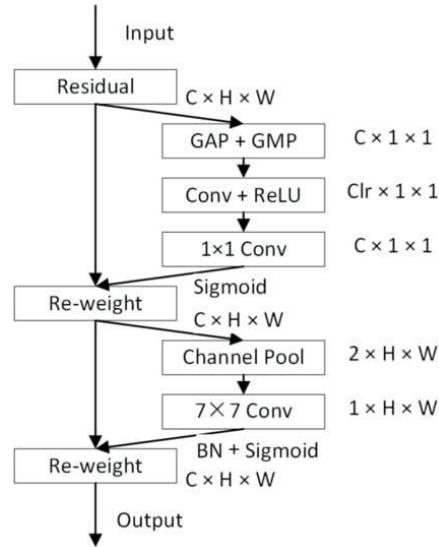
Fig.5. BoT structure diagram

منبع:

https://www.iaeng.org/IJCS/issues_v51/issue_3/IJCS_51_3_17.pdf

Surface Defect Detection Algorithm for Strip Steel Based on Improved YOLOv7 Model

: CBAM مژول



باعث می شود که feature map میانی ورودی روی ویژگی های مهم تمرکز کند و ویژگی های غیر ضروری را سرکوب کند و باعث کاهش نویز در هم تنیدگی نامریبوط شود، که در نهایت باعث می شود شبکه به طور صحیح تری روی شی تمرکز کند.

منبع:

<https://www.mdpi.com/2076-3417/13/16/9173>

YOLOv7 Optimization Model Based on Attention Mechanism Applied in Dense Scenes

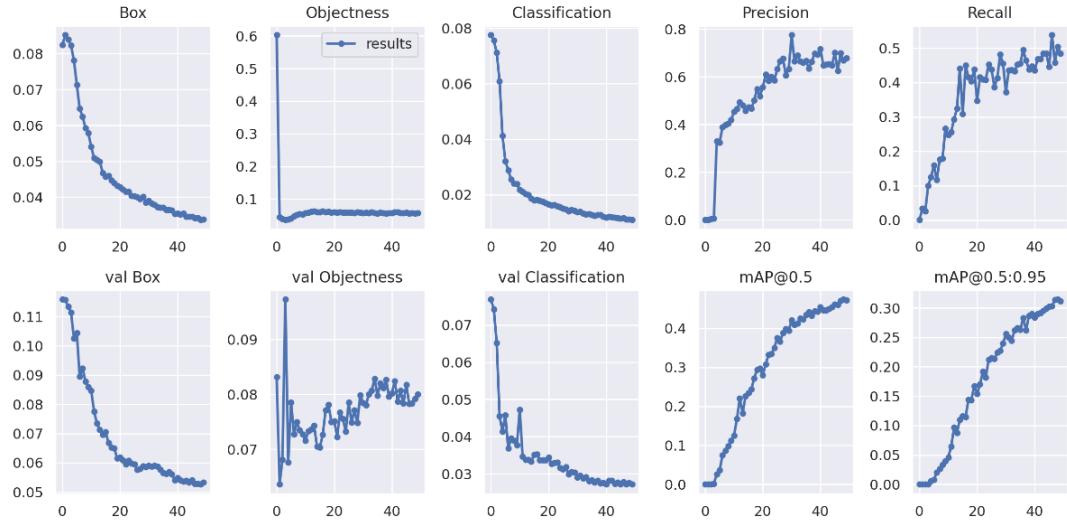
حال **train** کردن yolov7 با مدل بهبود یافته:

```
!python train.py --batch 16 --cfg
/content/drive/MyDrive/yolov7/cfg/training/yolov7-transBoT3-ConvNext-
attention.yaml --epochs 50 --data /content/drive/MyDrive/yolov7/sat-
1/data.yaml --weights 'yolov7.pt' --device 0
```

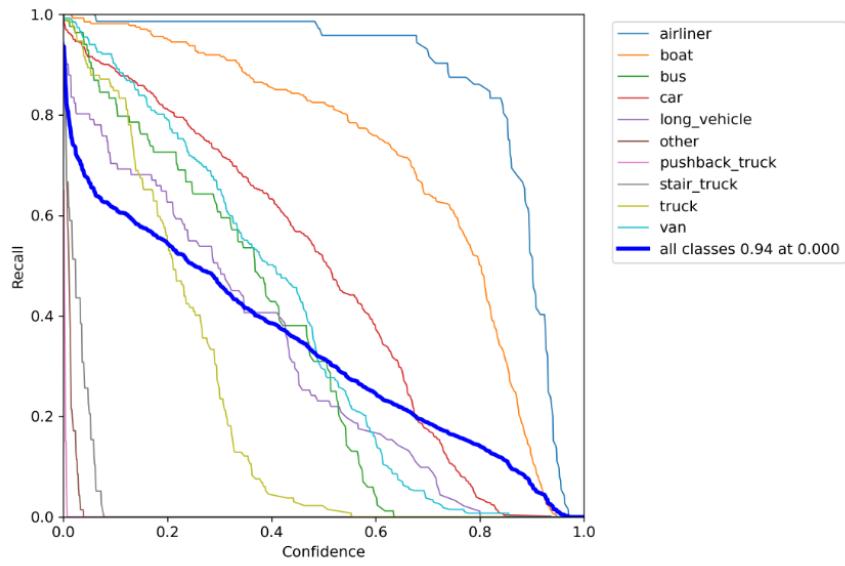
بعد از اضافه کردن مازول های مورد نظر در فایل common.py و اسم مازول در yolo.py مدل را train می کنیم. طبیعی است قرار است به ارور هایی مثل: module is missing x بخوریم که با سرچ کردن در گیت هاب yoloair می توان کد آن را پیدا کرد و اضافه کرد.

: **train** بعد از نتایج

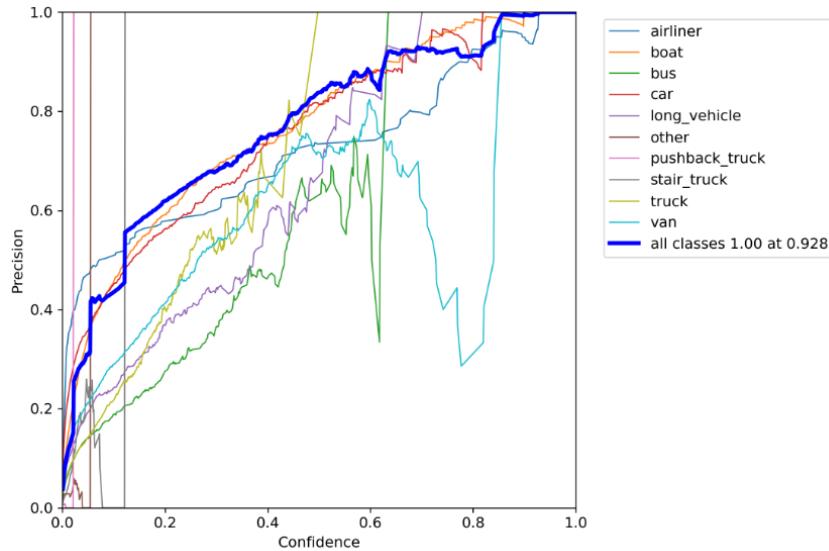
: results فایل



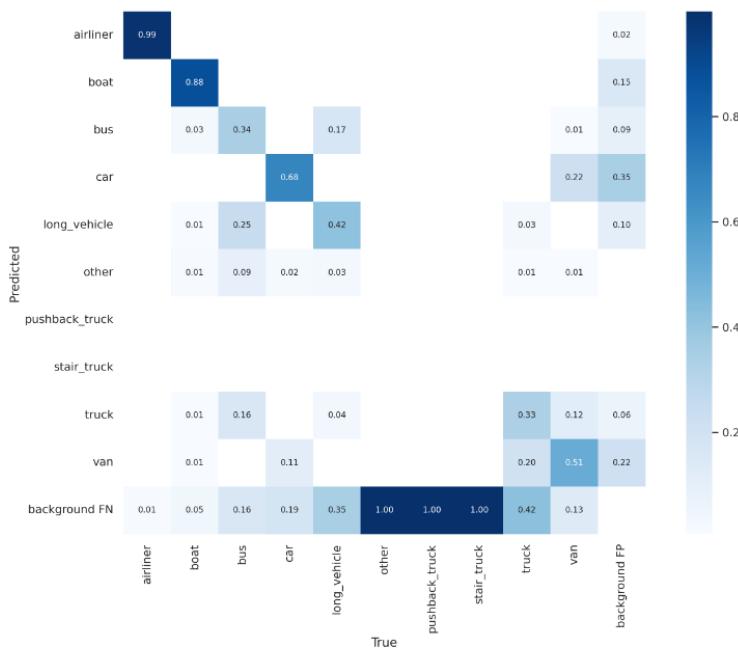
: R_Curve فایل



: P_curve فایل



: confusion ماتریس



سپس فایل detect.py را اجرا می کنیم:

```
!python detect.py --weights  
/content/drive/MyDrive/yolov7/runs/train/exp53/weights/best.pt --source  
/content/drive/MyDrive/yolov7/sat-1/test/images
```

نمونه:



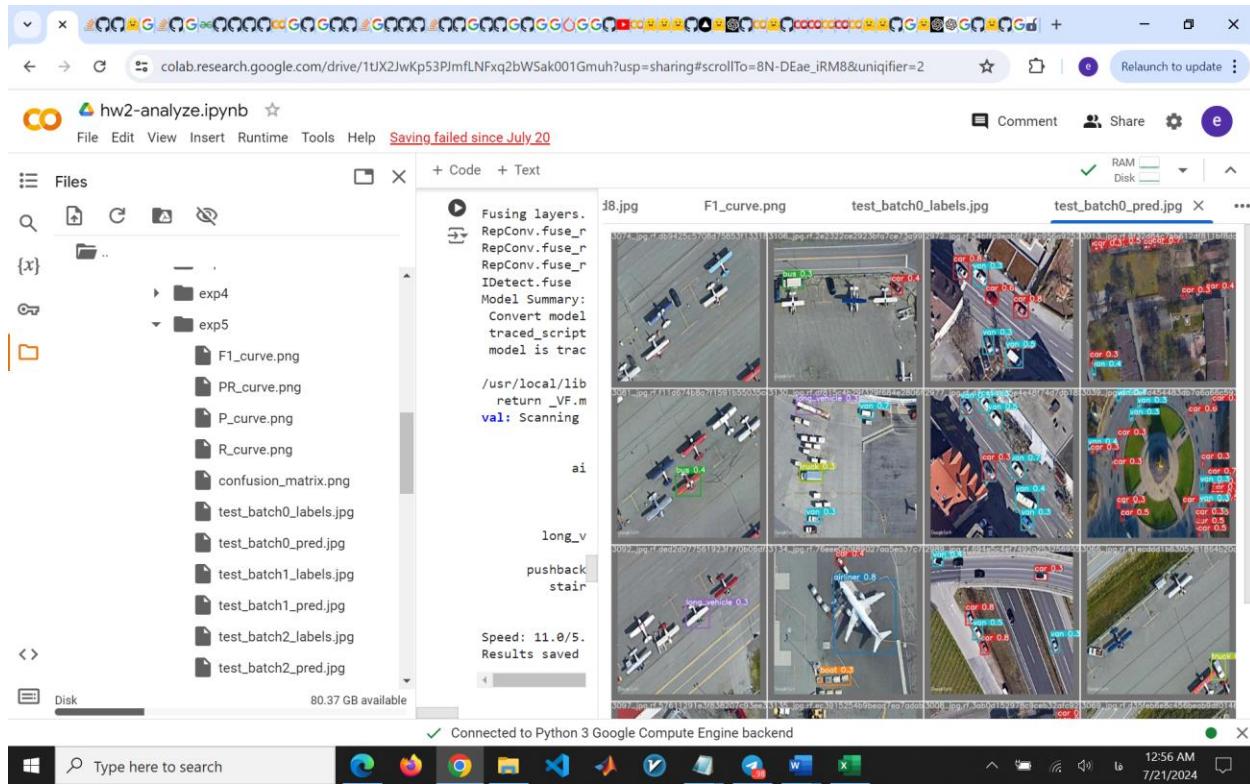
فایل **test.py** را اجرا می کنیم تا نتایج را مشخص کنیم:

```
!python test.py --data /content/drive/MyDrive/yolov7/sat-1/data.yaml --
batch 16 --conf 0.001 --iou 0.65 --img 640 --device 0 --weights
/content/drive/MyDrive/yolov7/runs/train/exp53/weights/best.pt
```

نتایج:

```
/usr/local/lib/python3.10/dist-packages/torch/functional.py:512: UserWarning: torch.meshgrid: in
    return _VF.meshgrid(tensors, **kwargs) # type: ignore[attr-defined]
val: Scanning 'sat-1/valid/labels.cache' images and labels... 272 found, 0 missing, 28 empty, 0
      Class      Images      Labels         P          R      mAP@.5  mAP@.5: .95: 10
        all       272       1572     0.661      0.504      0.471      0.312
      airliner    272        72      0.66      0.986      0.924      0.741
        boat       272       274     0.688      0.872      0.864      0.482
        bus        272        84     0.288      0.655      0.422      0.272
        car        272       523     0.588      0.795      0.758      0.479
      long_vehicle 272        91     0.389      0.604      0.533      0.329
      other       272        70        1          0      0.0309      0.0164
      pushback_truck 272        20        1          0      0.00633      0.00321
      stair_truck   272        39        1          0      0.139      0.0757
        truck       272       132     0.555      0.341      0.423      0.294
        van         272       267     0.446      0.79      0.607      0.427
Speed: 11.0/5.7/16.7 ms inference/NMS/total per 640x640 image at batch-size 16
Results saved to runs/test/exp5
```

نمونه:



مدل دوم:

2-yolov7-C3C2-SEAttention.yaml

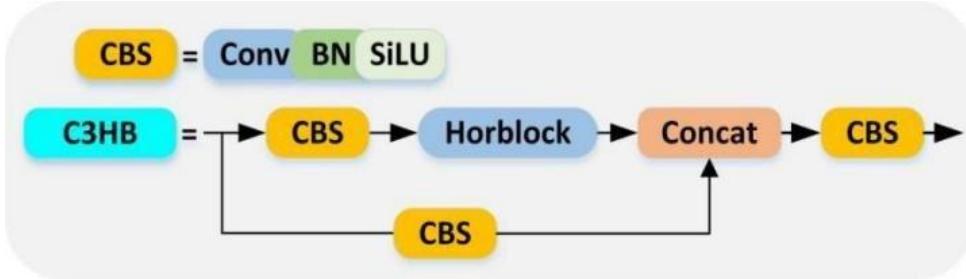
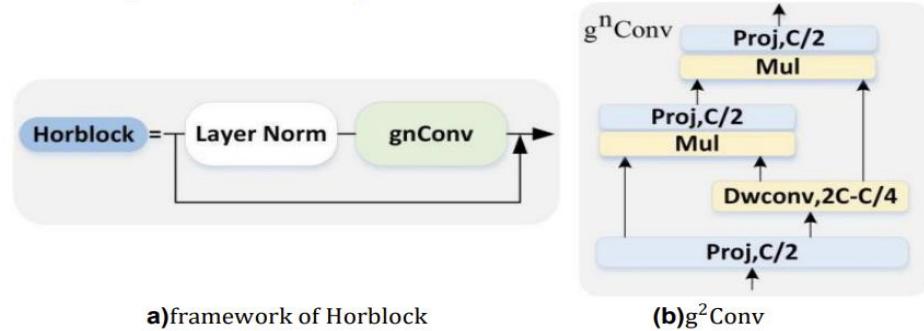
Modules: C3HB, C3STR, SEAttention

این مژول ارور آن رفع نمی شد و با هماهنگی با تدریسیار یک مژول دیگر به دلخواه انتخاب شد

ولی مژول های آن توضیح داده شده:

مژول : C3HB

مزایای HorNet (RecursiveGatedConvolutions) و VisionTransformer را ترکیب می کند. با طراحی بازگشتی، تعاملات اطلاعات کارآمد، مقیاس پذیرتر و با مرتبه بالاتر را پشنهداد می دهد و همچنین از طریق طراحی بازگشتی، تعاملات اطلاعات مکانی مرتبه بالاتر را کارآمد، مقیاس پذیر و متحرک انجام می دهد. همانطور که در شکل پایین نشان داده شده است ساختار gated convolution است که با استفاده از کانولوشن استاندارد، ضرب عنصری و طرح خطی تشکیل شده است، اما با یک تابع adaptive mixing شبیه به self-attentiveness است. عملیات ابتدا تعداد کانال های ویژگی را تنظیم می کند (توسط دو لایه کانولوشن)، و ثانیاً ویژگی های خروجی کانولوشن depth-separable را به چند multiple blocks می کند، که هر کدام با ویژگی های بلوک قبلی با ضرب عنصر به عنصر تعامل می کنند تا در نهایت ویژگی های خروجی را به دست آورند. با الهام از HorNet، مژول C3HB طراحی شده



ماژول C3HB عمدتاً از ماژول CBS، ماژول Horblock و ساختار Concat تشکیل شده است. ویژگی اولی ورودی اطلاعات به دو شاخه اختصاص داده می شود. در مرحله اول، تعداد کانال های اطلاعات ویژگی ورودی نصف شده و استخراج ویژگی از طریق CBS انجام می شود در شاخه یک و قسمت دیگر ویژگی اطلاعات از طریق ماژول Horblock و CBS منتقل می شود در شاخه دو، و اطلاعات ویژگی خروجی از شاخه های یک و دو با استفاده از Concat متصل می شوند. در نهایت، بهبود ویژگی انجام می شود توسط یک ساختار CBS دیگر، که در آن اندازه ویژگی خروجی همان اندازه ورودی C3HB است.

```
class C3HB(nn.Module):
```

```
# CSP HorBlock with 3 convolutions by iscyy/yoloair

def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5): # ch_in, ch_out, number, shortcut, groups, expansion

    super().__init__()

    c_ = int(c2 * e) # hidden channels

    self.cv1 = Conv(c1, c_, 1, 1)

    self.cv2 = Conv(c1, c_, 1, 1)

    self.cv3 = Conv(2 * c_, c2, 1)

    self.m = nn.Sequential(*(HorBlock(c_) for _ in range(n)))
```

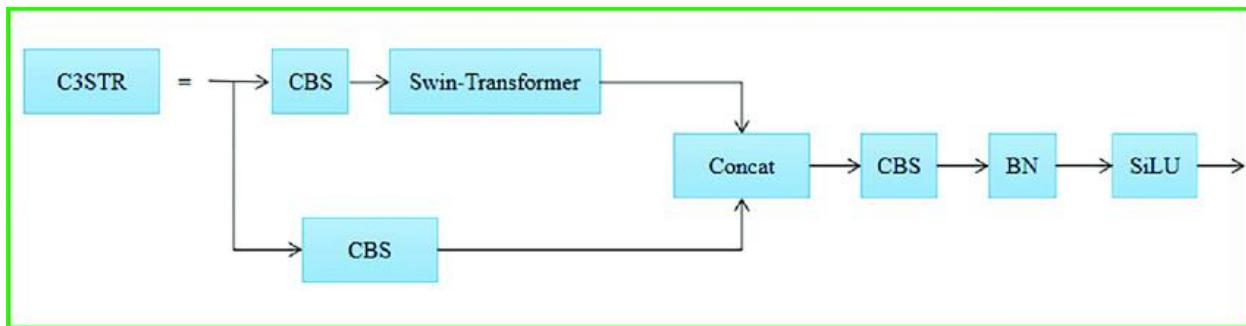
```
def forward(self, x):
```

```
return self.cv3(torch.cat((self.m(self.cv1(x)), self.cv2(x)), dim=1))
```

: ماژول **seattention**

توجه فشرده سازی و تحریک (SE) را اجرا می کند، مکانیزمی که برای بهبود قدرت بازنمایی یک شبکه با مدل سازی صریح وابستگی های متقابل بین کanal های ویژگی های کانولوشنی آن طراحی شده است.

: ماژول **C3STR**



ماژول C3STR ناحیه محاسباتی را در هر پنجره با تقسیم کردن کنترل می کند پنجره محلی برای دستیابی به تعامل اطلاعات متقابل پنجره، کاهش طول دنباله و پیچیدگی محاسباتی در مقایسه با خودتوجی چند سر در ترانسفورماتور conventional.

منبع:

Swin-Transformer -YOLOv5 for lightweight hot-rolled steel strips surface defect detection algorithm

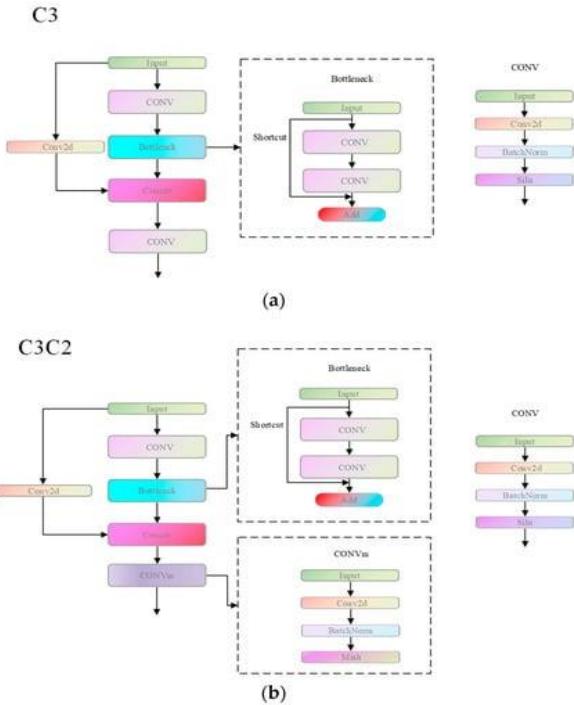
https://www.researchgate.net/publication/377699494_Swin-Transformer_-YOLOv5_for_lightweight_hot-rolled_steel_strips_surface_defect_detection_algorithm

مدل دوم: (انتخاب دلخواه)

2_yolov7-C3C2-CA.yaml

Modules: CNeB ,C3C2

: ماژول **C3C2**



ماژول کانولوشن C3C2 از ساختار شبکه Swin Transformer الهام گرفته شده است. این بر اساس ماژول اصلی C3 است، مایل به تغییر می کند به یک ساختار residual branch convolution (BN) ساده. لایه (BN) و لایه تابع فعال Silu را حذف می کند تا میزان محاسبه پارامتر کاهش یابد. با توجه به اینکه تابع فعال سازی میش نسبت به تابع فعال سازی Silu توانایی بهتری برای سرکوب بیشبرازش دارد، تابع فعال سازی میش نسبت به فرایامترهای مختلف قوی تر است.

منبع:

<https://www.mdpi.com/2079-6412/13/3/536>

Development of an Improved YOLOv7-Based Model for Detecting Defects on Strip Steel Surfaces

حال **yolov7** train کردن با مدل بهبود یافته:

```
!python train.py --batch 16 --cfg
/content/drive/MyDrive/yolov7/cfg/training/yolov7-C3C2-CA.yaml --epochs
100 --data /content/drive/MyDrive/yolov7/sat-1/data.yaml --weights
'yolov7.pt' --device 0
```

:1 رور

AttributeError: module 'numpy' has no attribute 'int'.

`np.int` was a deprecated alias for the builtin `int`. To avoid this error in existing code, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `int64` or `int32` to specify the precision. If you wish to review your current use, check the release note link for additional information. The aliases was originally deprecated in NumPy 1.20; for more details and guidance see the original release note at: <https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations>. Did you mean: 'inf'?

که برای رفع این ارور دو راه وجود داشت:

- ورژن numpy را پایین تر بردن

- جیگذاری int32 int64 با np.int یا

گزینه اول به انواع مختلفی انجام شد ولی ارور رفع نشد. در نهایت گزینه دوم جواب داد، که به این صورت است:

```
# Modify the train.py script to add np.int = np.int32
import os

# Path to the train.py script
train_script_path = '/content/drive/MyDrive/yolov7/train.py'

# Read the current content of the train.py file
with open(train_script_path, 'r') as file:
    content = file.readlines()

# Insert np.int = np.int32 at the beginning
content.insert(0, 'import numpy as np\nnp.int = np.int32\n')

# Write the modified content back to the train.py file
with open(train_script_path, 'w') as file:
    file.writelines(content)

print("Modification complete.")
```

:2 ارور

RuntimeError: indices should be either on cpu or on the same device as the indexed tensor (cpu)

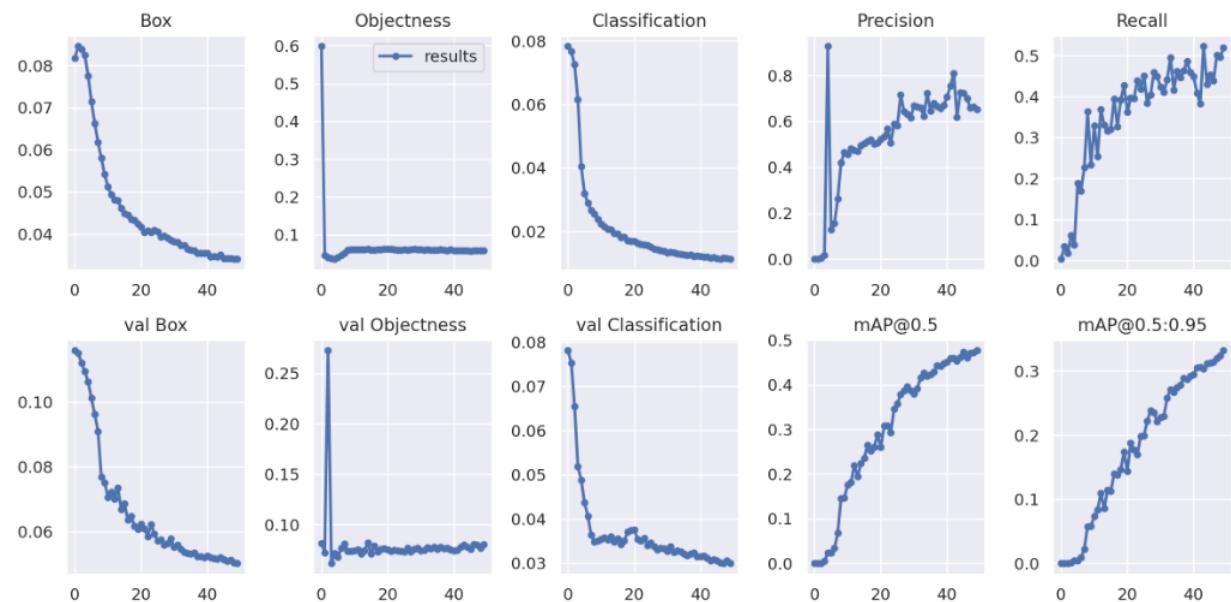
که برای رفع این ارور از لینک زیر کمک گرفته شده:

Indices should be either on CPU or on the same device as the indexed tensor:
<https://stackoverflow.com/questions/74372636/indices-should-be-either-on-cpu-or-on-the-same-device-as-the-indexed-tensor>

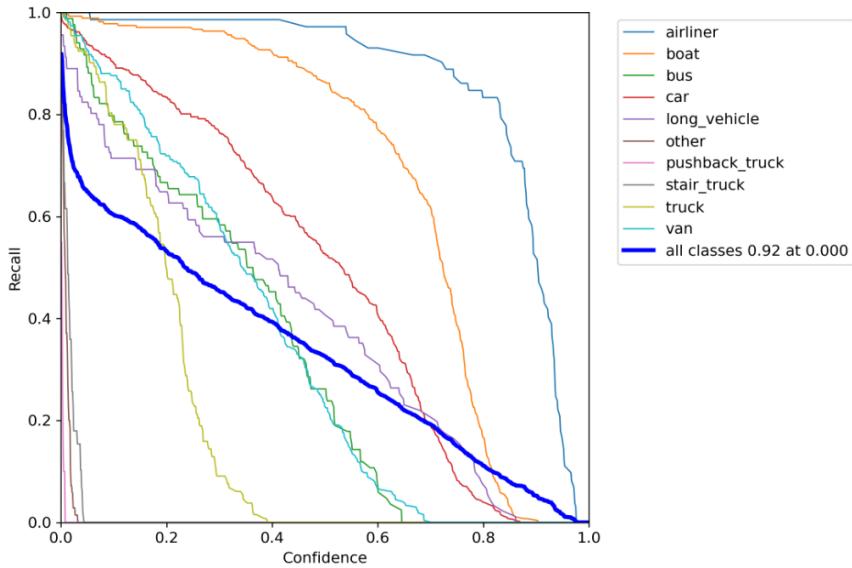
:train نتایج بعد از

Epoch	gpu_mem	box	obj	cls	total	labels	img_size
49/49	12.9G	0.03409	0.05727	0.01127	0.1026	42	640: 100% 105/105 [02: mAP@.5: .95: 100%
	Class	Images	Labels		P	R	
	all	272	1572	0.651	0.519	0.477	0.332
	airliner	272	72	0.728	0.986	0.951	0.783
	boat	272	274	0.67	0.971	0.922	0.552
	bus	272	84	0.248	0.655	0.462	0.333
	car	272	523	0.564	0.811	0.757	0.479
	long_vehicle	272	91	0.405	0.626	0.605	0.425
	other	272	70	1	0	0.0209	0.0125
	pushback_truck	272	20	1	0	0.00531	0.00318
	stair_truck	272	39	1	0	0.0965	0.0644
	truck	272	132	0.421	0.435	0.356	0.252
	van	272	267	0.479	0.704	0.593	0.417
39 epochs completed in 1.710 hours.							

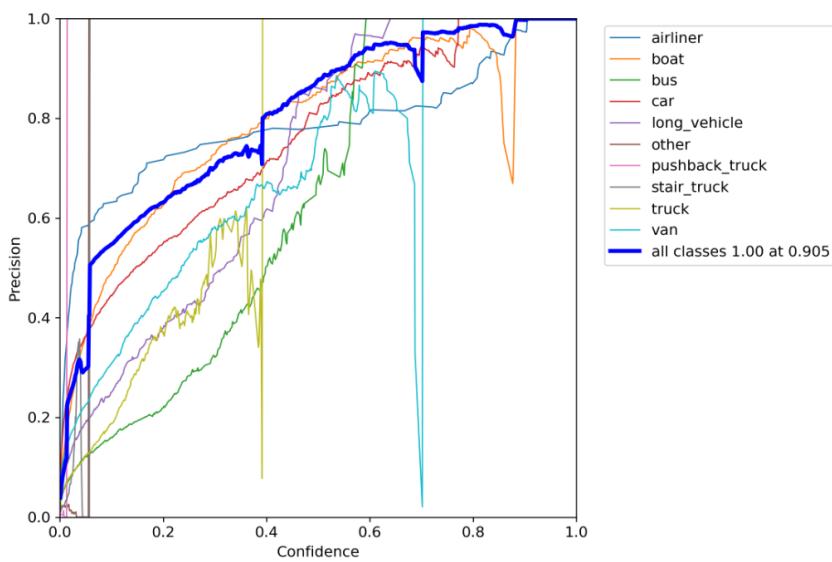
:results فایل



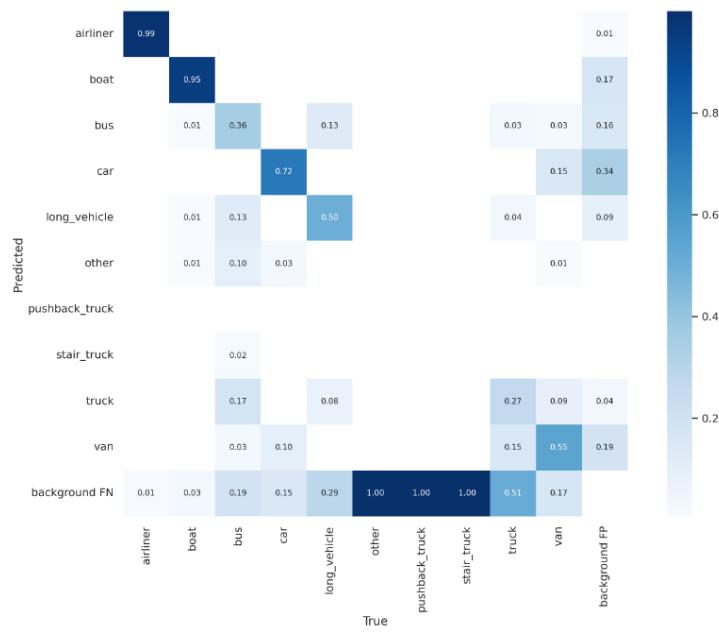
:R_curve فایل



:P_curve فایل



:confusion ماتریس



سبس فایل detect.py را اجرا می کنیم:

```
!python detect.py --weights  
/content/drive/MyDrive/yolov7/runs/train/exp43/weights/best.pt --source  
/content/drive/MyDrive/yolov7/sat-1/test/images
```

:نمونه



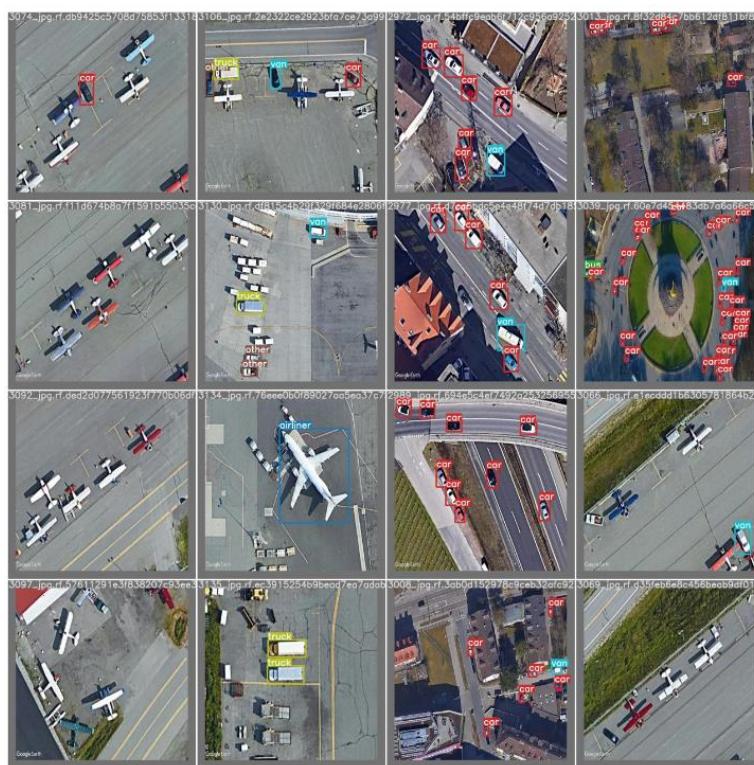
فایل **test.py** را اجرا می کنیم تا نتایج را مشخص کنیم:

```
!python test.py --data /content/drive/MyDrive/yolov7/sat-1/data.yaml --  
batch 16 --conf 0.001 --iou 0.65 --img 640 --device 0 --weights  
/content/drive/MyDrive/yolov7/runs/train/exp43/weights/best.pt
```

نتایج:

```
/usr/local/lib/python3.10/dist-packages/torch/functional.py:512: UserWarning: torch.meshgrid: in an  
    return _VF.meshgrid(tensors, **kwargs) # type: ignore[attr-defined]  
val: Scanning 'sat-1/valid/labels.cache' images and labels... 272 found, 0 missing, 28 empty, 0 corr  
      Class   Images   Labels      P      R    mAP@.5  mAP@.5:.95: 100% 1  
        all     272     1572  0.645  0.519  0.476  0.331  
    airliner     272       72  0.723  0.986  0.952  0.783  
      boat     272     274  0.618  0.971  0.913  0.547  
      bus     272      84  0.248  0.655  0.462  0.333  
      car     272     523  0.558  0.811  0.755  0.478  
long_vehicle     272      91  0.405  0.626  0.604  0.424  
    other     272      70      1      0  0.0211  0.0125  
pushback_truck     272      20      1      0  0.00533  0.00315  
stair_truck     272      39      1      0  0.0966  0.0644  
    truck     272     132  0.422  0.438  0.356  0.252  
      van     272     267  0.479  0.704  0.592  0.417  
Speed: 9.7/4.7/14.4 ms inference/NMS/total per 640x640 image at batch-size 16  
Results saved to runs/test/exp4
```

نمونه: **test_batch0_labels**



نمونه : test_batch0_pred



مدل سوم:

3_yolov7-ETVR.yaml

Modules: ETVR,C3C2

ماژول ETVR در هیچ کدام از فایل های گیت هاب yoloair وجود نداشت بنابراین با همانگی با تدریسیار مدل دیگری به دلخواه انتخاب شده

3_yolov7-Swin-HorNet.yaml

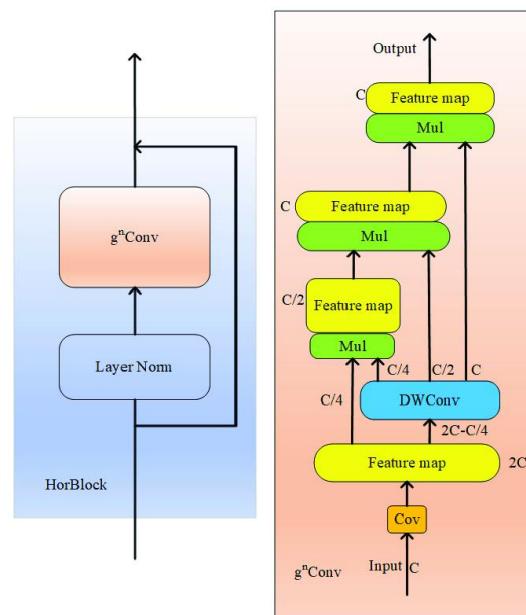
Modules: HorBlock ,C3STR

: horblock ماژول

پس از عبور شبکه از سومین ماژول CBS ، اندازه feature map به نصف کاهش می یابد و اطلاعات ویژگی به شدت کاهش می یابد. برای حفظ توانایی غیرخطی، توجه long-range ایجاد کنید، پدیده پراکندگی

گرادیان را کاهش دهید، و بهبود تشخیص عیوب عایق بلوکهای HorBlock متشکل از [28] gnConv پیچیدگی دروازه‌دار بازگشتی و نرمال‌سازی لایه لایه بین مازول‌های سوم و چهارم CBS اضافه شده‌اند.

شماتیک HorBlock. مازول gnConv با استفاده از کانولوشن استاندارد، نگاشت خطی، و ضرب المان ساخته شده است، اما با قابلیت‌های ترکیب فضایی تطبیقی ورودی مشابه توجه به خود. هنجر لایه میانگین و واریانس تمام پارامترها را در همه کanal‌ها محاسبه و سپس آنها را نرمال می‌کند. ساختار اصلی پیچش دروازه‌ای تفاوت زیادی با CNN استاندارد ندارد، اما مکانیسم دروازه در لایه کانولوشن معرفی شده است. اول، تعداد کanal‌های ویژگی توسط دو لایه کانولوشن تنظیم می‌شود. در مرحله بعد، نقشه ویژگی خروجی با پیچیدگی قابل جداسازی به چند قسمت تقسیم می‌شود و هر قسمت عنصر به عنصر ضرب می‌شود تا نقشه ویژگی خروجی به دست آید. بازگشت در اینجا ضرب ثابت عناصر است که از طریق آن می‌توان اطلاعات مرتبه بالاتر را ذخیره کرد. مکانیسم‌های توجه به طور کلی می‌توان به مکانیسم‌های توجه کanal، مکانیسم‌های توجه فضایی و ترکیبی از دو مکانیسم توجه تقسیم کرد. مازول‌های مکانیزم توجه سنتی مانند توجه فشرده و تحریک (SE) و مازول توجه بلوک کانولوشن (CBAM) نتایج خوبی را در مدل‌سازی روابط کanal به کanal به دست آورده‌اند، اما مستعد اطلاعات موقعیت مکانی هستند. اگرچه سایر مازول‌های توجه بدون این مشکل دارای اثرات خوبی هستند، اما تعداد پارامترها بسیار زیاد است و برای استقرار برنامه مناسب نیست.



منابع:

https://www.researchgate.net/publication/365223077_Real-time_Monitoring_Method_of_Strawberry_Fruit_Growth_State_Based_on_YOLO_Improved_Model#pf4

"Squeeze-and-Excitation Networks" by Jie Hu, Li Shen, and Gang Sun

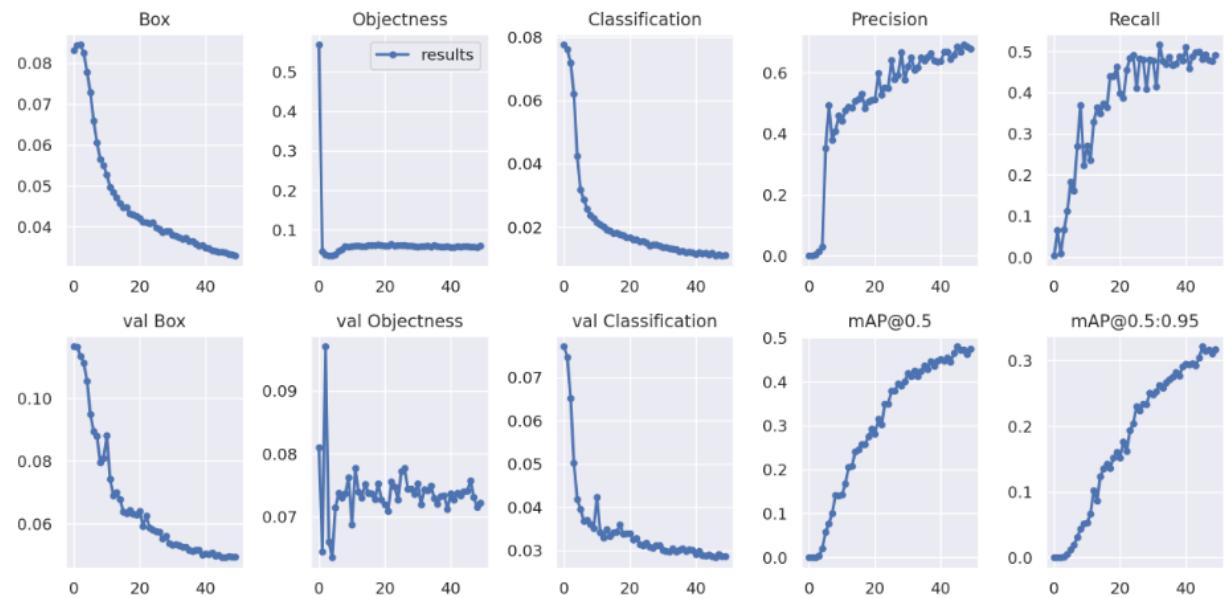
حال train کردن yolov7 با مدل بهبود یافته:

```
!python train.py --batch 16 --cfg  
/content/drive/MyDrive/yolov7/cfg/training/yolov7-Swin-HorNet.yaml --  
epochs 50 --data /content/drive/MyDrive/yolov7/sat-1/data.yaml --weights  
'yolov7.pt' --device 0
```

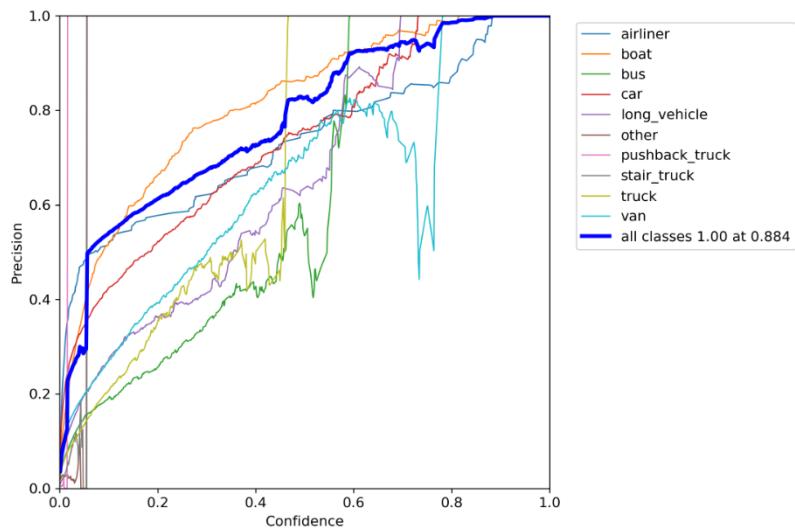
نتایج:

Epoch	gpu_mem	box	obj	cls	total	labels	img_size
49/49	13.4G	0.03288	0.05979	0.01111	0.1038	19	640: 100% 105/105 [02:35<00:00, 1.48]
	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 9/9 [00:10<00:00]
	all	272	1572	0.679	0.492	0.474	0.317
	airliner	272	72	0.623	0.986	0.952	0.757
	boat	272	274	0.774	0.92	0.907	0.534
	bus	272	84	0.361	0.643	0.43	0.289
	car	272	523	0.616	0.778	0.755	0.463
	long_vehicle	272	91	0.418	0.582	0.591	0.37
	other	272	70	1	0	0.0298	0.0177
	pushback_truck	272	20	1	0	0.00533	0.00272
	stair_truck	272	39	1	0	0.0727	0.0431
	truck	272	132	0.489	0.341	0.393	0.273
	van	272	267	0.511	0.665	0.605	0.423
23 epochs completed in 1.271 hours.							
Optimizer stripped from runs/train/exp50/weights/last.pt, 84.0MB							
Optimizer stripped from runs/train/exp50/weights/best.pt, 84.0MB							

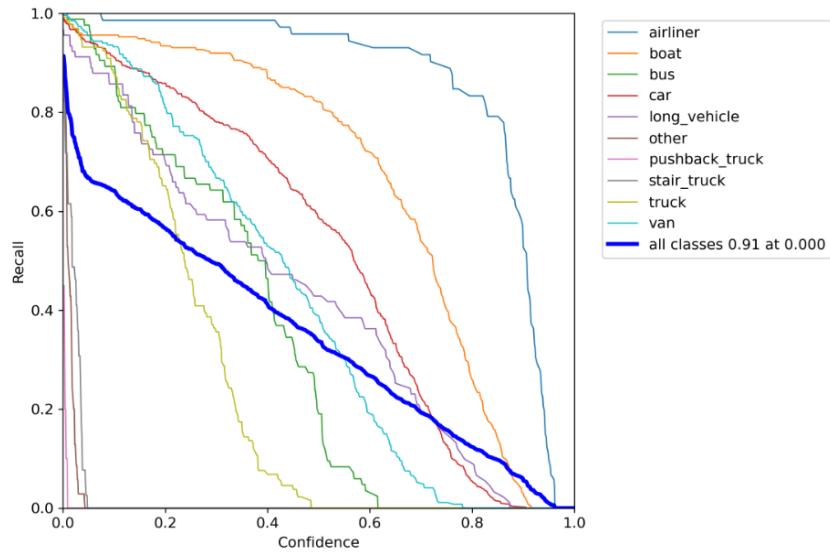
:results فایل



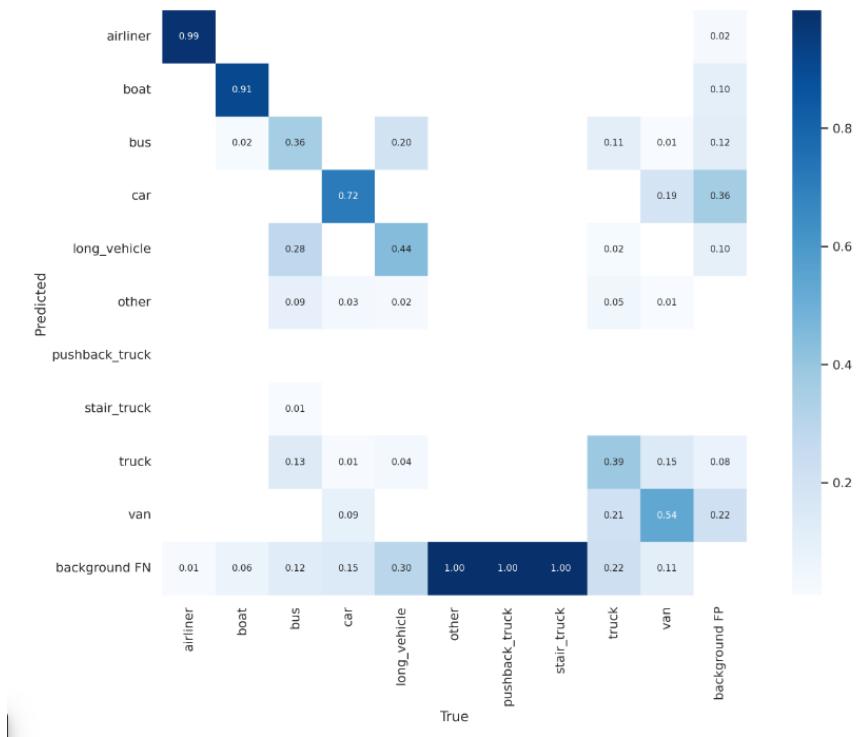
: P_curve فایل



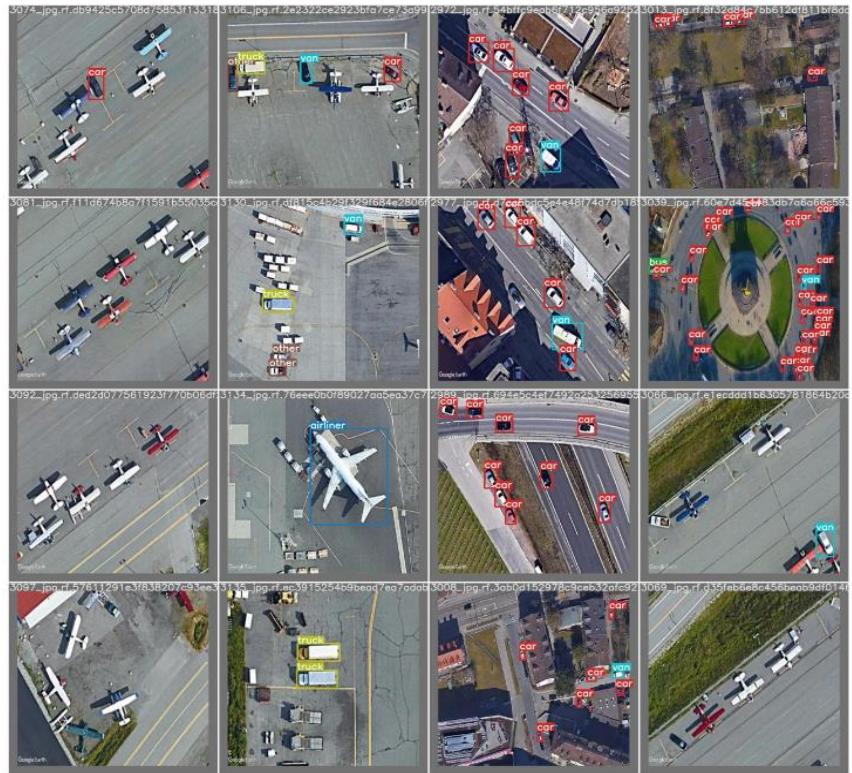
: R_curve فایل



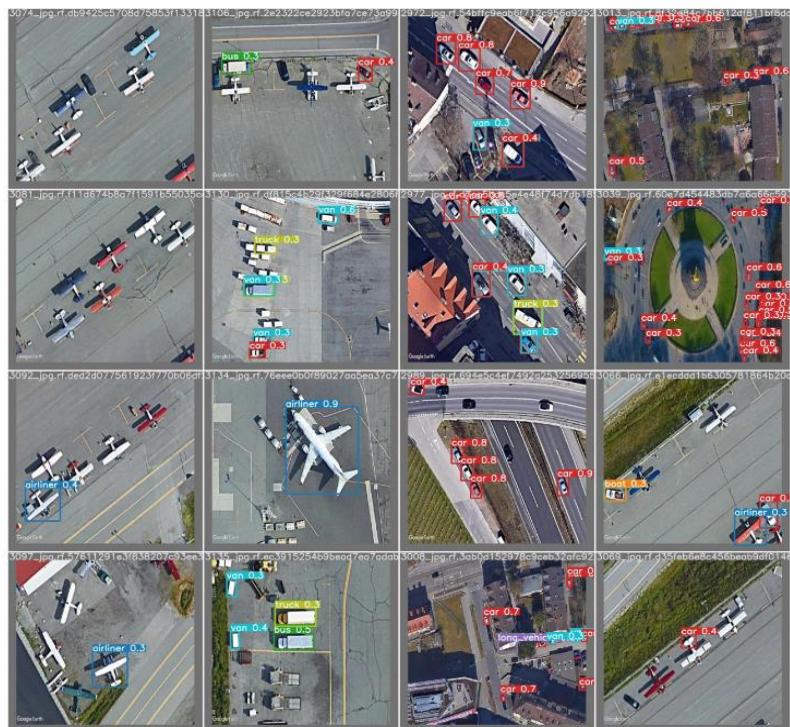
confusion ماتریس



نمونه: test_batch0_label



نمونه: test_batch0_pred



سپس فایل detect.py را اجرا می کنیم:

```
!python detect.py --weights  
/content/drive/MyDrive/yolov7/runs/train/exp50/weights/best.pt --source  
/content/drive/MyDrive/sat-1/test/images
```

: نمونه



بخش دوم:

<https://colab.research.google.com/drive/1I8JlobOLGg2PYKx0vLKnWgFQxfWS4yS?usp=sharing>

با استفاده از کد گیت گزارش شده در فایل پروژه yolov9 را به تنها بی آموزش دادیم.

از بین مدل های زیر مدل m برای yolov9 انتخاب کردیم:

Model	Test Size	AP ^{val}	AP ₅₀ ^{val}	AP ₇₅ ^{val}	Param.	FLOPs
YOLOv9-T	640	38.3%	53.1%	41.3%	2.0M	7.7G
YOLOv9-S	640	46.8%	63.4%	50.7%	7.1M	26.4G
YOLOv9-M	640	51.4%	68.1%	56.1%	20.0M	76.3G
YOLOv9-C	640	53.0%	70.2%	57.8%	25.3M	102.1G
YOLOv9-E	640	55.6%	72.8%	60.6%	57.3M	189.0G

سپس با این دستور آن را لود می کنیم:

```
%%bash
wget -P /content/drive/MyDrive/yolov9
https://github.com/WongKinYiu/yolov9/releases/download/v0.1/yolov9-m-
converted.pt
```

سپس آن را train می کنیم:

```
!python train.py --batch 2 --cfg
/content/drive/MyDrive/yolov9/models/detect/yolov9-m.yaml --epochs 40 --
data /content/drive/MyDrive/yolov9/sat-1/data.yaml --weights 'yolov9-m-
converted.pt' --device 0
```

ارور:

```
File "/content/drive/MyDrive/yolov9/utils/loss_tal.py", line 171, in
<listcomp>
    pred_distri, pred_scores = torch.cat([xi.view(feats[0].shape[0],
self.no, -1) for xi in feats], 2).split(
AttributeError: 'list' object has no attribute 'view'
```

راه حل: اضافه کردن این خط کد به جایی که ارور می دهد

```
feats = [torch.tensor(feat) if isinstance(feat, list) else feat for feat in feats]
```

ارور :2

```
File "/content/drive/MyDrive/yolov9/utils/loss_tal.py", line 169, in
<listcomp>
    feats = [torch.tensor(feat) if isinstance(feat, list) else feat for
feat in feats] #added
ValueError: only one element tensors can be converted to Python scalars
```

راه حل:

```
# Flatten nested lists and ensure elements are tensors
def flatten_and_convert(feats):
    flattened_feats = []
    for feat in feats:
        if isinstance(feat, list):
            flattened_feats.extend(flatten_and_convert(feat))
        elif isinstance(feat, torch.Tensor):
            flattened_feats.append(feat)
        else:
            # If feat is neither a list nor a tensor, print an error
message
            print(f"Unexpected type in feats: {type(feat)}")
    return flattened_feats

feats = flatten_and_convert(feats)

# Ensure all elements are tensors
feats = [torch.tensor(feat) if not isinstance(feat, torch.Tensor)
else feat for feat in feats]
```

ارور :3

```
RuntimeError: view size is not compatible with input tensor's size and
stride (at least one dimension spans across two contiguous subspaces). Use
.reshape(...) instead.
```

راه حل:

```
# Ensure the dimensions match
```

```

if distance.shape[0] != anchor_points.shape[0] or distance.shape[1] != anchor_points.shape[1] * 2:
    raise ValueError(f"Shape mismatch: distance shape {distance.shape} and anchor_points shape {anchor_points.shape} are incompatible.")

```

شروع به train

```

shape of lt: torch.Size([16, 8400, 2])
0/499      15G      5.426      6.773      5.502      245      640:  0% 0,
0/499      15G      5.426      6.773      5.502      245      640:  1% 1,
Initial shape of anchor_points: torch.Size([8400, 2])
Expanded shape of anchor_points: torch.Size([16, 8400, 2])
Shape of lt: torch.Size([16, 8400, 2])
Shape of rb: torch.Size([16, 8400, 2])
Shape of x1y1: torch.Size([16, 8400, 2])
Shape of x2y2: torch.Size([16, 8400, 2])
Initial shape of distance: torch.Size([16, 8400, 4])
Initial shape of anchor_points: torch.Size([8400, 2])
Expanded shape of anchor_points: torch.Size([16, 8400, 2])

```

سپس train با مدل بهبود یافته yolov7

```

!python train_dual.py --workers 8 --device 0 --batch 16 --data
/content/drive/MyDrive/yolov9/sat-1/data.yaml --img 640 --cfg
/content/drive/MyDrive/yolov9/models/detect/yolov7-Swin-HorNet.yaml --
weights 'yolov9-m-converted.pt' --name yolov9-improved --hyp
/content/drive/MyDrive/yolov9/data/hyps/hyp.scratch-high.yaml --min-items
0 --epochs 50 --close-mosaic 15

```

ارور:

NameError: name 'RepConv' is not defined. Did you mean: 'RepConvN'?

راه حل:

در فایل های yoloair و repconv common را پیدا کرده و در پوشه آن را اضافه می کنیم

ارور 2:

NameError: name 'IDetect' is not defined. Did you mean: 'Detect'?

راه حل 1:

در هر دو فایل common و yolo به دنبال IDetect گشتمیم ولی اصلا همچین کلمه ای وجود نداشت تا به جای آن بنویسیم و ارور رفع شود

راه حل 2:

نوشتن تابعی برای جایگذاری این دو کلمه:

```
def replace_idetect_with_detect(file_path):
    # Read the file
    with open(file_path, 'r') as file:
        file_content = file.read()

    # Replace IDetect with Detect
    updated_content = file_content.replace('IDetect', 'Detect')

    # Write the updated content back to the file
    with open(file_path, 'w') as file:
        file.write(updated_content)

    print(f'Replaced \'IDetect\' with \'Detect\' in {file_path}')

# Example usage
file_path = '/content/drive/MyDrive/yolov9/models/common.py'
replace_idetect_with_detect(file_path)
```

راه حل 3:

یک مدل بمبود یافته دیگه امتحان شد.

نتیجه: در نهایت هیچ کدام از راه حل ها ارور را ببرطرف نکرد

بخش سوم:

قسمت اول: yolov5

https://colab.research.google.com/drive/1HQVUW06gLaQKVmomz-CHwV_tqW4TGbi?usp=sharing

در این قسمت یک بار yolov5 را به تنهایی آموزش میدهیم. برای این قسمت از وزنهای مدل [YOLOv5m](#) استفاده شده است

```
      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     4709184  models.common.C3HB      [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      60615  models.yolo.Detect      [10, [[10, 13, 16, 30, 33, :
```

Model summary: 309 layers, 21481959 parameters, 21481959 gradients, 48.8 GFLOPs

Transferred 505/505 items from runs/train/exp11/weights/last.pt

آموزش روی 40 ایپاک انجام شده است و نتایج به صورت زیر به دست آمده است.



```
40 epochs completed in 1.799 hours.  
Optimizer stripped from runs/train/exp2/weights/last.pt, 42.3MB  
Optimizer stripped from runs/train/exp2/weights/best.pt, 42.3MB
```

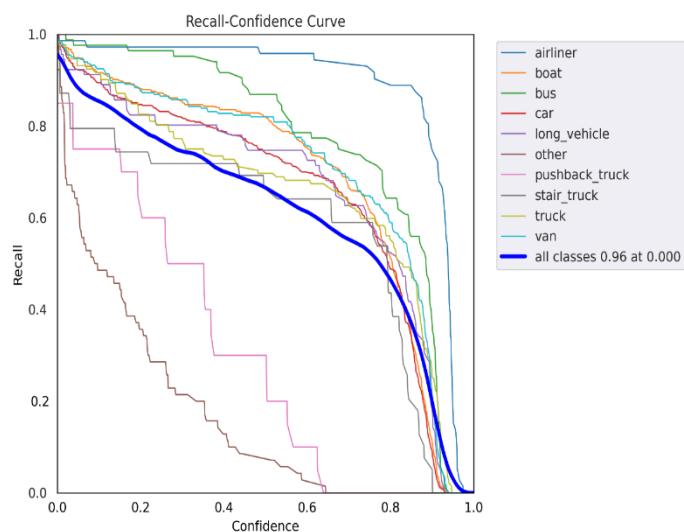
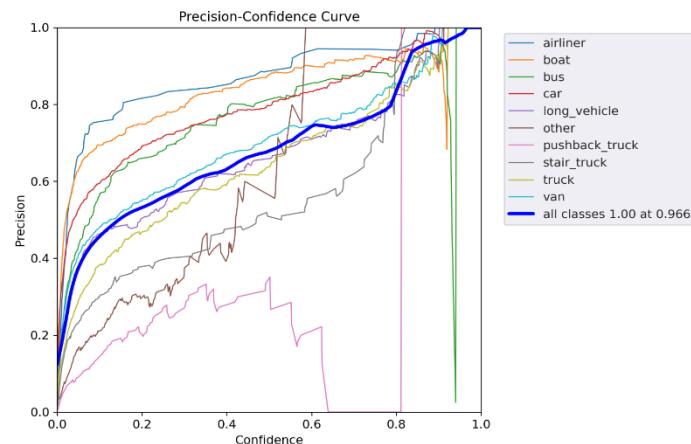
```
Validating runs/train/exp2/weights/best.pt...  
Fusing layers...
```

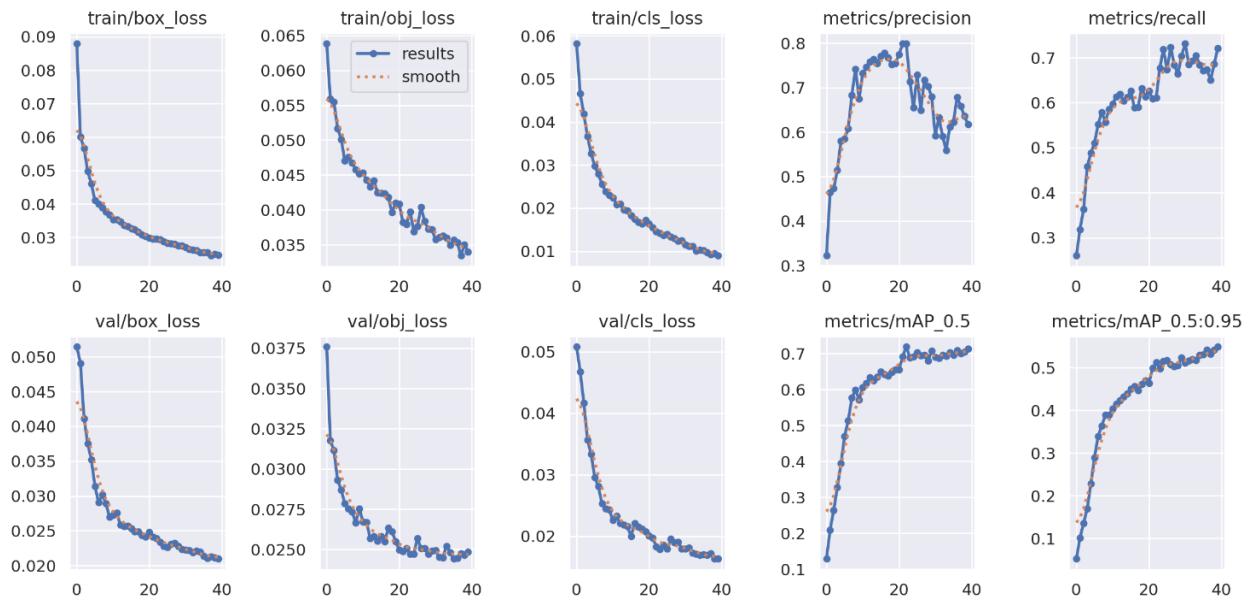
```
YOLOv5m summary: 212 layers, 20889303 parameters, 0 gradients, 48.0 GFLOPs
```

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 68/68
all	272	1572	0.617	0.721	0.713	0.548
airliner	272	72	0.844	0.972	0.975	0.872
boat	272	274	0.831	0.843	0.883	0.652
bus	272	84	0.746	0.944	0.893	0.695
car	272	523	0.76	0.799	0.853	0.641
long_vehicle	272	91	0.611	0.802	0.769	0.594
other	272	70	0.411	0.157	0.27	0.195
pushback_truck	272	20	0.31	0.4	0.24	0.164
stair_truck	272	39	0.434	0.718	0.668	0.411
truck	272	132	0.579	0.739	0.756	0.606
van	272	267	0.643	0.835	0.823	0.655

```
Results saved to runs/train/exp2
```

: P_curve فایل





improved yolov5: قسمت دوم

برای این قسمت از مدل بهبودیافته EMO/yolov5-EMO-CSRMBC.yaml استفاده کردیم. در این مدل بهبود یافته از مازول CSRRMB استفاده شده است. معماری مازول CSRRMB با هدف بهبود عملکرد و کارایی YOLOv5 طراحی شده است. این مازول از قسمت های زیر تشکیل شده است:

Residual Connections . 1

این اتصالات به شبکه کمک می کنند تا مشکلات ناشی از ناپدید شدن گرادیان در شبکه های عمیق را کاهش دهد و فرآیند آموزش را بهبود بخشد. همچنین از طریق این اتصالات، اطلاعات به طور مستقیم از ورودی به خروجی منتقل می شوند.

Bottleneck Layers . 2

معمولًاً شامل یک کانولوشن یک در یک برای کاهش بعد و یک کانولوشن سه در سه برای استخراج ویژگی های فضایی می باشد.

Cross-Stage Partial (CSP) Architecture: .3

معماری CSP، ویژگی‌ها را به دو بخش تقسیم می‌کند: یک بخش تحت تغییرات مختلف قرار می‌گیرد و بخش دیگر به صورت مستقیم عبور داده می‌شود. سپس این دو بخش در انتهای با هم ترکیب می‌شوند. این طراحی به کاهش هزینه محاسباتی کمک می‌کند و در عین حال غنای ویژگی‌ها را حفظ می‌کند.

Convolutional Layers .4

این لایه‌ها برای استخراج و تبدیل ویژگی‌ها به کار می‌روند و شامل یک سری عملیات کانولوشنی با اندازه کرنل، گام و پدینگ خاص می‌باشند.

عملکرد مازول CSRRMB

- غنی‌سازی ویژگی‌ها: با تقسیم نقشه ویژگی و پردازش بخش‌های مختلف آن، مازول قادر است ویژگی‌های متنوع‌تری را استخراج کند. ویژگی‌های عبوری اطلاعات اصلی را حفظ می‌کنند در حالی که ویژگی‌های تغییر یافته، زمینه اضافی را اضافه می‌کنند.
- کارایی: معماری CSP به کاهش تعداد پارامترها و پیچیدگی محاسباتی کمک می‌کند. این امر برای وظایف تشخیص اشیاء در زمان واقعی که YOLOv5 اغلب به کار می‌رود، بسیار حیاتی است.
- جریان گرادیان: استراتژی تقسیم-تبدیل-ترکیب، جریان گرادیان‌ها را در طی فرآیند پس‌انتشار بهبود می‌بخشد که به آموزش مؤثرتر شبکه‌های عمیق کمک می‌کند.

```
13     # YOLOv5 backbone
14     backbone:
15         # [from, number, module, args]
16         [[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
17          [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
18          [-1, 3, CSRRMB, [128]],
19          [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
20          [-1, 6, CSRRMB, [256]],
21          [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
22          [-1, 9, CSRRMB, [512]],
23          [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
24          [-1, 3, CSRRMB, [1024]],
25          [-1, 1, SPPF, [1024, 5]], # 9
26      ]
```

معماری این بلاک به صورت زیر است.

```

28     # YOLOv5 head
29     head:
30         [[-1, 1, Conv, [512, 1, 1]],
31          [-1, 1, nn.Upsample, [None, 2, 'nearest']]],
32          [[-1, 6], 1, Concat, [1]], # cat backbone P4
33          [-1, 3, C3, [512, False]], # 13
34
35          [-1, 1, Conv, [256, 1, 1]],
36          [-1, 1, nn.Upsample, [None, 2, 'nearest']],
37          [[-1, 4], 1, Concat, [1]], # cat backbone P3
38          [-1, 3, C3, [256, False]], # 17 (P3/8-small)
39
40          [-1, 1, Conv, [256, 3, 2]],
41          [[-1, 14], 1, Concat, [1]], # cat head P4
42          [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
43
44          [-1, 1, Conv, [512, 3, 2]],
45          [[-1, 10], 1, Concat, [1]], # cat head P5
46          [-1, 3, C3, [1024, False]], # 23 (P5/32-large)
47
48          [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
49      ]

```

به علت پاک شدن تابع `define` مازول YOLOAIR در گیت هاب CSRRMB مجبور شدیم از مازول دیگری که کد تعریف آن وجود داشت برای آموزش YOLO5 استفاده کنیم. مازول دومی که استفاده شد است که معماری آن به صورت زیر است. مازول C3HB می‌تواند به عنوان یک ابزار مکمل در کنار معماری YOLO استفاده شود تا دقیق و کارایی این مدل را بهبود بخشد.

```

+++
12 # YOLOv5 v6.0 backbone
13 backbone:
14     # [from, number, module, args]
15     [[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
16        [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
17        [-1, 3, C3, [128]],
18        [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
19        [-1, 6, C3, [256]],
20        [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
21        [-1, 9, C3, [512]],
22        [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
23        [-1, 3, C3HB, [1024]],
24        [-1, 1, SPPF, [1024, 5]], # 9
25    ]
26

```

```

27 # YOLOv5 v6.0 head
28 head:
29   [[-1, 1, Conv, [512, 1, 1]],
30    [-1, 1, nn.Upsample, [None, 2, 'nearest']],
31    [[-1, 6], 1, Concat, [1]], # cat backbone P4
32    [-1, 3, C3, [512, False]], # 13
33
34   [-1, 1, Conv, [256, 1, 1]],
35   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
36   [[-1, 4], 1, Concat, [1]], # cat backbone P3
37   [-1, 3, C3, [256, False]], # 17 (P3/8-small)
38
39   [-1, 1, Conv, [256, 3, 2]],
40   [[-1, 14], 1, Concat, [1]], # cat head P4
41   [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
42
43   [-1, 1, Conv, [512, 3, 2]],
44   [[-1, 10], 1, Concat, [1]], # cat head P5
45   [-1, 3, C3, [1024, False]], # 23 (P5/32-large)
46
47   [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
48 ]

```

برای استفاده از مازول بالا نیاز بود تا تعریف توابع زیر را در **COMMN.PY** اضافه کنیم تا ارورها برطرف شوند.

```

class C3HB(nn.Module):
    # CSP HorBlock with 3 convolutions by iscyy/yoloair
    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5): # ch_in,
        ch_out, number, shortcut, groups, expansion
            super().__init__()
            c_ = int(c2 * e) # hidden channels
            self.cv1 = Conv(c1, c_, 1, 1)
            self.cv2 = Conv(c1, c_, 1, 1)
            self.cv3 = Conv(2 * c_, c2, 1)
            self.m = nn.Sequential(*([HorBlock(c_) for _ in range(n)]))
    def forward(self, x):
        return self.cv3(torch.cat((self.m(self.cv1(x)), self.cv2(x)),
dim=1))

class C3(nn.Module):
    # CSP Bottleneck with 3 convolutions
    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5): # ch_in,
        ch_out, number, shortcut, groups, expansion
            super().__init__()
            c_ = int(c2 * e) # hidden channels
            self.cv1 = Conv(c1, c_, 1, 1)
            self.cv2 = Conv(c1, c_, 1, 1)

```

```

        self.cv3 = Conv(2 * c_, c2, 1) # act=FReLU(c2)
        self.m = nn.Sequential(*Bottleneck(c_, c_, shortcut, g, e=1.0)
for _ in range(n))

    def forward(self, x):
        return self.cv3(torch.cat((self.m(self.cv1(x)), self.cv2(x)),
dim=1))

class gnconv(nn.Module):
    def __init__(self, dim, order=5, gflayer=None, h=14, w=8, s=1.0):
        super().__init__()
        self.order = order
        self.dims = [dim // 2 ** i for i in range(order)]
        self.dims.reverse()
        self.proj_in = nn.Conv2d(dim, 2*dim, 1)

        if gflayer is None:
            self.dwconv = get_dwconv(sum(self.dims), 7, True)
        else:
            self.dwconv = gflayer(sum(self.dims), h=h, w=w)

        self.proj_out = nn.Conv2d(dim, dim, 1)

        self.pws = nn.ModuleList(
            [nn.Conv2d(self.dims[i], self.dims[i+1], 1) for i in
range(order-1)])
    )
    self.scale = s

    def forward(self, x, mask=None, dummy=False):
        # B, C, H, W = x.shape gnconv [512]by iscyy/air
        fused_x = self.proj_in(x)
        pwa, abc = torch.split(fused_x, (self.dims[0], sum(self.dims)), dim=1)
        dw_abc = self.dwconv(abc) * self.scale
        dw_list = torch.split(dw_abc, self.dims, dim=1)
        x = pwa * dw_list[0]
        for i in range(self.order -1):
            x = self.pws[i](x) * dw_list[i+1]
        x = self.proj_out(x)

        return x

```

```

def get_dwconv(dim, kernel, bias):
    return nn.Conv2d(dim, dim, kernel_size=kernel, padding=(kernel-1)//2
, bias=bias, groups=dim)

class HorBlock(nn.Module):
    r""" HorNet block
    """
    def __init__(self, dim, drop_path=0., layer_scale_init_value=1e-6,
gnconv=gnconv):
        super().__init__()

        self.norm1 = HorLayerNorm(dim, eps=1e-6,
data_format='channels_first')
        self.gnconv = gnconv(dim)
        self.norm2 = HorLayerNorm(dim, eps=1e-6)
        self.pwconv1 = nn.Linear(dim, 4 * dim)
        self.act = nn.GELU()
        self.pwconv2 = nn.Linear(4 * dim, dim)

        self.gamma1 = nn.Parameter(layer_scale_init_value *
torch.ones(dim),
                           requires_grad=True) if
layer_scale_init_value > 0 else None

        self.gamma2 = nn.Parameter(layer_scale_init_value *
torch.ones((dim)),
                           requires_grad=True) if
layer_scale_init_value > 0 else None
        self.drop_path = DropPath(drop_path) if drop_path > 0. else
nn.Identity()

    def forward(self, x):
        B, C, H, W = x.shape # [512]
        if self.gamma1 is not None:
            gamma1 = self.gamma1.view(C, 1, 1)
        else:
            gamma1 = 1
        x = x + self.drop_path(gamma1 * self.gnconv(self.norm1(x)))

        input = x
        x = x.permute(0, 2, 3, 1) # (N, C, H, W) -> (N, H, W, C)
        x = self.norm2(x)
        x = self.pwconv1(x)

```

```

        x = self.act(x)
        x = self.pwconv2(x)
        if self.gamma2 is not None:
            x = self.gamma2 * x
        x = x.permute(0, 3, 1, 2) # (N, H, W, C) -> (N, C, H, W)

        x = input + self.drop_path(x)
        return x

class DropPath(nn.Module):
    def __init__(self, drop_prob=None):
        super(DropPath, self).__init__()
        self.drop_prob = drop_prob

    def forward(self, x):
        return drop_path_f(x, self.drop_prob, self.training)

def drop_path_f(x, drop_prob: float = 0., training: bool = False):
    if drop_prob == 0. or not training:
        return x
    keep_prob = 1 - drop_prob
    shape = (x.shape[0],) + (1,) * (x.ndim - 1) # work with diff dim tensors, not just 2D ConvNets
    random_tensor = keep_prob + torch.rand(shape, dtype=x.dtype,
device=x.device)
    random_tensor.floor_() # binarize
    output = x.div(keep_prob) * random_tensor
    return output


class HorLayerNorm(nn.Module):
    def __init__(self, normalized_shape, eps=1e-6,
data_format="channels_last"):
        super().__init__()
        self.weight = nn.Parameter(torch.ones(normalized_shape))
        self.bias = nn.Parameter(torch.zeros(normalized_shape))
        self.eps = eps
        self.data_format = data_format
        if self.data_format not in ["channels_last", "channels_first"]:
            raise NotImplementedError # by iscyy/air
        self.normalized_shape = (normalized_shape, )

    def forward(self, x):
        if self.data_format == "channels_last":

```

```

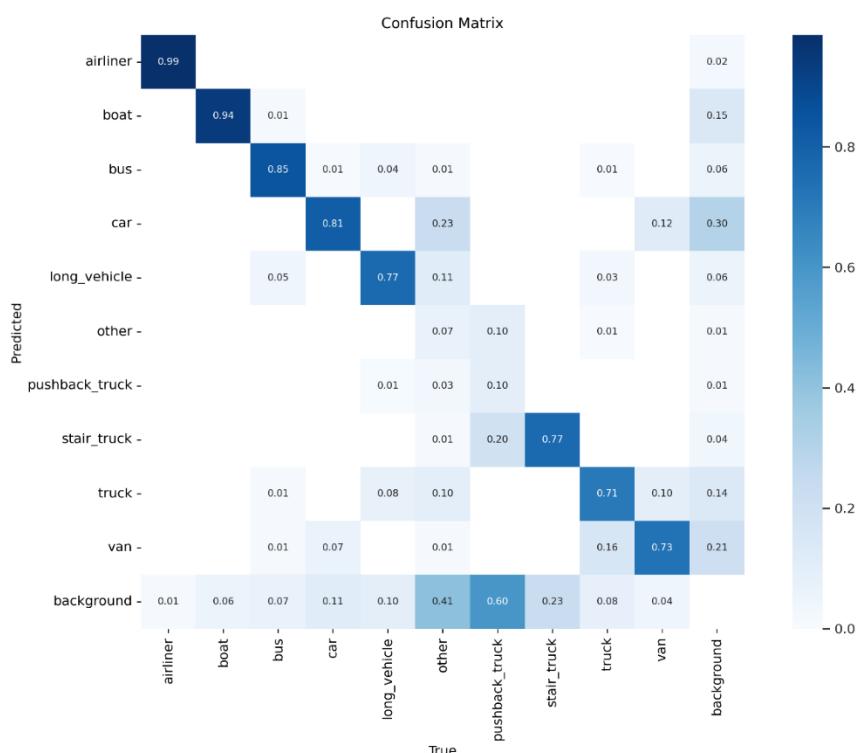
        return F.layer_norm(x, self.normalized_shape, self.weight,
self.bias, self.eps)
    elif self.data_format == "channels_first":
        u = x.mean(1, keepdim=True)
        s = (x - u).pow(2).mean(1, keepdim=True)
        x = (x - u) / torch.sqrt(s + self.eps)
        x = self.weight[:, None, None] * x + self.bias[:, None, None]
    return x

```

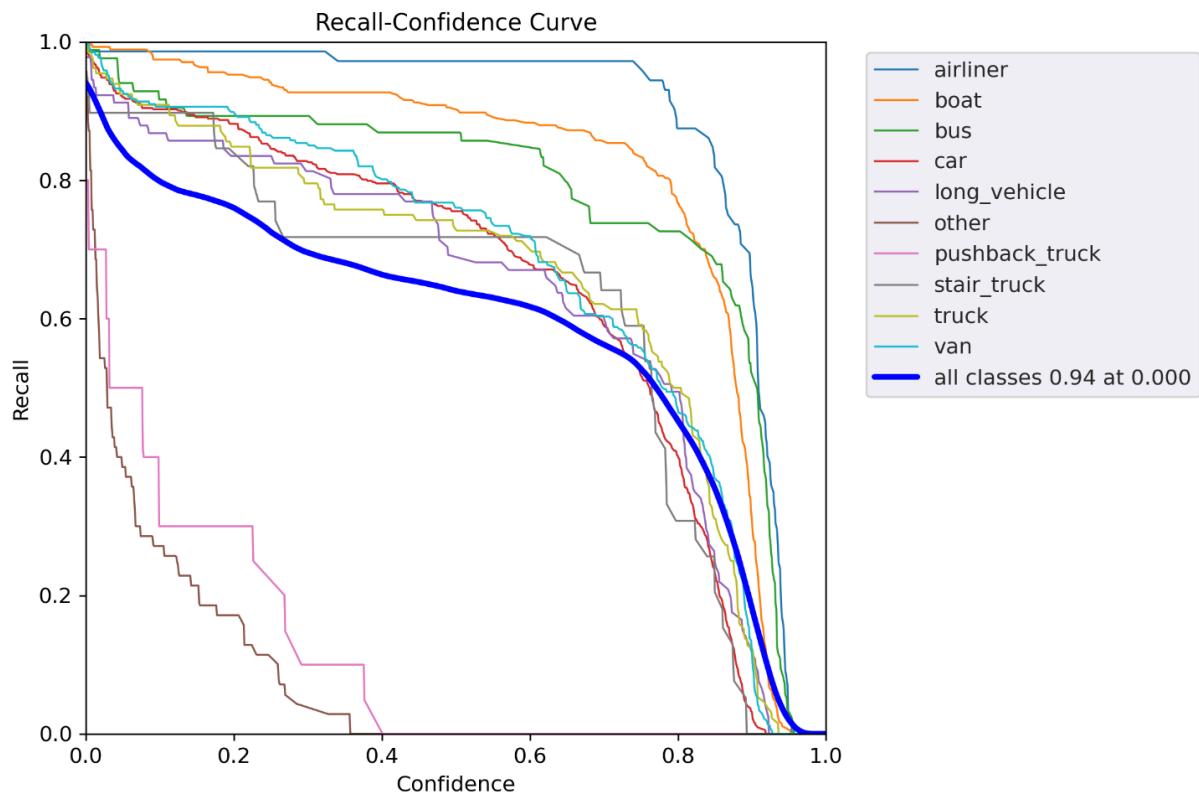
در مرحله بعد با برطرف کردن ارورها مدل را در دوم مرحله ترین کردیم و نتایج زیر به دست آمد.

Class	Images	Instances	P	R	mAP50	mAP50-95: 100%	68/68 [
all	272	1572	0.822	0.622	0.701	0.528	
airliner	272	72	0.834	0.972	0.977	0.838	
boat	272	274	0.884	0.886	0.94	0.691	
bus	272	84	0.781	0.85	0.886	0.658	
car	272	523	0.828	0.689	0.827	0.61	
long_vehicle	272	91	0.746	0.67	0.77	0.578	
other	272	70	1	0	0.174	0.124	
pushback_truck	272	20	1	0	0.16	0.118	
stair_truck	272	39	0.663	0.718	0.72	0.432	
truck	272	132	0.724	0.712	0.729	0.579	
van	272	267	0.765	0.723	0.825	0.655	

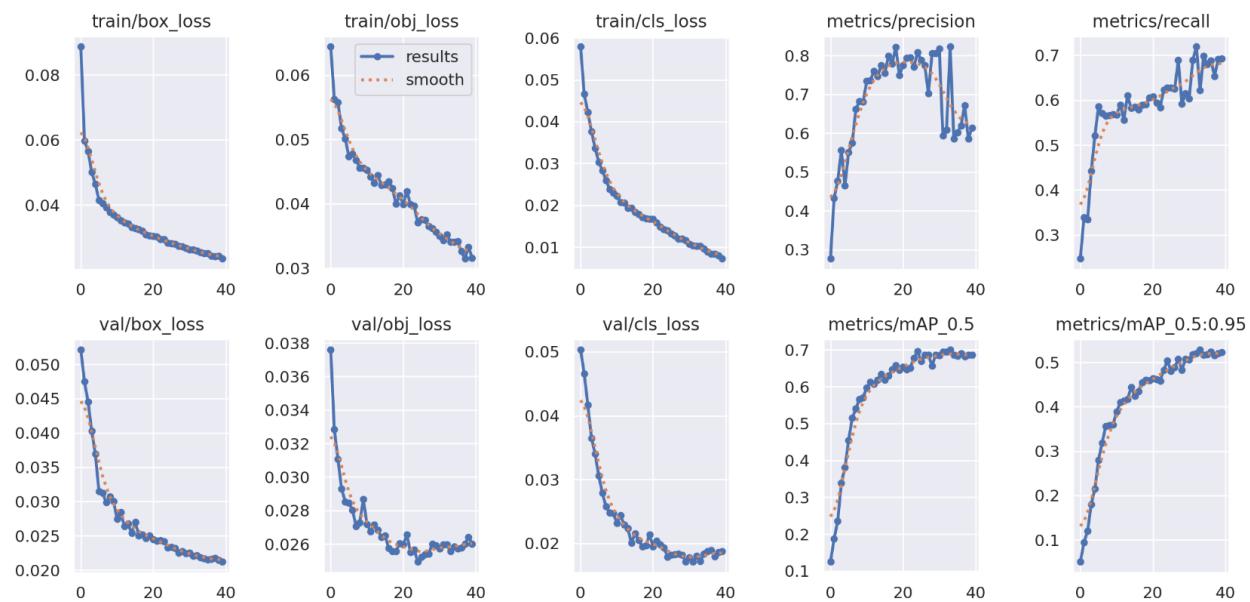
Results saved to runs/train/exp11



:R_curve فایل



:results فایل



قسمت سوم: YOLOV8 IMPROVED

<https://colab.research.google.com/drive/1lRs5p7t5lKpvogjw-pfCbiYuJPhyA3Ji?usp=sharing>

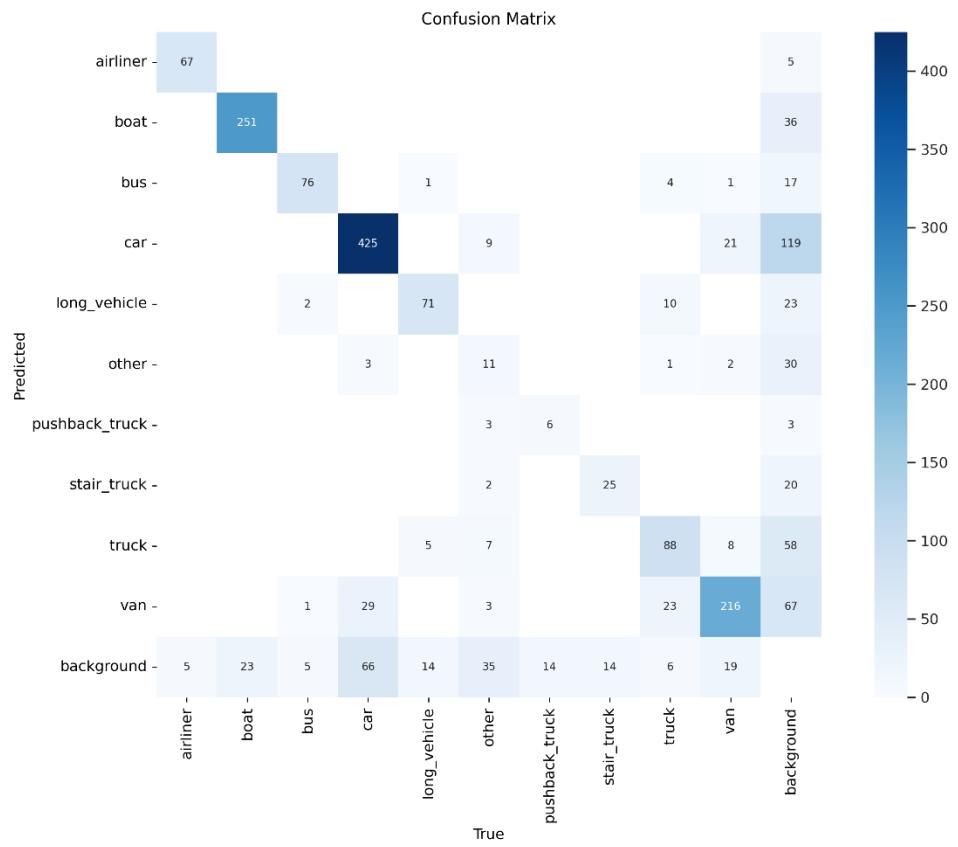
در این قسمت از گیت هاب YOLOV8 موجود استفاده میکنیم و از وزنهای از قبیل آموزش دیده مدل YOLOV8M استفاده کرده و با اضافه کردن ماذول معرفی شده در قسمت قبل مدل را آموزش میدهیم. دقت شود که نحوه آموزش مدل YOLOV8 متفاوت با یولوهای دیگر است.

		from	n	params	module	arguments
0		-1	1	1392	ultralytics.nn.modules.conv.Conv	[3, 48, 3, 2]
1		-1	1	41664	ultralytics.nn.modules.conv.Conv	[48, 96, 3, 2]
2		-1	2	111360	ultralytics.nn.modules.block.C2f	[96, 96, 2, True]
3		-1	1	166272	ultralytics.nn.modules.conv.Conv	[96, 192, 3, 2]
4		-1	4	813312	ultralytics.nn.modules.block.C2f	[192, 192, 4, True]
5		-1	1	664320	ultralytics.nn.modules.conv.Conv	[192, 384, 3, 2]
6		-1	4	3248640	ultralytics.nn.modules.block.C2f	[384, 384, 4, True]
7		-1	1	1991808	ultralytics.nn.modules.conv.Conv	[384, 576, 3, 2]
8		-1	2	3985920	ultralytics.nn.modules.block.C2f	[576, 576, 2, True]
9		-1	1	831168	ultralytics.nn.modules.block.SPPF	[576, 576, 5]
10		-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11		[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
12		-1	2	1993728	ultralytics.nn.modules.block.C2f	[960, 384, 2]
13		-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14		[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
15		-1	2	517632	ultralytics.nn.modules.block.C2f	[576, 192, 2]
16		-1	1	332160	ultralytics.nn.modules.conv.Conv	[192, 192, 3, 2]
17		[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat	[1]
18		-1	2	1846272	ultralytics.nn.modules.block.C2f	[576, 384, 2]
19		-1	1	1327872	ultralytics.nn.modules.conv.Conv	[384, 384, 3, 2]
20		[-1, 9]	1	0	ultralytics.nn.modules.conv.Concat	[1]
21		-1	2	4207104	ultralytics.nn.modules.block.C2f	[960, 576, 2]
22		[15, 18, 21]	1	3781486	ultralytics.nn.modules.head.Detect	[10, [192, 384, 576]]
Model summary: 295 layers, 25,862,110 parameters, 25,862,094 gradients						

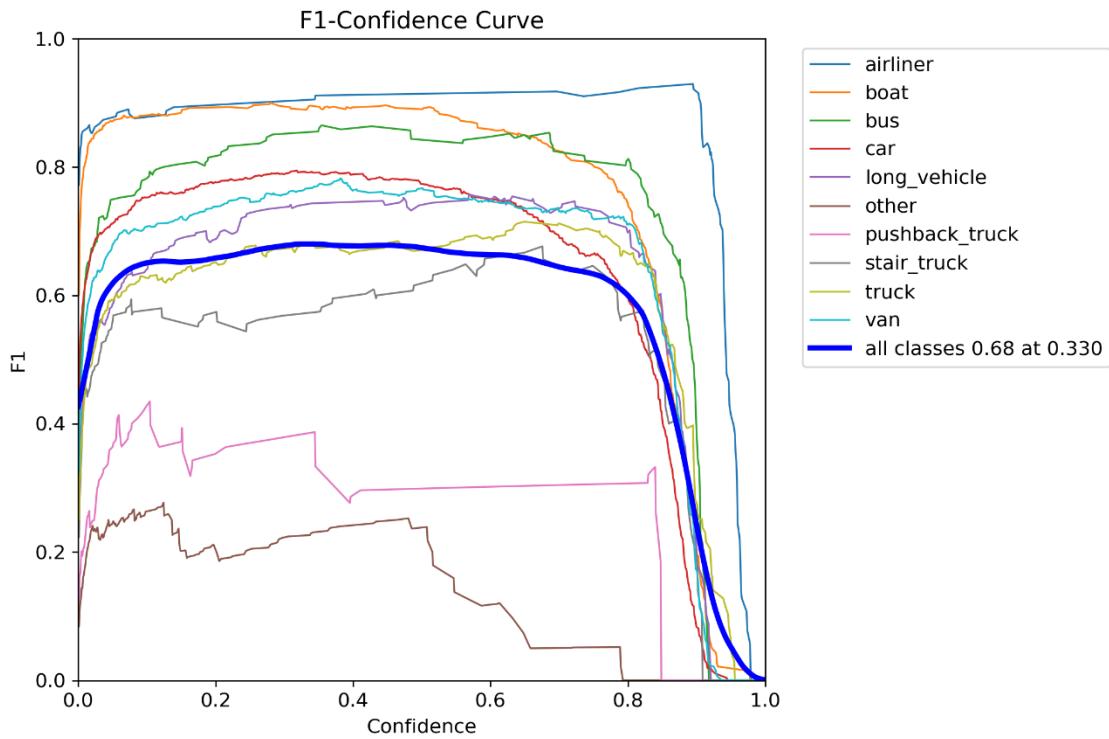
آموزش را روی 40 ایپاک انجام دادیم و نتایج به صورت زیر به دست آمد.

```
Validating /content/drive/MyDrive/ultralytics/runs/detect/train5/weights/best.pt...
Ultralytics YOLOv8.2.57 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25,845,550 parameters, 0 gradients
      Class    Images   Instances     Box(P       R      mAP50    mAP50-95): 100%|██████████
        all      272      1572     0.686     0.697     0.728     0.568
      airliner     57       72     0.881     0.931     0.946     0.869
       boat      21      274     0.897     0.893     0.936     0.702
       bus       33       84     0.801     0.917     0.883      0.7
       car      125      523     0.792     0.793     0.857     0.654
  long_vehicle     30       91     0.71      0.78     0.798     0.622
      other      38       70     0.349     0.171     0.192     0.135
 pushback_truck     16       20     0.542      0.3     0.407     0.281
  stair_truck      28       39     0.519     0.641     0.678     0.437
      truck      75      132     0.656     0.712     0.768     0.628
       van      105      267     0.71     0.835     0.812     0.656
Speed: 0.6ms preprocess, 10.9ms inference, 0.0ms loss, 10.9ms postprocess per image
Results saved to /content/drive/MyDrive/ultralytics/runs/detect/train5
ultralytics.utils.metrics.DetMetrics object with attributes:
```

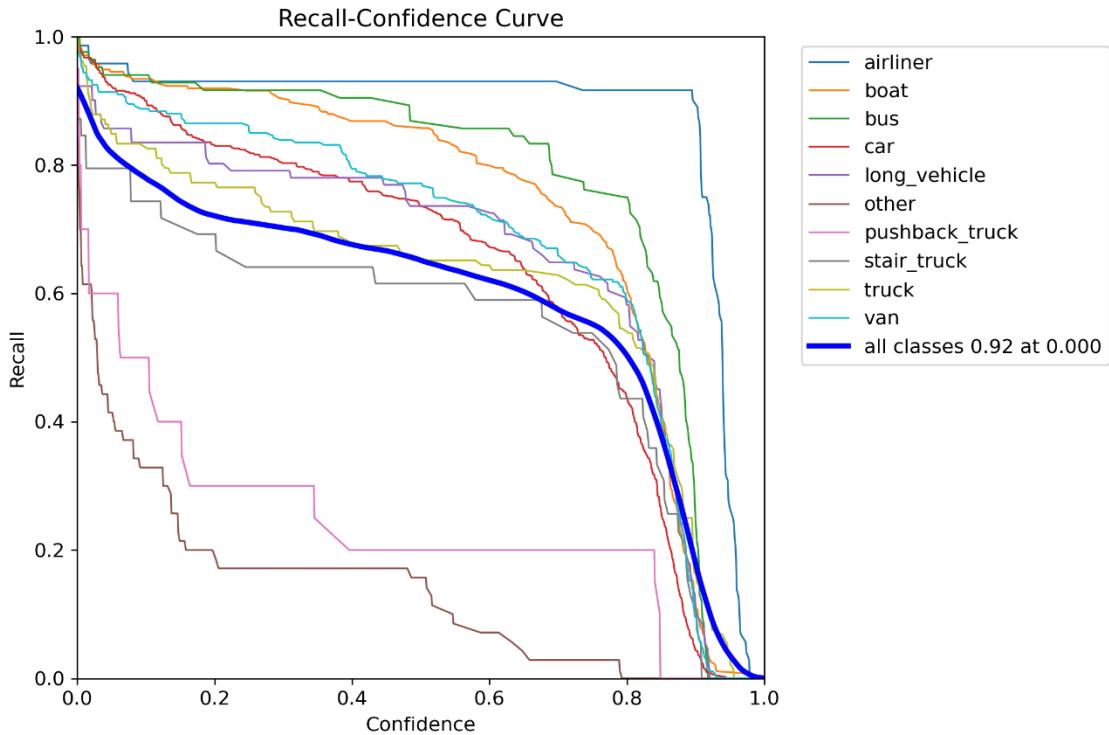
:confusion ماتریس



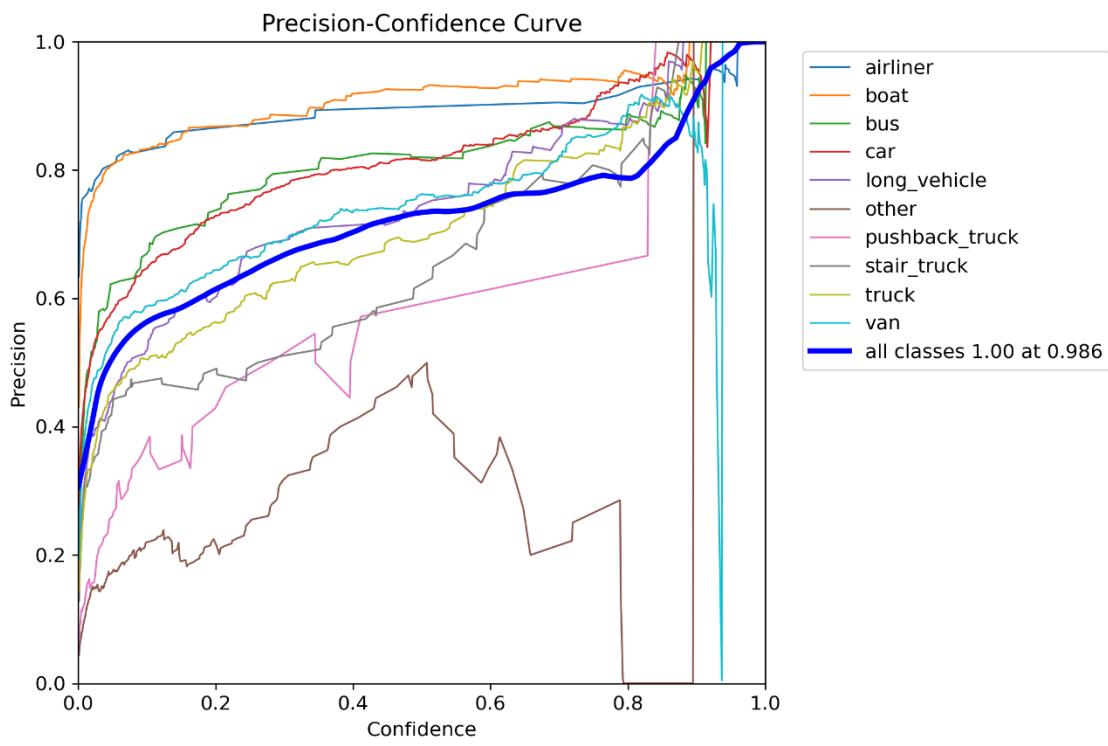
فایل F1:



:R_curve فایل



فایل P_Curve



results فایل

