به نام خدا

گزارش کد و توضیحات مرتبط با داده‌های استفاده شده

درس: یادگیری ماشین آماری

امیرحسین حیدری

بهمن ۱۴۰۳

در این فایل ابتدا نوع و ساختار داده‌ها توضیح داده شده سپس به سراغ گزارش کار کدها رفته و در مرحله آخر هم توضیحات مرتب با خروجی و دقت مدل و نمونه خروجی مدل آورده شده‌است.

توضیحات مرتبط با داده:

۱ .نوآوری در استفاده از:YOLO4

یکی از دلایل اصلی انتخاب این دیتابیس، عدم وجود کد و پیاده‌سازی موجود برای نسخه YOLO4بر روی همین مجموعه داده‌ها بود. این موضوع از یک سو چالشی برای پژوهشگران به حساب می‌آید و از سوی دیگر، فرصتی طلایی برای به‌روزرسانی و ارتقای مدل‌های تشخیص شیء محسوب می‌شود. به عبارت دیگر، با استفاده از این دیتابیس می‌توانیم از نوآوری و خلاقیت در بهبود عملکرد YOLO4 بهره ببریم و مدل را در شرایط واقعی آزمایش کنیم.

۲ .کیفیت بالای برچسب‌گذاری و ساختار داده‌ای مطمئن:

این دیتابیس دارای برچسب‌گذاری دقیق و حرفه‌ای است که یکی از عوامل کلیدی در آموزش موفق مدل‌های یادگیری عمیق به شمار می‌آید. برچسب‌های صحیح و دقیق به مدل کمک می‌کند تا با دقت بیشتری ویژگی‌های موجود در تصاویر را تشخیص دهد و در نتیجه عملکرد بهتری ارائه کند. از طرفی، ساختار داده‌ای این مجموعه به گونه‌ای است که استفاده از آن در کاربردهای واقعی راحت و مطمئن می‌باشد.

۳ .کاربرد عملی در دنیای واقعی:

دیتابیس انتخاب شده تنها تصاویر ساده و با کیفیت بالا را شامل نمی‌شود؛ بلکه شرایط واقعی محیط‌های عملی را نیز به خوبی منعکس می‌کند. به عنوان مثال، مدل آموزش دیده بر روی این داده‌ها قابلیت استفاده در صنعت راهداری را داراست. با توجه به اینکه دوربین‌های آنلاین در اتوبوس‌های سطح کشور نصب شده‌اند، این مدل می‌تواند به عنوان ابزاری کارآمد برای نظارت بر وضعیت خودروهای کار، وسایل نقلیه ترابری و اتوبوس‌ها به کار رود. در واقع، این دیتابیس شرایط واقعی نظارت و چالش‌های مربوط به محیط‌های عملی را در بر دارد و این امر تضمین می‌کند که مدل نهایی در مواجهه با مسائل روزمره و متنوع عملکرد قابل اطمینانی داشته باشد.

۴ .بهبود عملکرد مدل در شرایط متنوع:

با استفاده از دیتابیس‌هایی که شامل تصاویر تمیز و با کیفیت به همراه تصاویر دارای شرایط واقعی و چالش‌برانگیز هستند، می‌توانیم مدل را طوری آموزش دهیم که در مواجهه با هر دو حالت ایده‌آل و واقعی عملکرد مناسبی داشته باشد. این موضوع به ویژه در کاربردهایی که سرعت و دقت در شناسایی مسائل مهم هستند، از اهمیت ویژه‌ای برخوردار است.

به طور کلی، این انتخاب دیتابیس به ما امکان داده تا از یک سو با چالش‌های جدید در استفاده از YOLO4روبه‌رو شویم و از سوی دیگر، از داده‌هایی بهره ببریم که به واسطه برچسب‌گذاری دقیق و پوشش‌دهی شرایط واقعی، تضمین‌کننده عملکرد بهینه مدل در دنیای واقعی هستند. این ترکیب از نوآوری، کیفیت داده و کاربردی بودن، اساس تصمیم ما برای استفاده از این دیتابیس را تشکیل می‌دهد.

منبع داده‌ها

https://universe.roboflow.com/ts-rek4f/car-dashboard-icons-gsrxy/dataset/2

ساختار داده‌ها

مجموعه داده‌های این پروژه شامل تصاویر صفحه کیلومتر خودرو است که در آن‌ها آیکون‌های هشدار مختلف روشن شده‌اند. این داده‌ها به سه مجموعه‌ی جداگانه تقسیم شده‌اند:

Train شامل داده‌های آموزشی که مدل با استفاده از آن‌ها یادگیری خود را انجام می‌دهد.

Test شامل داده‌های آزمون که پس از آموزش مدل برای ارزیابی عملکرد آن مورد استفاده قرار می‌گیرند.

Valid شامل داده‌های اعتبارسنجی که برای تنظیم هایپرپارامترهای مدل استفاده می‌شود.

هر تصویر دارای یک فایل متنی همراه با همان نام است که حاوی اطلاعات برچسب‌های YOLO می‌باشد.

مجموعه داده شامل 2406 تصویر برای آموزش (Train)، 240 تصویر برای اعتبارسنجی (Valid)، و 24 تصویر برای آزمون (Test) است. این تقسیم‌بندی به بهبود عملکرد مدل و ارزیابی دقت آن کمک می‌کند.

لیست هشدارهای شناسایی‌شده

در این پروژه، ۱۰ آیکون هشدار مختلف برای شناسایی انتخاب شده‌اند که شامل موارد زیر می‌شوند:

کد کلاس	نام هشدار
0	هشدار سیستم ترمز ABS (ABS Warning)
1	هشدار ترمز دستی
2	هشدار باتری (Battery Warning)
3	هشدار موتوری (Check Engine Light)
4	هشدار ترمز (Brake Warning)
5	هشدار دمای مایع خنک‌کننده (Coolant Temperature Warning)
6	هشدار فشار روغن (Oil Pressure Warning)
7	هشدار فشار تایر (Tire Pressure Warning)

8 هشدار کمربند ایمنی(Seatbelt Warning)

9 هشدار ایربگ(Airbag Warning)

فرمت داده‌های برچسب‌گذاری‌شده

هر فایل متنی همراه تصویر حاوی اطلاعات مربوط به مکان و نوع هشدار در تصویر است. ساختار این داده‌ها مطابق با فرمت YOLO به‌صورت زیر است:

height width y_center x_center class_id

به عنوان مثال، برای یک آیکون هشدار موتور الکتریکی، فایل متنی ممکن است شامل داده‌های زیر باشد:

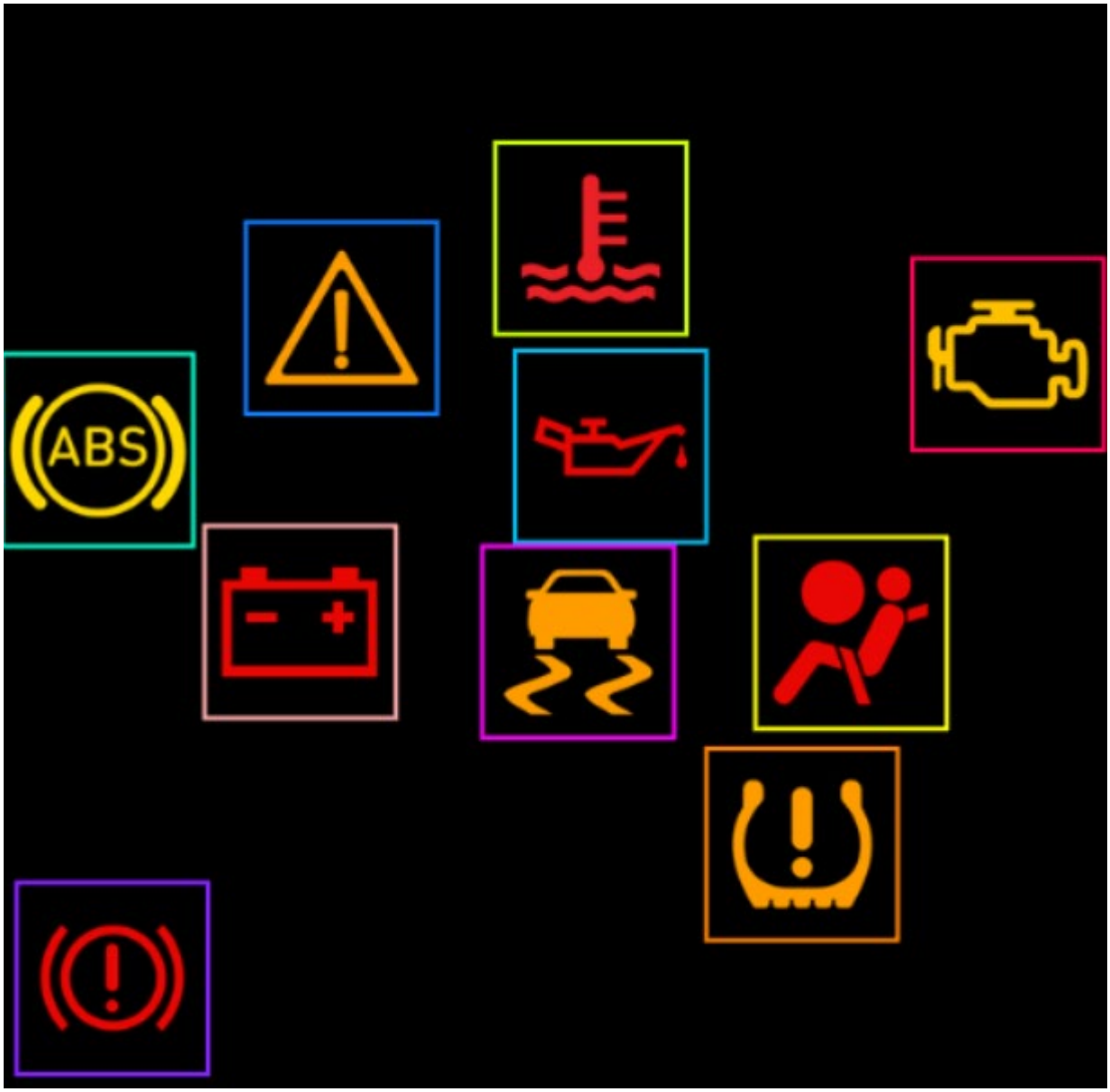
3 0.591346 0.421875 0.305288 0.349759

توضیح مقادیر:

class_id (کد کلاس): مقدار عددی که نشان‌دهنده‌ی نوع هشدار در تصویر است. در این مثال، مقدار 3 نشان می‌دهد که آیکون مربوط به هشدار ترمز است.

x_center و y_center (مختصات مرکز باکس محدودکننده): این مقادیر نشان‌دهنده‌ی موقعیت مرکز باکس تشخیص‌یافته در تصویر هستند. مقدار 0.591346 مربوط به مختصات افقی (X) و مقدار 0.421875 مربوط به مختصات عمودی (Y) است. این مقادیر نرمال‌سازی‌شده هستند و مقدار آن‌ها بین 0 تا 1 خواهد بود.

width و height (ابعاد باکس محدودکننده): این مقادیر نشان‌دهنده‌ی عرض و ارتفاع باکس محدودکننده هستند. مقدار 0.305288 عرض باکس و مقدار 0.349759 ارتفاع باکس را نشان می‌دهد. این مقادیر نیز نرمال‌سازی‌شده و نسبت به اندازه‌ی کلی تصویر داده می‌شوند.

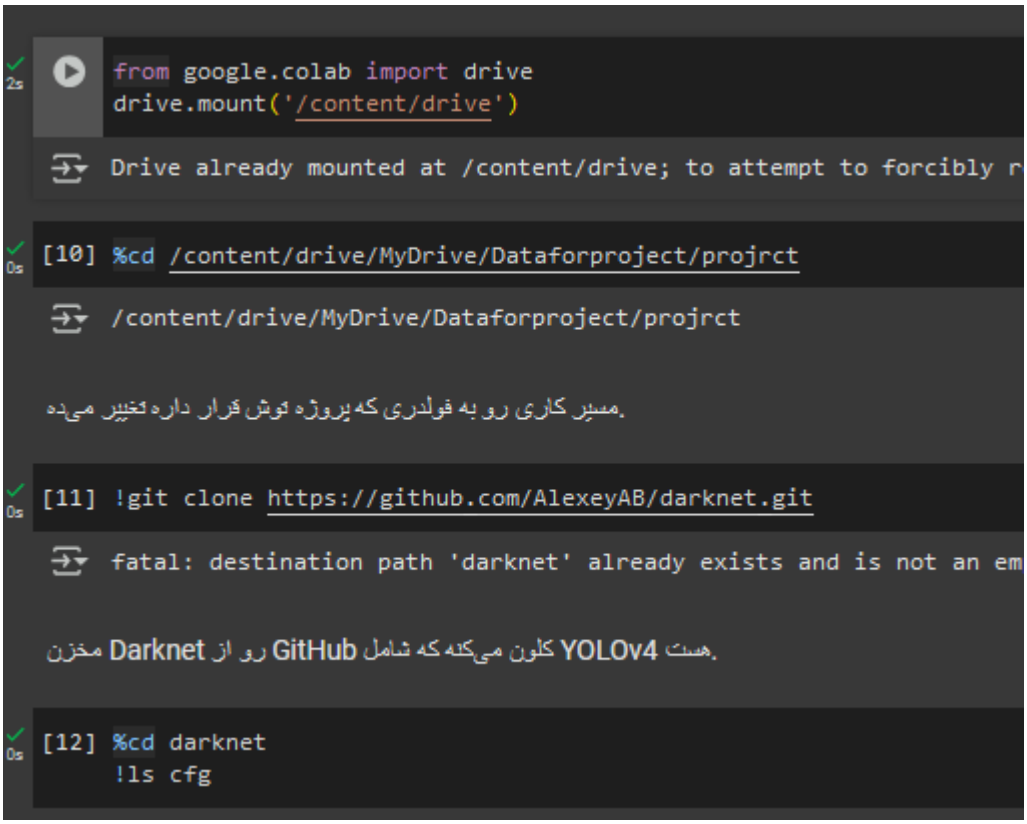


گزارش کد:

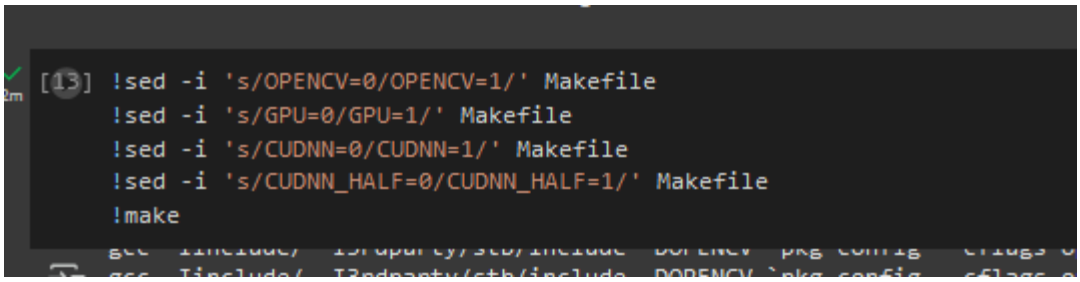
کد در محیط Google Colab ران شده. داده‌ها رو برای دسترسی بهتر به drive انتقال داده‌ایم.

ابتدا من Google Drive رو به Google Colab وصل کردم تا به فایل‌های پروژه دسترسی داشته باشم:

سپس مسیر پروژه رو به فولدری که فایل‌های پروژه در آن قرار دارند تغییر دادم. بعد از آن من مخزن Darknet رو از GitHub کلون کردم. سپس وارد پوشه Darknet شدم و فایل‌های تنظیمات (cfg) رو لیست کردم.



برای افزایش سرعت پردازش، من تغییراتی در فایل Makefile انجام دادم تا از OpenCV، GPU، CUDNN و CUDNN_HALF استفاده بشه



```
[13] !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
     !sed -i 's/GPU=0/GPU=1/' Makefile
     !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
     !sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
     !make
```

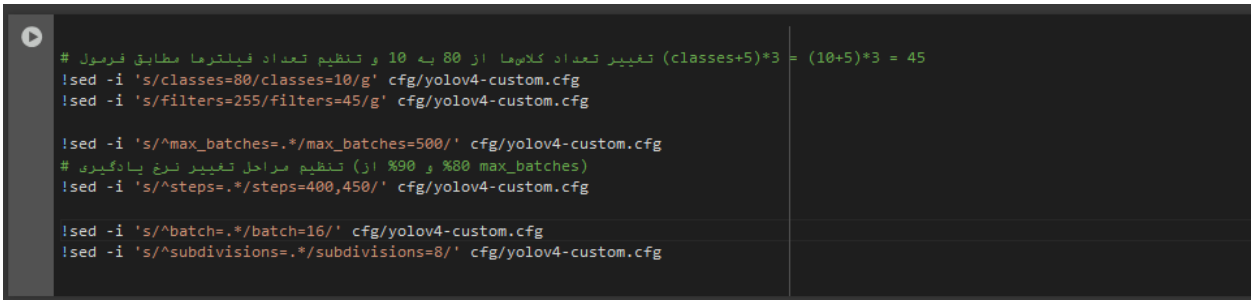
برای استفاده از دیتاست سفارشی، من ابتدا فایل پیکربندی اصلی رو کپی کردم

چون دیتاست من شامل 10 کلاس است، تعداد کلاس‌ها رو از 80 به 10 تغییر دادم.

طبق فرمول 3*(classes+5)، تعداد فیلترها رو به 45 تغییر دادم.

من تعداد کل batchها (max_batches) رو به 500 کاهش دادم تا آموزش سریع‌تر انجام بشه. معمولا MAX BATCH روی ۱۰۰۰۰ تا تنظیم میشود.

همچنین مقادیر batch و subdivisions رو به ترتیب به 8 و 4 تغییر دادم.



```
# تغییر تعداد کلاس‌ها از 80 به 10 و تنظیم تعداد فیلترها مطابق فرمول (classes+5)*3 = (10+5)*3 = 45
!sed -i 's/classes=80/classes=10/g' cfg/yolov4-custom.cfg
!sed -i 's/filters=255/filters=45/g' cfg/yolov4-custom.cfg

!sed -i 's/^max_batches=.*/max_batches=500/' cfg/yolov4-custom.cfg
# تنظیم مراحل تغییر نرخ یادگیری (از %80 و %90 max_batches)
!sed -i 's/^steps=.*/steps=400,450/' cfg/yolov4-custom.cfg

!sed -i 's/^batch=.*/batch=16/' cfg/yolov4-custom.cfg
!sed -i 's/^subdivisions=.*/subdivisions=8/' cfg/yolov4-custom.cfg
```

برای اینکه مدل بدون اشتباه بدون بره، من یک تابع نوشتم تا مسیر تمام تصاویر موجود در پوشه‌های "train" و "valid" رو استخراج کنم

سپس مسیر پوشه‌های آموزش و اعتبارسنجی رو مشخص کردم و لیست تصاویر رو تولید کردم

```python
import os

def generate_image_list(directory, output_file):
    image_extensions = ('.jpg', '.jpeg', '.png')
    with open(output_file, 'w') as f:
        for root, dirs, files in os.walk(directory):
            for file in files:
                if file.lower().endswith(image_extensions):
                    f.write(os.path.join(root, file) + "\n")

# مسیر پوشه‌های train و valid (در صورت نیاز این مسیرها را به مسیرهای صحیح تغییر دهید)
train_dir = '/content/drive/MyDrive/Dataforproject/projrct/train'
valid_dir = '/content/drive/MyDrive/Dataforproject/projrct/valid'

generate_image_list(train_dir, '/content/drive/MyDrive/Dataforproject/projrct/train.txt')
generate_image_list(valid_dir, '/content/drive/MyDrive/Dataforproject/projrct/valid.txt')

print("فایل‌های train.txt و valid.txt ایجاد شدند.")
```

من یک فایل متنی ایجاد کردم که شامل نام 10 کلاس مورد نظر (class0 تا class9) است

```
%%bash
cat <<EOF > "/content/drive/MyDrive/Dataforproject/projrct/obj.names"
class0
class1
class2
class3
class4
class5
class6
class7
class8
class9
EOF
```

سپس یک فایل دیگر به نام obj.data ایجاد کردم تا اطلاعات مربوط به تعداد کلاس‌ها، مسیر فایل‌های لیست تصاویر و نام کلاس‌ها رو به Darknet معرفی کنم.

```
[18] %%bash
     cat <<EOF > "/content/drive/MyDrive/Dataforproject/projrct/obj.data"
     classes = 10
     train = /content/drive/MyDrive/Dataforproject/projrct/train.txt
     valid = /content/drive/MyDrive/Dataforproject/projrct/valid.txt
     names = /content/drive/MyDrive/Dataforproject/projrct/obj.names
     backup = /content/drive/MyDrive/Dataforproject/projrct/backup/
     EOF
```

سپس پوشه backup رو ایجاد کردم تا وزن‌های مدل در آن ذخیره بشه

```
[19] !mkdir -p "/content/drive/MyDrive/Dataforproject/projrct/backup"
```

برای این که آموزش مدل سریع‌تر انجام بشه، من وزن‌های اولیه

YOLOv4 (pre-trained weights) رو دانلود کردم.

```
[20] !wget https://github.com/AlexeyAB/darknet/releases/download/yolov4/yolov4.conv.137
```

سپس آموزش مدل رو با استفاده از وزن‌های اولیه و تنظیمات سفارشی شروع کردم

```
!./darknet detector train "/content/drive/MyDrive/Dataforproject/projrct/obj.data" cfg/yolov4-custom.cfg yolov4.conv.137 -dont_show -map
```

تحلیل و بررسی دقت مدل و برازش مدل و توضیحات خروجی:

ابتدا با اجرای دستور زیر، مدل رو روی مجموعه داده‌های اعتبارسنجی اجرا کردم تا نتایج تشخیص را به صورت متنی مشاهده کنم. این فرمان اطلاعات جزئی از عملکرد مدل (مانند تعداد تشخیص‌های صحیح و اشتباه، جزئیات لایه‌های تشخیص و غیره) رو نمایش می‌ده.

```
!./darknet detector valid \
  "/content/drive/MyDrive/Dataforproject/projrct/obj.data" \
  cfg/yolov4-custom.cfg \
  /content/drive/MyDrive/Dataforproject/projrct/backup/yolov4-custom_last.weights \
  -dont_show -ext_output
```

خروجی:

```
CUDA-version: 12050 (12040)
 Warning: CUDA-version is higher than Driver-version!
, cuDNN: 9.2.1, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 4.5.4
results: Using default 'results'
 0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4
net.optimized_memory = 0
mini_batch = 1, batch = 4, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
reate cudnn-handle 0
conv     32       3 x 3/ 1    608 x 608 x   3 ->  608 x 608 x  32 0.639 BF
   1 conv     64       3 x 3/ 2    608 x 608 x  32 ->  304 x 304 x  64 3.407
BF
   2 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757
BF
   3 route  1                                       ->  304 x 304 x  64
   4 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757
BF
   5 conv     32       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  32 0.379
BF
   6 conv     64       3 x 3/ 1    304 x 304 x  32 ->  304 x 304 x  64 3.407
BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 304 x 304 x  64 0.006 BF
   8 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757
BF
   9 route  8 2                                     ->  304 x 304 x 128
  10 conv     64       1 x 1/ 1    304 x 304 x 128 ->  304 x 304 x  64 1.514
BF
  11 conv    128       3 x 3/ 2    304 x 304 x  64 ->  152 x 152 x 128 3.407
BF
  12 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379
BF
  13 route  11                                      ->  152 x 152 x 128
```

```
  14 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379
BF
  15 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189
BF
  16 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703
BF
  17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  18 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189
BF
  19 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703
BF
  20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  21 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189
BF
  22 route  21 12                                  ->  152 x 152 x 128
  23 conv    128       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x 128 0.757
BF
  24 conv    256       3 x 3/ 2    152 x 152 x 128 ->   76 x  76 x 256 3.407
BF
  25 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
  26 route  24                                     ->   76 x  76 x 256
  27 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
  28 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  29 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  31 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  32 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  33 Shortcut Layer: 30,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  34 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  35 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  36 Shortcut Layer: 33,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  37 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  38 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  39 Shortcut Layer: 36,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  40 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  41 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  42 Shortcut Layer: 39,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  43 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  44 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  45 Shortcut Layer: 42,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  46 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
```

```
  47 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  48 Shortcut Layer: 45,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  49 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  50 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
BF
  51 Shortcut Layer: 48,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
  52 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
BF
  53 route  52 25                                  ->   76 x  76 x 256
  54 conv    256       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 256 0.757
BF
  55 conv    512       3 x 3/ 2     76 x  76 x 256 ->   38 x  38 x 512 3.407
BF
  56 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
  57 route  55                                     ->   38 x  38 x 512
  58 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
  59 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  60 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  61 Shortcut Layer: 58,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  62 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  63 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  64 Shortcut Layer: 61,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  65 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  66 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  67 Shortcut Layer: 64,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  68 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  69 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  70 Shortcut Layer: 67,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  71 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  72 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  73 Shortcut Layer: 70,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  74 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  75 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  76 Shortcut Layer: 73,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  77 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  78 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  79 Shortcut Layer: 76,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  80 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
```

```
  81 conv    256       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 256 1.703
BF
  82 Shortcut Layer: 79,  wt = 0, wn = 0, outputs:  38 x  38 x 256 0.000 BF
  83 conv    256       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 256 0.189
BF
  84 route  83 56                                  ->   38 x  38 x 512
  85 conv    512       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 512 0.757
BF
  86 conv   1024       3 x 3/ 2     38 x  38 x 512 ->   19 x  19 x1024 3.407
BF
  87 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
  88 route  86                                     ->   19 x  19 x1024
  89 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
  90 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189
BF
  91 conv    512       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x 512 1.703
BF
  92 Shortcut Layer: 89,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
  93 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189
BF
  94 conv    512       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x 512 1.703
BF
  95 Shortcut Layer: 92,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
  96 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189
BF
  97 conv    512       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x 512 1.703
BF
  98 Shortcut Layer: 95,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
  99 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189
BF
 100 conv    512       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x 512 1.703
BF
 101 Shortcut Layer: 98,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
 102 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189
BF
 103 route  102 87                                ->   19 x  19 x1024
 104 conv   1024       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x1024 0.757
BF
 105 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 106 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 107 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 108 max               5x 5/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.005
BF
 109 route  107                                     ->   19 x  19 x 512
 110 max               9x 9/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.015
BF
 111 route  107                                     ->   19 x  19 x 512
 112 max              13x13/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.031
BF
 113 route  112 110 108 107                       ->   19 x  19 x2048
 114 conv    512       1 x 1/ 1     19 x  19 x2048 ->   19 x  19 x 512 0.757
BF
```

```
 115 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 116 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 117 conv    256       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 256 0.095
BF
 118 upsample                 2x    19 x  19 x 256 ->   38 x  38 x 256
 119 route  85                                     ->   38 x  38 x 512
 120 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 121 route  120 118                                ->   38 x  38 x 512
 122 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 123 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 124 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 125 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 126 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 127 conv    128       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 128 0.095
BF
 128 upsample                 2x    38 x  38 x 128 ->   76 x  76 x 128
 129 route  54                                     ->   76 x  76 x 256
 130 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 131 route  130 128                                ->   76 x  76 x 256
 132 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 133 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407
BF
 134 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 135 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407
BF
 136 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 137 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407
BF
 138 conv     45       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x  45 0.133
BF
 139 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm:
1.00, delta_norm: 1.00, scale_x_y: 1.20
nms_kind: greedynms (1), beta = 0.600000
 140 route  136                                    ->   76 x  76 x 128
 141 conv    256       3 x 3/ 2     76 x  76 x 128 ->   38 x  38 x 256 0.852
BF
 142 route  141 126                                ->   38 x  38 x 512
 143 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 144 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 145 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
```

```
 146 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 147 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 148 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 149 conv     45       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x  45 0.067
BF
 150 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm:
1.00, delta_norm: 1.00, scale_x_y: 1.10
nms_kind: greedynms (1), beta = 0.600000
 151 route  147                                           ->   38 x  38 x 256
 152 conv    512       3 x 3/ 2     38 x  38 x 256 ->   19 x  19 x 512 0.852
BF
 153 route  152 116                               ->   19 x  19 x1024
 154 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 155 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 156 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 157 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 158 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 159 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 160 conv     45       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x  45 0.033
BF
 161 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm:
1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 127.372
avg_outputs = 1048740
 Allocate additional workspace_size = 52.44 MB
Loading weights from
/content/drive/MyDrive/Dataforproject/projrct/backup/yolov4-
custom_last.weights...
 seen 64, trained: 14 K-images (0 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
Learning Rate: 0.0013, Momentum: 0.949, Decay: 0.0005
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
eval: Using default 'voc'
4
8
12
16
20
24
28
32
36
40
```

```
44
48
52
56
60
64
68
72
76
80
84
88
92
96
100
104
108
112
116
120
Total Detection Time: 36.000000 Seconds
```

تحلیل خروجی:

سیستم به خوبی کتابخانه‌هایCUDA ، cuDNNو OpenCV را شناسایی و استفاده کرده است.

ساختار مدل شامل 162 لایه با استفاده از لایه‌های کانولوشن، Shortcutو Route می‌باشد که نشان‌دهنده پیچیدگی و توانایی استخراج ویژگی‌های مناسب از تصاویر است.

وزن‌های مدل با موفقیت بارگذاری شده و پارامترهای یادگیری به درستی تنظیم شده‌اند.

مدل از سه لایه تشخیص (YOLO) بهره می‌برد که تنظیمات دقیقی برای بهبود عملکرد تشخیص در آن‌ها اعمال شده است.

زمان کل تشخیص‌ها 36 ثانیه گزارش شده است که نشان‌دهنده سرعت و کارایی مدل در پردازش تصاویر می‌باشد.

سپس با اجرای دستور زیر، میانگین دقت متوسط (mAP) مدل رو محاسبه کردم. این معیار، به عنوان یک شاخص کلی برای کیفیت تشخیص مدل استفاده می‌شود

```
!./darknet detector map \
/content/drive/MyDrive/Dataforproject/projrct/obj.data \
cfg/yolov4-custom.cfg \
/content/drive/MyDrive/Dataforproject/projrct/backup/yolov4-custom_last.weights \
-dont_show
```

خروجی:

```
 126 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 127 conv    128       1 x 1/ 1     38 x  38 x 256 ->   38 x  38 x 128 0.095
BF
 128 upsample                 2x    38 x  38 x 128 ->   76 x  76 x 128
 129 route  54                                     ->   76 x  76 x 256
 130 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 131 route  130 128                                ->   76 x  76 x 256
 132 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 133 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407
BF
 134 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 135 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407
BF
 136 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
BF
 137 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407
BF
 138 conv     45       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x  45 0.133
BF
 139 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm:
1.00, delta_norm: 1.00, scale_x_y: 1.20
nms_kind: greedynms (1), beta = 0.600000
 140 route  136                                    ->   76 x  76 x 128
 141 conv    256       3 x 3/ 2     76 x  76 x 128 ->   38 x  38 x 256 0.852
BF
 142 route  141 126                                ->   38 x  38 x 512
 143 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 144 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 145 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 146 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 147 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379
BF
 148 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407
BF
 149 conv     45       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x  45 0.067
BF
 150 yolo
```

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm:
1.00, delta_norm: 1.00, scale_x_y: 1.10
nms_kind: greedynms (1), beta = 0.600000
 151 route  147                                           ->   38 x  38 x 256
 152 conv    512       3 x 3/ 2     38 x  38 x 256 ->   19 x  19 x 512 0.852
BF
 153 route  152 116                                 ->   19 x  19 x1024
 154 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 155 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 156 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 157 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 158 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379
BF
 159 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407
BF
 160 conv     45       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x  45 0.033
BF
 161 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm:
1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 127.372
avg_outputs = 1048740
 Allocate additional workspace_size = 52.44 MB
Loading weights from
/content/drive/MyDrive/Dataforproject/projrct/backup/yolov4-
custom_last.weights...
 seen 64, trained: 14 K-images (0 Kilo-batches_64)
Done! Loaded 162 layers from weights-file

 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
120
 detections_count = 13451, unique_truth_count = 1060
class_id = 0, name = class0, ap = 84.43%       (TP = 106, FP = 73)
class_id = 1, name = class1, ap = 66.66%       (TP = 105, FP = 108)
class_id = 2, name = class2, ap = 89.57%       (TP = 103, FP = 61)
class_id = 3, name = class3, ap = 77.40%       (TP = 103, FP = 44)
class_id = 4, name = class4, ap = 58.08%       (TP = 97, FP = 137)
class_id = 5, name = class5, ap = 66.17%       (TP = 97, FP = 81)
class_id = 6, name = class6, ap = 67.94%       (TP = 102, FP = 98)
class_id = 7, name = class7, ap = 84.75%       (TP = 97, FP = 84)
class_id = 8, name = class8, ap = 93.04%       (TP = 98, FP = 54)
class_id = 9, name = class9, ap = 88.21%       (TP = 102, FP = 40)

 for conf_thresh = 0.25, precision = 0.56, recall = 0.95, F1-score = 0.71
 for conf_thresh = 0.25, TP = 1010, FP = 780, FN = 50, average IoU = 39.22 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.776243, or 77.62 %
Total Detection Time: 116 Seconds
```

```
Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

تحلیل خروجی و فرآیند:

در مرحله اعتبارسنجی، ابتدا فرآیند محاسبه میانگین دقت متوسط (mAP) آغاز گردید. در این مرحله، مدل با استفاده از لایه‌های تشخیص (به ترتیب لایه‌های 139، 150 و 161 با نوع 28) برای شناسایی اشیاء عمل کرد. در مجموع، ۱۳۴۵۱ تشخیص انجام شد و تعداد واقعی اشیاء (truth)برابر با ۱۰۶۰ مورد گزارش گردید.

برای هر کلاس به‌صورت جداگانه، دقت متوسط (AP) به شرح زیر محاسبه شد:

کلاس 0(class0): AP = 84.43٪ (TP = 106, FP = 73)

کلاس 1(class1): AP = 66.66٪ (TP = 105, FP = 108)

کلاس 2(class2): AP = 89.57٪ (TP = 103, FP = 61)

کلاس 3(class3): AP = 77.40٪ (TP = 103, FP = 44)

کلاس 4(class4): AP = 58.08٪ (TP = 97, FP = 137)

کلاس 5(class5): AP = 66.17٪ (TP = 97, FP = 81)

کلاس 6(class6): AP = 67.94٪ (TP = 102, FP = 98)

کلاس 7(class7): AP = 84.75٪ (TP = 97, FP = 84)

کلاس 8(class8): AP = 93.04٪ (TP = 98, FP = 54)

کلاس 9(class9): AP = 88.21٪ (TP = 102, FP = 40)

با تنظیم آستانه اطمینان (conf_thresh) بر روی 0.25، دقت کلی (precision) به 56٪ و بازیابی (recall) به 95٪ دست یافت که در نتیجه F1-score برابر با 0.71 محاسبه شد. در این سطح، تعداد تشخیص‌های صحیح (TP) برابر با 1010، تشخیص‌های نادرست (FP) برابر با 780 و موارد از دست رفته (FN) برابر با 50 گزارش گردید. همچنین، میانگین Intersection over Union (IoU)برابر با 39.22٪ به دست آمد.

در نهایت، با استفاده از آستانه IoU برابر با 50٪ و محاسبه مساحت زیر منحنی (AUC) برای هر مقدار recall منحصر به فرد، میانگین دقت متوسط (mAP@0.50) برابر با 77.62٪ محاسبه شد. زمان کل صرف شده برای انجام تشخیص‌ها نیز 116 ثانیه بود.

دلایل دقت خروجی داده شده:

در این گزارش می‌توان توضیح داد که برای افزایش سرعت پردازش مدلYOLO4 ، تعداد تصاویری که در هر دور (BATCH) به‌طور همزمان پردازش می‌شوند، نسبت به حالت استاندارد کاهش یافته است. در حالت عادی، استفاده از یک بچ بزرگتر به مدل کمک می‌کند تا با نمونه‌های متنوع‌تری از داده‌ها به‌صورت همزمان کار کند و ویژگی‌های دقیق‌تر و جزئی‌تری از تصاویر استخراج کند؛ که نتیجه آن افزایش دقت نهایی (mAP) مدل است.

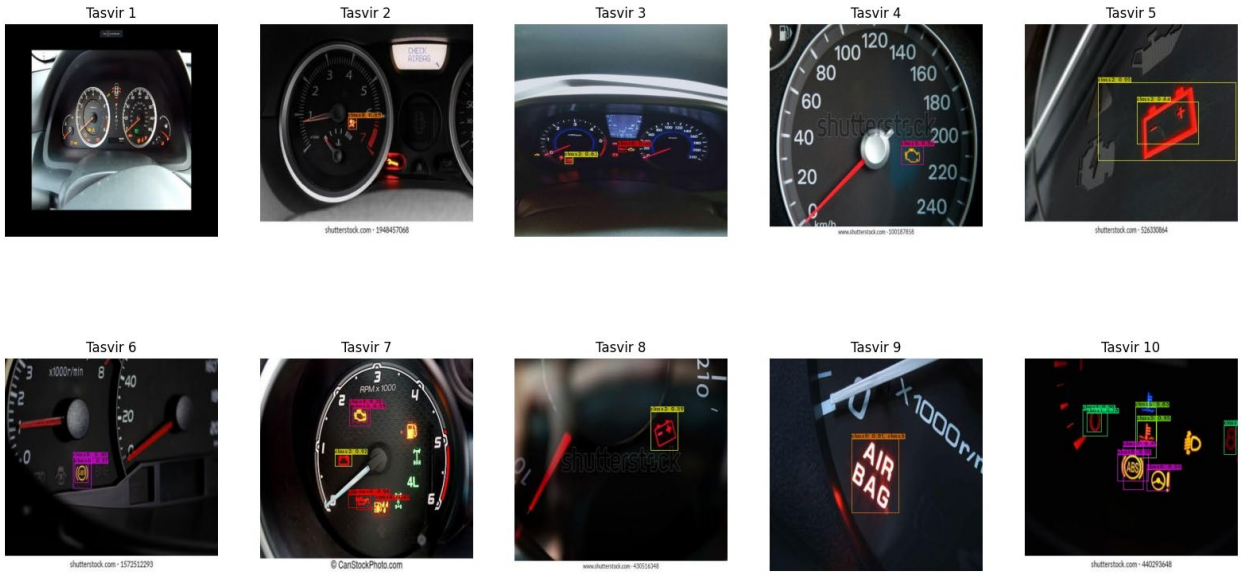
با کاهش اندازه بچ، مدل در هر تکرار تعداد کمتری از تصاویر را بررسی می‌کند. این موضوع باعث می‌شود مصرف حافظه و زمان پردازش کاهش یابد و اجرای مدل سریع‌تر صورت گیرد. اما در مقابل، مدل فرصت کافی برای یادگیری از تنوع کامل داده‌های آموزشی را ندارد و ممکن است برخی جزئیات مهم در تصاویر به درستی استخراج نشود، که نهایتاً دقت تشخیص کاهش می‌یابد.

به عبارت ساده‌تر، کاهش اندازه بچ یک راهکار مناسب برای بهبود سرعت اجرا است؛ اما این افزایش سرعت، با کاهش کمی در دقت مدل همراه است. بنابراین، در کاربردهایی که سرعت پردازش از اهمیت بالایی برخوردار است، کاهش اندازه بچ قابل قبول است؛ ولی در مسائلی که دقت تشخیص اهمیت بیشتری دارد، توصیه می‌شود از اندازه بچ استاندارد استفاده شود تا مدل بتواند به بهترین شکل از داده‌های آموزشی بهره ببرد.نمونه‌ی تشخیص مدل:

کلاس به درستی تشخیص داده شد ولی احتمال پایین‌است که دلیلش میتواند نوع مدلی هست که ما طراحی کرده ایم. تنظیمات را جوری تنظیم کرده بودم که زمان زیادی اجرای کد نگیرد.

۱۰ تا تصویر از فایل تست برداشته شده و تلاش شده تا نمونه‌ای از اجرا

object detection



در اینجا خطاهای مدل و نقاط ضعف مدل به این شکل مشخص میشود که در تصویر ۱ ما هیچ شناسایی رو توسط مدل نداشتیم به نظر میرسد دلیلش کیفیت پایین تصویر باشد. ولی باز ما نیاز

داریم با بررسی و اصلاح دادگان و استفاده از متد دیتافزونی با محو کردن داده‌های آموزش مدل را اصلاح کنیم.

در تصویر ۱۰ چند نمونه اشتباه در تشخیص مشخص شده است شکل ناشناسی  که شبیه علامت هشدار ABS است به اشتباه به عنوان هشدار ABS شناسایی شده است.  در شکل ۷ هم خطای عدم تشخیص داریم به نظر میرسد این خطای ذاتی مدل باشد.