



دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گزارش پژوهشی دوم یادگیری عمیق

Hand Detection

پدیدآورندگان:

رادمهر آفاخانی

۴۰۰۳۶۶۳۰۰۲

محمد امین کیانی

۴۰۰۳۶۱۳۰۵۲

دانشجویان کارشناسی، دانشکده کامپیوتر، دانشگاه اصفهان، اصفهان.

استاد درس: جناب اقای دکتر کیانی

نیمسال اول تحصیلی ۱۴۰۳-۰۴

فهرست مطالب

۳	مستندات
۳	۱-مسئله و تحلیل کلی آن:
۴	۲-تهیهی دیتاست:
۸	۳-آموزش مدل:
۱۶	۴-اعتبارسنجی و تست مدل:
۲۵	۵-متريکها و ارزیابی مدل:
۳۲	۷-نتایج:
۴۰	۸- منابع

مستندات

۱- مسئله و تحلیل کلی آن:

تشخیص حالت دست یکی از زمینه‌های پردازش تصویر و بینایی ماشین است که کاربردهای زیادی در فناوری‌های روز دارد. استفاده از مدل (YOLO You Only Look Once) به عنوان یک مدل سریع و دقیق برای شناسایی اشیاء، می‌تواند به طور مؤثری در تشخیص حالت دست به کار گرفته شود.

اهداف:

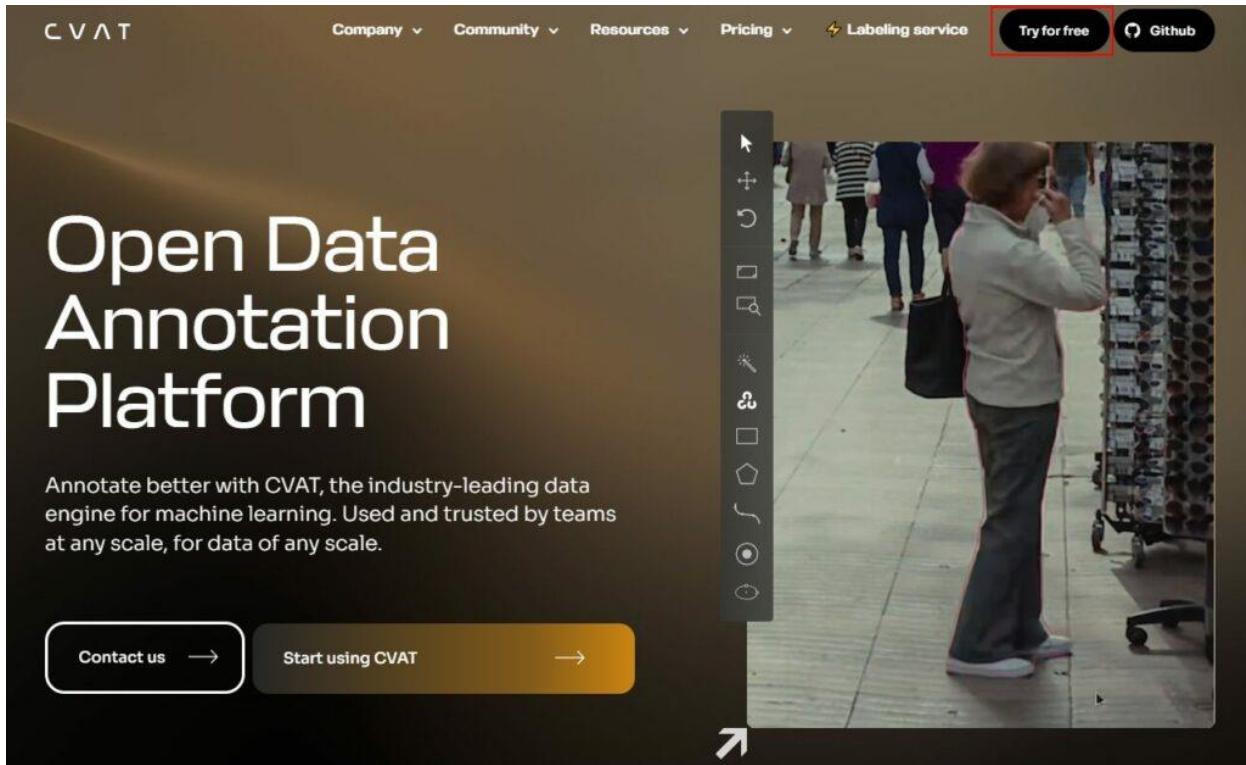
۱. شناسایی دقیق حالت‌های مختلف دست (باز، بسته، اشاره، و غیره).
۲. پیاده‌سازی الگوریتم YOLO برای تشخیص حالت دست در زمان واقعی.
۳. ارزیابی دقت و سرعت مدل در شرایط مختلف محیطی.

مراحل پیاده‌سازی:

۱. جمع‌آوری داده‌ها
۲. پیش‌پرداخت داده‌ها
۳. آموزش YOLO
۴. آموزش مدل
۵. تشخیص و ارزیابی

۲- تهیه‌ی دیتاست:

سایت CVAT (Computer Vision Annotation Tool) یک ابزار کاربرپسند و متن باز برای لیبل زدن داده‌های بصری است که به طور ویژه برای کمک به پروژه‌های بینایی ماشین طراحی شده است. این ابزار به پژوهشگران و توسعه‌دهندگان کمک می‌کند تا به راحتی داده‌های تصویر و ویدئو را برای آموزش مدل‌های یادگیری ماشین برچسب‌گذاری کنند.



ویژگی‌های CVAT برای لیبل زدن حالات دست:

۱. رابط کاربری گرافیکی (GUI):

- CVAT دارای یک رابط کاربری کاربرپسند است که کاربران می‌توانند به راحتی با آن کار کنند و داده‌ها را پس از ساخت پروژه و تعریف تسهک‌های مربوطه، بارگذاری کنند.

۲. انواع ابزارهای برچسب‌گذاری:

- این ابزار امکان ایجاد برچسب‌های مختلف مانند **جعبه‌های محاط** کننده، نقشه‌های بصری (Key Points) و نقاط کلیدی (Polygons) و **(Bounding Boxes)** را برای تشخیص حالات مختلف دست فراهم می‌کند.

۳. پشتیبانی از ویدئو:

- قابلیت **annotating** ویدئوها به کاربران اجازه می‌دهد تا حالات دست را در شرایط مختلف ضبط کنند و بر اساس زمان برچسب‌گذاری کنند.

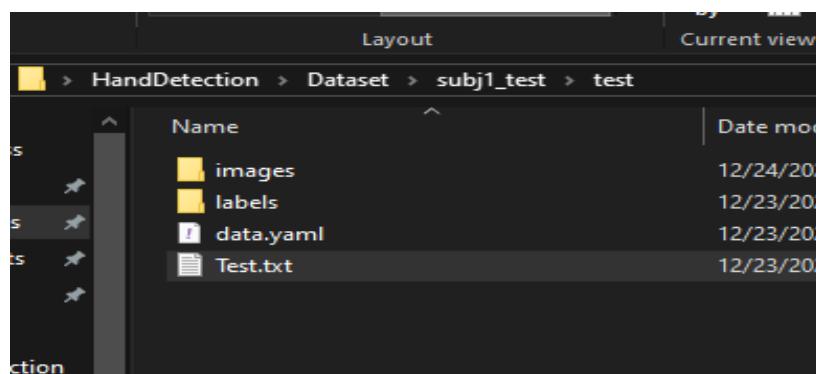
۴. مدیریت پروژه:

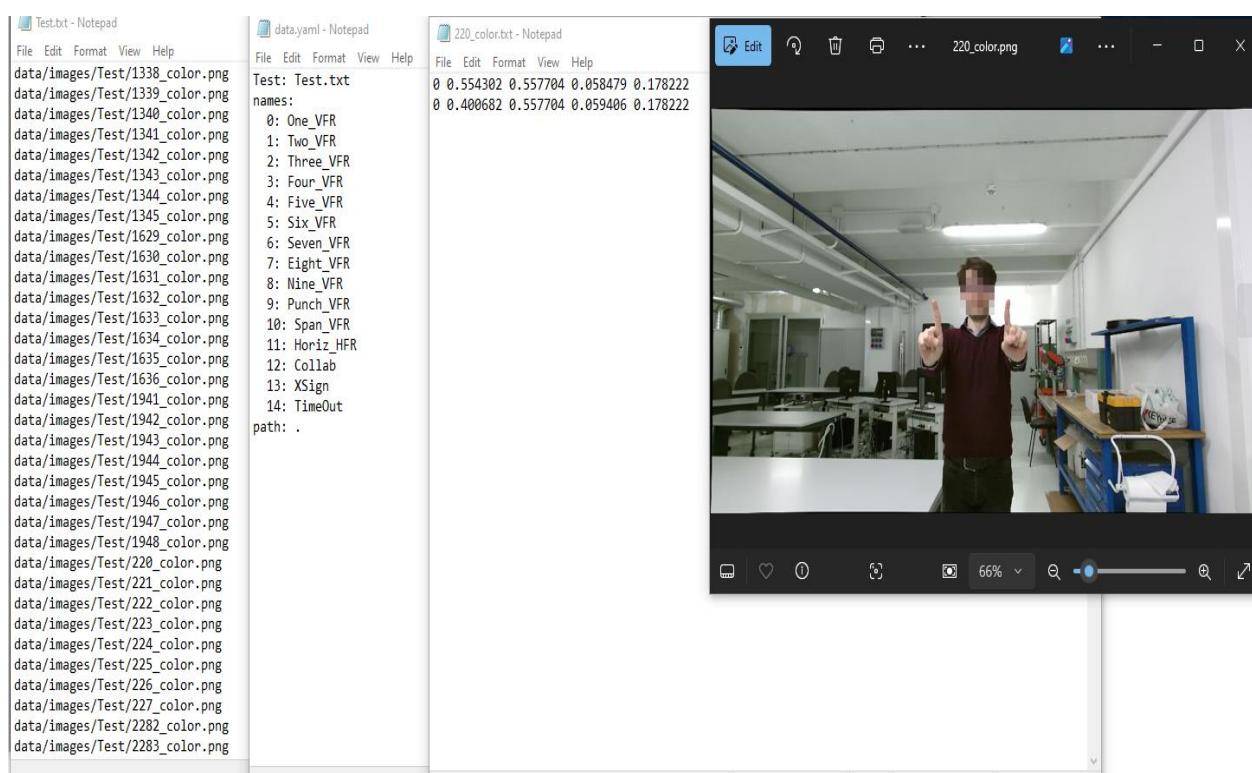
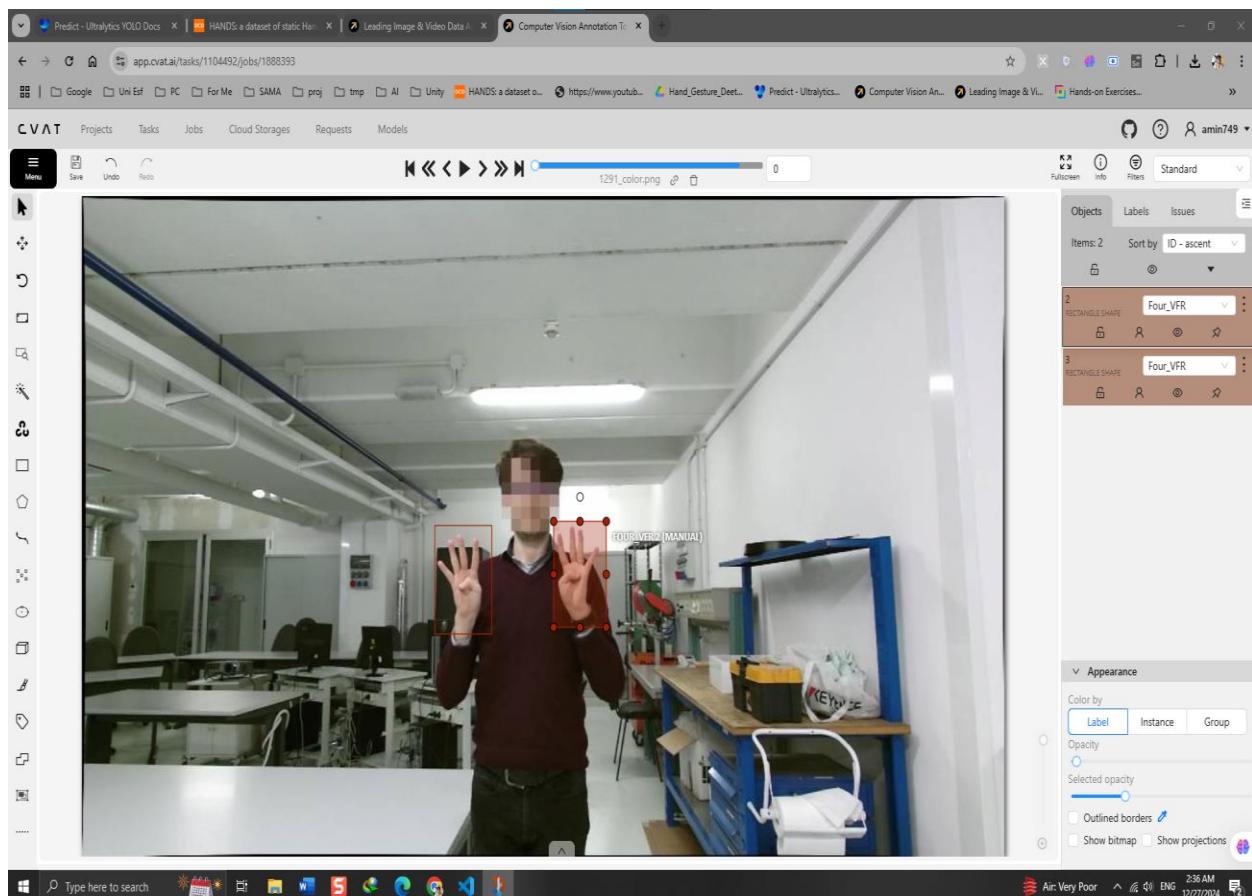
- کاربران می‌توانند پروژه‌های مختلف را مدیریت کنند، تیم‌ها را تشکیل دهند و بر روی داده‌های مشابه به صورت مشترک کار کنند.

۵. امکانات صادرات داده:

- بعد از لیبل زدن، می‌توان داده‌های آماده را به فرمتهای مختلفی (مثل YOLO، COCO و...) صادر کرد که برای آموزش مدل‌های یادگیری عمیق مناسب است. که ما **خروجی یولو ۸ نانو** را گرفتیم.

در نهایت با ساخت یک پوشه‌ی لیبل از ویژگی‌های هر عکس که شامل موارد زیر است و همچنین ساخت یک فایل **.yaml**. که تحويل مدل یولو می‌دهد، دیتابستی را برای فرآیندهای آموزش و اعتبارسنجی و تست تهیه می‌کند.





* مطابق تصاویر صفحه‌ی قبلی، فایل `data.yaml` برای مدل یولو (YOLO) اطلاعات مهمی در مورد داده‌ها و تنظیمات مربوط به آموزش مدل را فراهم می‌کند. این فایل معمولاً شامل موارد زیر است:

۱. مسیر به داده‌ها: شامل مسیرهای آدرس تصاویر و `notations` (برچسب‌ها) است.
۲. تعداد کلاس‌ها: مشخص می‌کند که چند کلاس در دیتاست وجود دارد.
۳. نام کلاس‌ها: لیستی از نام‌های کلاس‌ها که مدل باید شناسایی کند.
۴. تنظیمات دیگر: ممکن است بسته به ورزش یولوی استفاده شده، شامل تنظیمات مربوط به آموزش، نظیر تعداد `batch`, نرخ یادگیری و غیره باشد.

این اطلاعات به مدل یولو کمک می‌کند تا داده‌ها را به درستی بارگذاری کند و یادگیری را بر مبنای آن انجام دهد. پس از لیبل‌گذاری، کاربر باید مطمئن شود که فایل `data.yaml` به درستی تنظیم شده باشد تا مدل بتواند به درستی یاد بگیرد و عملکرد بهتری داشته باشد.

- * در خروجی لیبل‌گذاری `cvat` هر خط نمایانگر یک باندینگ باکس است که مختصات، ابعاد و نوع آن مشخص شده است و هر خط شامل چندین عدد است که به ترتیب به این ویژگی‌ها اشاره دارند:
۱. عدد اول (۰): معمولاً این عدد نمایانگر کلاس شیء است. در اینجا برای نوع خاصی از مدل یا شیء، اشاره دارد (مثلاً حالت دست).
 ۲. عدد دوم (۰.۵۵۴۳۰۲): مختصات X مرکز باندینگ باکس در فضای تصویر، به صورت نسبی (نسبت به عرض تصویر).
 ۳. عدد سوم (۰.۵۵۷۷۰۴): مختصات Y مرکز باندینگ باکس در فضای تصویر، به صورت نسبی (نسبت به ارتفاع تصویر).
 ۴. عدد چهارم (۰.۰۵۸۴۷۹): عرض باندینگ باکس به صورت نسبی (نسبت به عرض تصویر).
 ۵. عدد پنجم (۰.۱۷۸۲۲۲): ارتفاع باندینگ باکس به صورت نسبی (نسبت به ارتفاع تصویر).

۳-آموزش مدل:

```
# -*- coding: utf-8 -*-
"""Train_Phase_On_Colab.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1vTdAo5Dh-3KTL7tAioI3ZdP5tkuJG3Bp
"""

!pip install ultralytics==8.2.103 -q

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

from ultralytics import YOLO
model = YOLO("yolov8n.pt")
model.info()
# Train the model

from google.colab import drive
drive.mount('/content/drive')

train_results = model.train(
    data="/content/drive/MyDrive/Yolo/data/data.yaml",           # Path
    to your dataset YAML
    epochs=500,                                              # Total training epochs
    batch=16,                                                 # Image size
    device="cuda",
    save_period = 20
)

import shutil
import os
save_dir = "/content/drive/MyDrive/YOLO_Checkpoints"
weights_dir = "/content/runs/detect/train/weights"
shutil.copy(os.path.join(weights_dir, "last.pt"), save_dir)
shutil.copy(os.path.join(weights_dir, "best.pt"), save_dir)

save_dir = "/content/drive/MyDrive/final_model"
os.makedirs(save_dir, exist_ok=True)
```

```

model_path = os.path.join(save_dir, "final_model.pt")
model = model.save(model_path)

from google.colab import drive
drive.mount('/content/drive')

import shutil
import os
save_dir = '/content/drive/MyDrive/YOLO_Runs'
os.makedirs(save_dir, exist_ok=True)

# Copy the results to Google Drive
shutil.copytree("/content/runs", save_dir, dirs_exist_ok=True)
print(f"Training results saved to {save_dir}")

# import shutil
# import os
# if os.path.exists("/content/runs"):
#     shutil.rmtree("/content/runs")
#     print("Cleaned runs directory.")

train_results

```

آموزش مدل YOLO (You Only Look Once) برای تشخیص حالات دست :

۱. نصب بسته‌های ضروری

```
!pip install ultralytics==8.2.1 -q
```

این خط بسته ultralytics را نصب می‌کند که شامل پیاده‌سازی YOLO است.

۲. پاکسازی نمایشگر

```

from IPython import display

display.clear_output()

```

این کد صفحه خروجی را پاک می‌کند تا اطلاعات اضافی از نمایش قبلی حذف شود.

۳. بررسی نصب **ultralytics**

```
import ultralytics  
ultralytics.checks()
```

این قسمت اطمینان پیدا می کند که بسته به درستی نصب شده و همه چیز آماده است.
(Ultralytics YOLOv8.2.10.3 Python-3.10.12 torch-2.5.1+cu121 CUDA:: (Tesla T4, 15102MiB) Setup complete (2 CPUs, 12.7 GB RAM, 32.7/112.6 GB disk))

۴. بارگذاری مدل **YOLO**

```
from ultralytics import YOLO  
model = YOLO("yolov8n.pt")  
model.info()
```

در اینجا مدل YOLO با وزن های پیش ساخته بارگذاری می شود و اطلاعاتی از مدل نمایش داده می شود.
YOLOv8n summary: 225 layers, 3,157,200 parameters, 0 gradients, 0 GFLOPs

۵. اتصال به **Google Drive**

```
from google.colab import drive  
drive.mount('/content/drive')
```

این کد باعث می شود که Google Drive به محیط Colab متصل شود تا بتوانید فایل ها را ذخیره و بارگذاری کنید.

۶. آموزش مدل

```
train_results = model.train(  
    data="/content/drive/MyDrive/Yolo/data/data.yaml",  
    epochs=500,  
    batch=16,  
    device="cuda",  
    save_period=20)
```

در این بخش مدل آغاز به آموزش می‌کند:

: مسیر فایل YAML شامل تنظیمات و آدرس‌های دیتابست. data -

: تعداد دوره‌های آموزشی (۵۰۰ دوره). epochs -

: اندازه دسته (۱۶). batch -

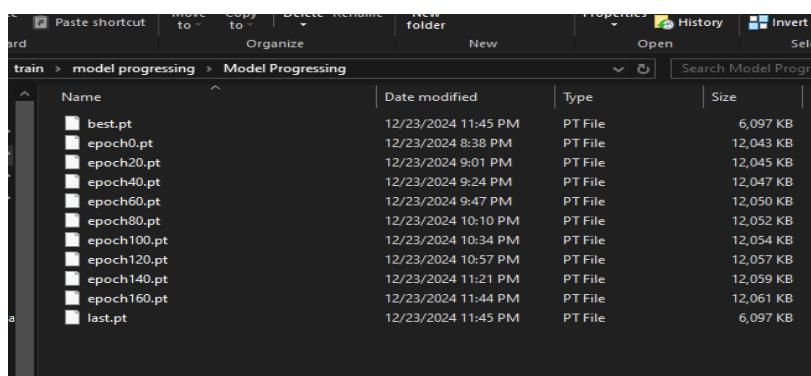
: مشخص می‌کند که آموزش بر روی GPU باشد. device -

: هرچند دوره یک نسخه از مدل ذخیره می‌شود (۲۰ دوره). save_period -

	from	n	params	module
0	-1	1	464	ultralytics.nn.modules.conv.Conv
1	-1	1	4672	ultralytics.nn.modules.conv.Conv
2	-1	1	7360	ultralytics.nn.modules.block.C2f
3	-1	1	18560	ultralytics.nn.modules.conv.Conv
4	-1	2	49664	ultralytics.nn.modules.block.C2f
5	-1	1	73984	ultralytics.nn.modules.conv.Conv
6	-1	2	197632	ultralytics.nn.modules.block.C2f
7	-1	1	295424	ultralytics.nn.modules.conv.Conv
8	-1	1	460288	ultralytics.nn.modules.block.C2f
9	-1	1	164608	ultralytics.nn.modules.block.SPP
10	-1	1	0	torch.nn.modules.upsampling.Upsample
11	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat
12	-1	1	148224	ultralytics.nn.modules.block.C2f
13	-1	1	0	torch.nn.modules.upsampling.Upsample
14	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat
15	-1	1	37248	ultralytics.nn.modules.block.C2f
16	-1	1	36992	ultralytics.nn.modules.conv.Conv
17	[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat
18	-1	1	123648	ultralytics.nn.modules.block.C2f

۷. کپی کردن وزن‌های آموزش دیده

```
import shutil  
import os  
  
save_dir = "/content/drive/MyDrive/YOLO_Checkpoints"  
  
weights_dir = "/content/runs/detect/train/weights"  
  
shutil.copy(os.path.join(weights_dir, "last.pt"), save_dir)  
  
shutil.copy(os.path.join(weights_dir, "best.pt"), save_dir)
```



این کد وزن‌های مدل (آخرین و بهترین) را به یک پوشه در Google Drive کپی می‌کند تا از آنها در ادامه استفاده شود.

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95): 100%	8/8 [00:02<00:00, 2.94it/s]
all	240	430	0.73	0.794	0.848	0.433	
128/500	2.23G	1.395	0.5334	1.21	23	640: 100% ██████████ 152/152 [01:04<00:00, 2.36it/s]	
	Class Images	Instances	Box(P)	R	mAP50	mAP50-95): 100%	8/8 [00:02<00:00, 2.87it/s]
	all	240	430	0.71	0.76	0.83	0.424
129/500	2.23G	1.4	0.5387	1.198	36	640: 100% ██████████ 152/152 [01:09<00:00, 2.19it/s]	
	Class Images	Instances	Box(P)	R	mAP50	mAP50-95): 100%	8/8 [00:04<00:00, 1.98it/s]
	all	240	430	0.809	0.694	0.81	0.415
130/500	2.23G	1.394	0.5494	1.203	41	640: 100% ██████████ 152/152 [01:04<00:00, 2.36it/s]	
	Class Images	Instances	Box(P)	R	mAP50	mAP50-95): 100%	8/8 [00:03<00:00, 2.27it/s]
	all	240	430	0.813	0.693	0.847	0.422
131/500	2.23G	1.4	0.5447	1.209	32	640: 100% ██████████ 152/152 [01:04<00:00, 2.35it/s]	
	Class Images	Instances	Box(P)	R	mAP50	mAP50-95): 100%	8/8 [00:03<00:00, 2.42it/s]
	all	240	430	0.745	0.67	0.716	0.375
132/500	2.25G	1.398	0.5435	1.216	35	640: 100% ██████████ 152/152 [01:07<00:00, 2.25it/s]	
	Class Images	Instances	Box(P)	R	mAP50	mAP50-95): 100%	8/8 [00:03<00:00, 2.39it/s]
	all	240	430	0.703	0.789	0.847	0.45

۸. ذخیره مدل نهایی

```
save_dir = "/content/drive/MyDrive/final_model"  
  
os.makedirs(save_dir, exist_ok=True)  
  
model_path = os.path.join(save_dir, "final_model.pt")  
  
model = model.save(model_path)
```

مدل نهایی آموزش دیده در این پوشه ذخیره می شود.

۹. دوباره اتصال به Google Drive

```
from google.colab import drive  
  
drive.mount('/content/drive')
```

دوباره Google Drive را متصل می شویم (اگر قبلاً قطع شده باشد).

۱۰. کپی کردن نتایج آموزش

```
save_dir = '/content/drive/MyDrive/YOLO_Runs'  
  
os.makedirs(save_dir, exist_ok=True)  
  
shutil.copytree("/content/runs", save_dir, dirs_exist_ok=True)  
  
print(f"Training results saved to {save_dir}")
```

نتایج آموزش (شامل گرافها و گزارش‌ها) به یک پوشه در Google Drive منتقل می شود.

۱۱. اختیاری - کامنت شده) پاکسازی پوشه

```
#import shutil  
  
#import os
```

```
#if os.path.exists("/content/runs"):

    #shutil.rmtree("/content/runs")

    #print("Cleaned runs directory.")
```

این قسمت به طور اختیاری پوشه نتایج را پاک می کند تا فضا آزاد شود.

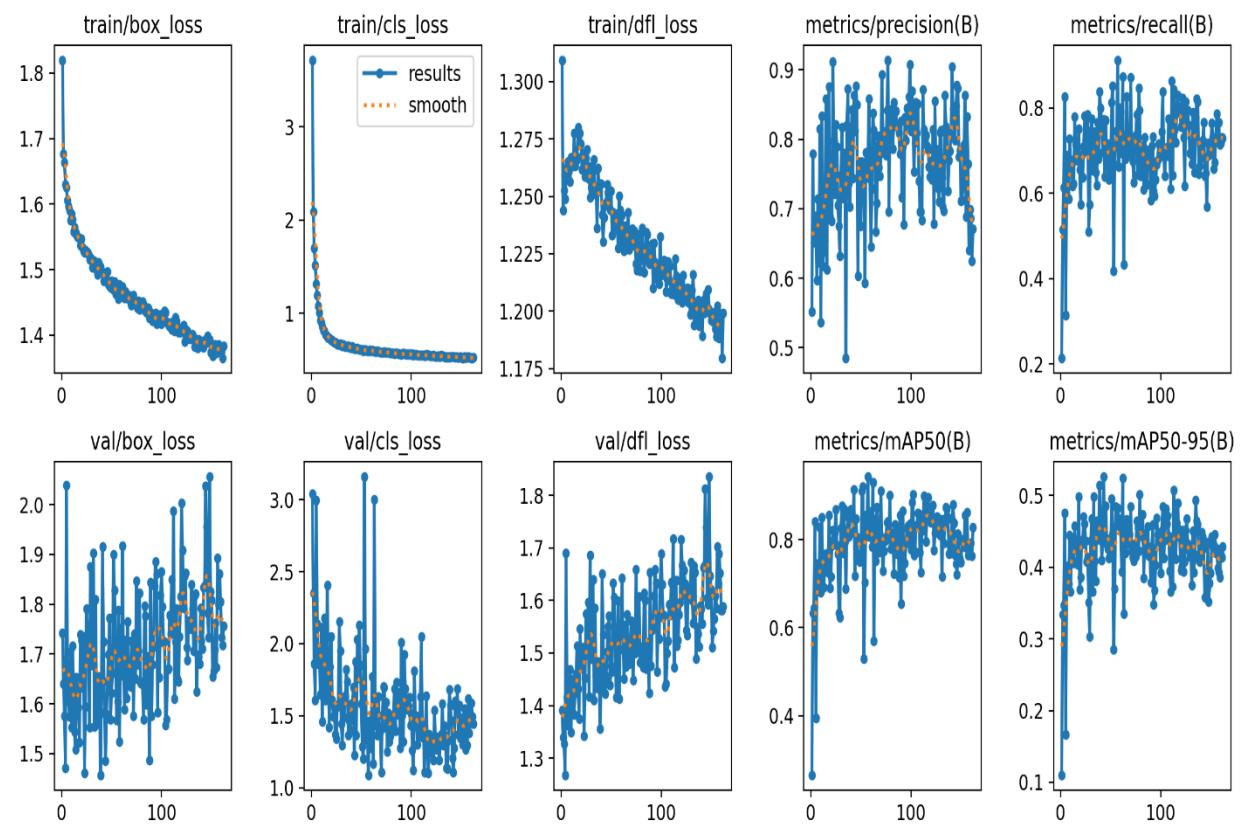
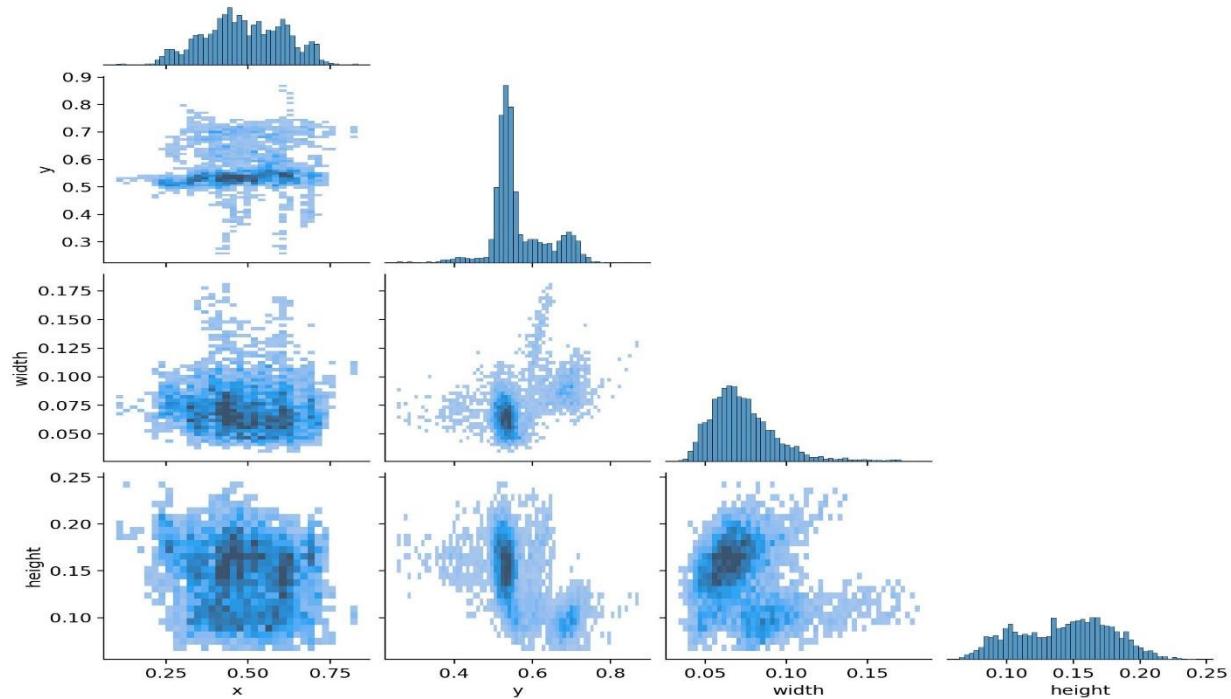
۱۲. نمایش نتایج آموزش

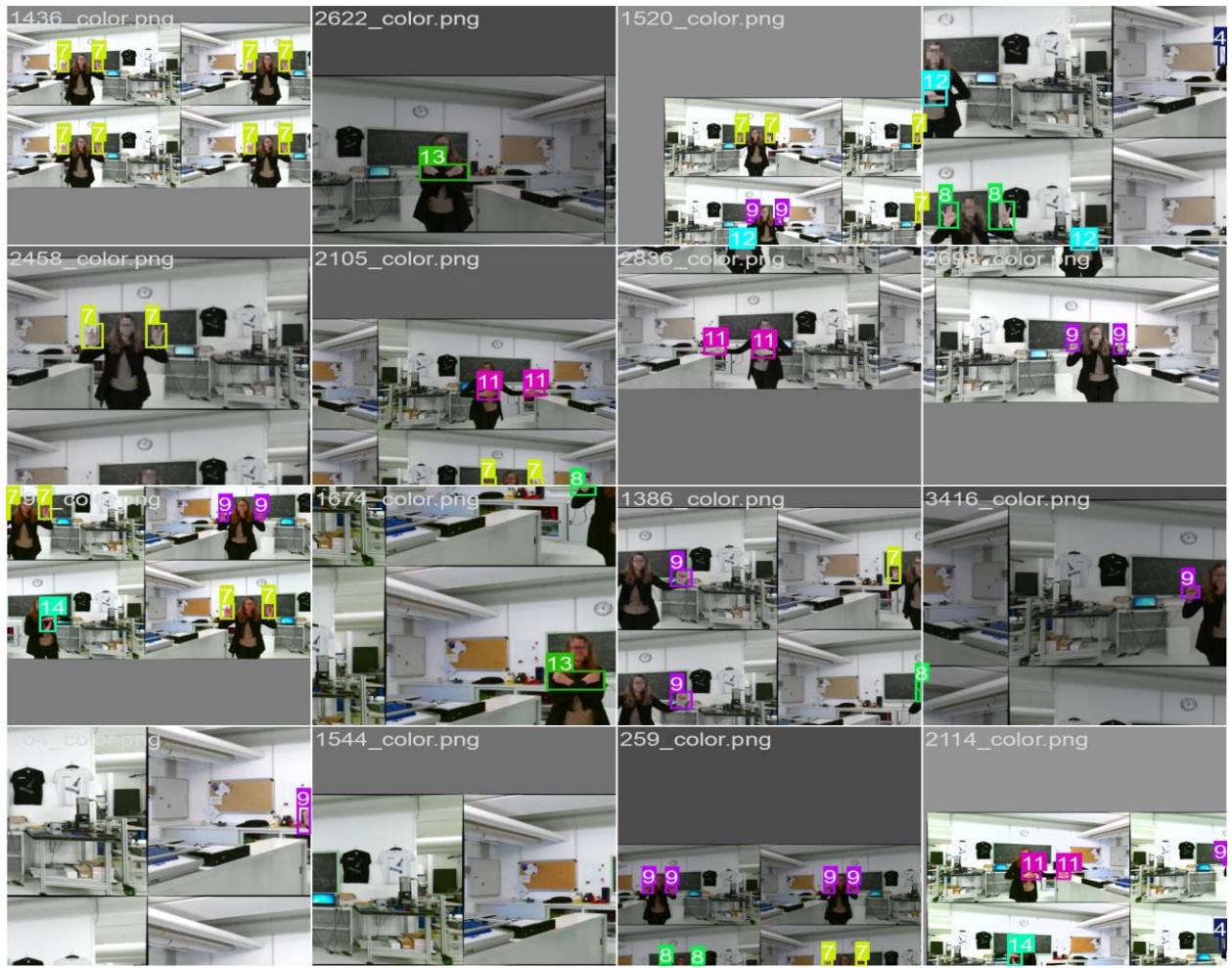
`train_results`

در نهایت نتایج آموزش نمایش داده می شود.

```
... ultralytics.utils.metrics.DetMetrics object with attributes:

ap_class_index: array([ 0,  1,  2,  3,  4,  6,  7,  8,  9, 10, 11, 12, 13, 14])
box: ultralytics.utils.metrics.Metric object
confusion_matrix: <ultralytics.utils.metrics.ConfusionMatrix object at 0x7deb705ae200>
curves: ['Precision-Recall(B)', 'F1-Confidence(B)', 'Precision-Confidence(B)', 'Recall-Confidence(B)']
curves_results: [[array([
    0, 0.001001, 0.002002, 0.003003, 0.004004, 0.005005, 0.006006, 0.007007, 0.008008, 0.009009,
    0.024024, 0.025025, 0.026026, 0.027027, 0.028028, 0.029029, 0.03003, 0.031031, 0.032032, 0.033033, 0.034034, 0.035
    0.048048, 0.049049, 0.05005, 0.051051, 0.052052, 0.053053, 0.054054, 0.055055, 0.056056, 0.057057, 0.058058, 0.059
    0.072072, 0.073073, 0.074074, 0.075075, 0.076076, 0.077077, 0.078078, 0.079079, 0.08008, 0.081081, 0.082082, 0.083
    0.096096, 0.097097, 0.098098, 0.099099, 0.1001, 0.1011, 0.1021, 0.1031, 0.1041, 0.10511, 0.10611, 0.10
    0.12012, 0.12112, 0.12212, 0.12312, 0.12412, 0.12513, 0.12613, 0.12713, 0.12813, 0.12913, 0.13013, 0.13
    0.14414, 0.14515, 0.14615, 0.14715, 0.14815, 0.14915, 0.15015, 0.15115, 0.15215, 0.15315, 0.15415, 0.15
    0.16817, 0.16917, 0.17017, 0.17117, 0.17217, 0.17317, 0.17417, 0.17518, 0.17618, 0.17718, 0.17818, 0.17
    0.19219, 0.19319, 0.19419, 0.1952, 0.1962, 0.1972, 0.1982, 0.1992, 0.2002, 0.2012, 0.2022, 0.20
    0.21622, 0.21722, 0.21822, 0.21922, 0.22022, 0.22122, 0.22222, 0.22322, 0.22422, 0.22523, 0.22623, 0.22
    0.24024, 0.24124, 0.24224, 0.24324, 0.24424, 0.24525, 0.24625, 0.24725, 0.24825, 0.24925, 0.25025, 0.25
    0.26426, 0.26527, 0.26627, 0.26727, 0.26827, 0.26927, 0.27027, 0.27127, 0.27227, 0.27327, 0.27427, 0.27
    0.28829, 0.28929, 0.29029, 0.29129, 0.29229, 0.29329, 0.29429, 0.2953, 0.2963, 0.2973, 0.2983, 0.2
    0.31231, 0.31331, 0.31431, 0.31532, 0.31632, 0.31732, 0.31832, 0.31932, 0.32032, 0.32132, 0.32232, 0.32
    0.33634, 0.33734, 0.33834, 0.33934, 0.34034, 0.34134, 0.34234, 0.34334, 0.34434, 0.34535, 0.34635, 0.34
    0.36036, 0.36136, 0.36236, 0.36336, 0.36436, 0.36537, 0.36637, 0.36737, 0.36837, 0.36937, 0.37037, 0.37
    0.38438, 0.38539, 0.38639, 0.38739, 0.38839, 0.38939, 0.39039, 0.39139, 0.39239, 0.39339, 0.39439, 0.3
    0.40841, 0.40941, 0.41041, 0.41141, 0.41241, 0.41341, 0.41441, 0.41542, 0.41642, 0.41742, 0.41842, 0.41
    0.43243, 0.43343, 0.43443, 0.43544, 0.43644, 0.43744, 0.43844, 0.43944, 0.44044, 0.44144, 0.44244, 0.44
    ...
plot: True
results_dict: {'metrics/precision(B)': 0.8371658875281166, 'metrics/recall(B)': 0.8733092329496073, 'metrics/mAP0(B)': 0.9301524196894155, 'metrics/mAP50-95(
save_dir: PosixPath('runs/detect/train')
speed: {'preprocess': 0.2554893493652344, 'inference': 2.1025816599527993, 'loss': 0.008342663447062174, 'postprocess': 3.553652763366699}
task: 'detect'
```





۴- اعتبارسنجی و تست مدل:

در ادامه پس از آموزش مدل و انتخاب بهترین pt. آن را سیو و برای ارزیابی استفاده می‌کنیم تا به صورت real time و با وبکم نیز میزان توانایی تشخیص آن سنجیده شود.

```
# -*- coding: utf-8 -*-
"""Hand_Detection_Val_Test_Track.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1eYiME8Ivb5th0LyomVq9CzKPmP7y2_YO
"""
```

```

from ultralytics import YOLO

# # Load a model
model = YOLO("final_model.pt")

validation_results =
model.val(data="C:\\Users\\Lenovo\\Desktop\\data\\data.yaml", imgsz=640,
batch=16, iou=0.6, device="cpu")

from ultralytics import YOLO

# # Load a model
model = YOLO("final_model.pt")
results = model.predict('E:\\2\\2804_color.png' , conf = .2)
# Process results list
for result in results:
    boxes = result.boxes # Boxes object for bounding box outputs
    masks = result.masks # Masks object for segmentation masks outputs
    keypoints = result.keypoints # Keypoints object for pose outputs
    probs = result.probs # Probs object for classification outputs
    obb = result.obb # Oriented boxes object for OBB outputs
    result.show() # display to screen
    result.save(filename="result.jpg") # save to disk

import cv2

from ultralytics import YOLO
model = YOLO("final_model.pt")
# Load the YOLO11 model
# model = YOLO("yolo11n.pt")

# Open the video file
video_path = "f.mp4"
cap = cv2.VideoCapture(0)

# Loop through the video frames
while cap.isOpened():
    # Read a frame from the video
    success, frame = cap.read()

    if success:
        # Run YOLO11 tracking on the frame, persisting tracks between frames
        results = model.track(frame )

        # Visualize the results on the frame

```

```

annotated_frame = results[0].plot()

# Display the annotated frame
cv2.imshow("model Tracking", annotated_frame)

# Break the Loop if 'q' is pressed
if cv2.waitKey(1) & 0xFF == ord("q"):
    break
else:
    # Break the Loop if the end of the video is reached
    break

# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()

```

from ultralytics import YOLO

- وارد کردن **YOLO**: این خط کتابخانه Ultralytics را برای استفاده از مدل YOLO (You Only Look Once) وارد می‌کند.

model = YOLO("final_model.pt")

- بارگذاری مدل: مدل نهایی (با نام final_model.pt) بارگذاری می‌شود. این مدل را قبلاً در بخش قبلی آموزش دادیم و آماده پیش‌بینی می‌باشد.

validation_results =
model.val(data="C:\\\\Users\\\\Lenovo\\\\Desktop\\\\data\\\\data.yaml",
imgsz=640, batch=16, iou=0.6, device="cpu")

- اعتبارسنجی مدل: این خط مدل را با استفاده از داده‌های مشخص شده در فایل YAML (که شامل اطلاعات مربوط به مجموعه داده است) اعتبارسنجی می‌کند.

- $\text{imgsz}=640$: اندازه تصویر را 640×640 پیکسل تنظیم می‌کند.

- $\text{batch}=16$: تعداد تصاویری که در هر بار پردازش می‌شوند ۱۶ عدد است. (سرجمع ۲۴۰ تا حاصل از ۱۵ تا کلاس با ۱۶ نمونه)

- "device="cpu": مدل بر روی CPU اجرا می‌شود.

- $\text{iou}=0.6$: ضریب همپوشانی (Intersection over Union) برای تعیین صحت پیش‌بینی‌ها.

(ضریب همپوشانی (IoU) یک معیار مهم در ارزیابی مدل‌های یادگیری ماشین، به ویژه در مسائل شناسایی اشیاء است. مقدار IoU نشان‌دهنده میزان همپوشانی بین پیش‌بینی‌های مدل و برچسب‌های واقعی (ground truth) است.

در اینجا، مقدار IoU برابر با ۰.۶۰. انتخاب شده است به دلایل زیر:

۱. تعادل بین دقت و فراخوانی: یک IoU پایین‌تر از ۰.۵ ممکن است پیش‌بینی‌های نادرست را به عنوان درست شناسایی کند، در حالی که یک IoU بالاتر از ۰.۷۰ یا ۰.۸۰ ممکن است به طور قابل توجهی تعداد پیش‌بینی‌های مثبت را کاهش دهد. مقدار ۰.۶۰ معمولاً به عنوان یک نقطه تعادل مناسب بین دقت و فراخوانی در بسیاری از مسائل شناسایی اشیاء در نظر گرفته شد.

۲. معیار استاندارد: در بسیاری از مسابقات و تحقیقات، IoU برابر با ۰.۵۰ یا ۰.۶۰ به عنوان معیار استاندارد برای ارزیابی عملکرد مدل‌ها استفاده می‌شود. این مقدار به طور گسترده‌ای پذیرفته شده است و می‌تواند به مقایسه نتایج بین مدل‌های مختلف کمک کند.

۳. سازگاری با داده‌ها: بسته به نوع داده‌ها و پیچیدگی اشیاء مورد شناسایی، ممکن است انتخاب مقدار مناسب IoU متفاوت باشد. برای برخی از مجموعه‌های داده، ۰.۶۰ می‌تواند به عنوان یک آستانه مناسب برای تعیین صحت پیش‌بینی‌ها باشد.)

```
results = model.predict('E:\\2\\*.png', conf=.2)
```

- پیش‌بینی بر روی تصویر: این خط تصاویر مشخص شده (*_color.png) را برای تشخیص دست‌ها پیش‌بینی می‌کند. (ارزیابی نتیجه روی ۱۲۰ مثال تست شامل ۱۵ کلاس با ۸ نمونه) $\text{conf}=0.2$ - سطح اطمینان برای پیش‌بینی‌ها، در اینجا ۲۰٪ است.

انتخاب سطح اطمینان ۲۰٪ (confidence threshold) برای پیش‌بینی‌ها در مدل‌های یادگیری ماشین، به ویژه در شناسایی اشیاء، می‌تواند به دلایل زیر باشد:

۱. افزایش تعداد پیش‌بینی‌ها: با تنظیم سطح اطمینان به ۲۰٪، مدل قادر است تعداد بیشتری از پیش‌بینی‌ها را تولید کند. این امر به ویژه در مواقعی مفید است که هدف شناسایی هر نوع شیء موجود در تصویر باشد، حتی اگر برخی از پیش‌بینی‌ها ممکن است دقیق نباشند.

۲. کاهش احتمال از دست دادن اشیاء: در برخی از سناریوهای، ممکن است اشیاء مهمی در تصویر وجود داشته باشند که مدل به راحتی آن‌ها را شناسایی نمی‌کند. با کاهش آستانه اطمینان، احتمال شناسایی این اشیاء افزایش می‌یابد.

۳. تجزیه و تحلیل اولیه: ممکن است این مقدار به عنوان یک نقطه شروع برای تجزیه و تحلیل اولیه انتخاب شده باشد. پس از بررسی نتایج، می‌توان آستانه اطمینان را تنظیم کرد تا دقت پیش‌بینی‌ها بهبود یابد.

۴. مدل‌های غیر دقیق: اگر مدل در شرایط خاص یا با داده‌های خاصی آموزش دیده باشد و دقت آن پایین باشد، انتخاب یک آستانه اطمینان پایین‌تر می‌تواند به شناسایی بیشتر کمک کند.

۵. تنظیمات خاص پروژه: ممکن است این انتخاب بر اساس نیازهای خاص پروژه یا مجموعه داده‌ای که مورد استفاده قرار می‌گیرد، انجام شود. برای مثال، در کاربردهایی که شناسایی سریع و ابتدایی اشیاء مهم‌تر از دقت بالاست، این نوع تنظیمات معمولاً رایج است.

for result in results:

```
    boxes = result.boxes # Boxes object for bounding box outputs  
    masks = result.masks # Masks object for segmentation masks outputs  
    keypoints = result.keypoints # Keypoints object for pose outputs  
    probs = result.probs # Probs object for classification outputs  
    obb = result.obb # Oriented boxes object for OBB outputs  
    result.show() # display to screen  
    result.save(filename="result.jpg") # save to disk
```

- پردازش نتایج: در اینجا برای هر نتیجه خروجی(عکس‌های تست)، ویژگی‌های مختلف استخراج می‌شوند:

- **boxes**: مختصات جعبه‌های محصور کننده. (مستطیل‌هایی هستند که دور اشیاء شناسایی شده کشیده می‌شوند).

- **masks**: ماسک‌های مربوط به شناسایی. (این ویژگی شامل ماسک‌های تقسیم‌بندی است که برای هر شیء شناسایی شده ایجاد شده‌اند. ماسک‌ها به صورت پیکسل به پیکسل نشان می‌دهند که کدام قسمت از تصویر متعلق به هر شیء است).

- **keypoints**: نقاط اصلی (مثلاً مفاصل). (این ویژگی شامل نقاط کلیدی (keypoints) است که معمولاً برای شناسایی موقعیت اعضای بدن یا اشیاء خاص در تصویر استفاده می‌شود. به عنوان مثال، در شناسایی حالت (pose estimation)، این نقاط ممکن است شامل موقعیت سر، دست‌ها و پاهای باشد).

- **probs**: احتمال دسته‌بندی. (این ویژگی شامل احتمال‌ها (probabilities) است که نشان‌دهنده اطمینان مدل در مورد طبقه‌بندی اشیاء شناسایی شده است. به عبارت دیگر، این

احتمال‌ها نشان می‌دهند که مدل چقدر مطمئن است که یک شیء خاص به یک کلاس خاص تعلق دارد).

- **obb**: جعبه‌های جهت‌دار. این ویژگی شامل جعبه‌های محدود‌کننده با جهت‌گیری (oriented bounding boxes) است. این نوع جعبه‌ها به مدل اجازه می‌دهند تا اشیاء را با دقت بیشتری شناسایی کند، به ویژه زمانی که اشیاء به صورت مایل یا غیرمستقیم قرار دارند.

- سپس تصویر را نمایش می‌دهد و آن را با نام result.jpg ذخیره می‌کند.

```
import cv2
```

- **وارد کردن OpenCV**: برای کار با ویدیو و تصویر، کتابخانه OpenCV وارد می‌شود.

```
model = YOLO("final_model.pt")
```

- **بارگذاری مجدد مدل**: بار دیگر مدل بارگذاری می‌شود که نشان می‌دهد ما به پیش‌بینی بر روی ویدیو ادامه می‌دهیم.

```
video_path = "f.mp4"
```

```
cap = cv2.VideoCapture()
```

- **پیکربندی ویدیو**: در اینجا، مسیر ویدیو مشخص می‌شود و شیء VideoCapture برای دریافت فریم‌های ویدیو ایجاد می‌شود. گزینه ۰ برای استفاده از دوربین وب است.

```
while cap.isOpened():
```

- حلقه ویدیو: این حلقه تا زمانی که ویدیو باز باشد (و فریم‌هایی وجود داشته باشد) اجرا می‌شود.

```
success, frame = cap.read()
```

- خواندن فریم: یک فریم از ویدیو خوانده می‌شود و متغیر `SUCCESS` نشان می‌دهد که آیا خواندن موفق بوده یا خیر.

```
if success:
```

```
    results = model.track(frame)
```

- تشخیص بر روی فریم: اگر فریم به درستی خوانده شده باشد، از مدل برای تعقیب اشیاء (دست‌ها) در فریم استفاده می‌شود.

```
annotated_frame = results[0].plot()
```

- نقاشی کردن نتایج: نتایج تشخیص بر روی فریم نقاشی می‌شود تا اشیاء شناسایی شده قابل مشاهده باشند.

```
cv2.imshow("model Tracking", annotated_frame)
```

- نمایش فریم: فریم حاوی تشخیص را نمایش می‌دهد.

```
if cv2.waitKey(1) & 0xFF == ord("q"):
```

```
    break
```

- پایان حلقه: اگر کاربر کلید "q" را فشار دهد، حلقه شکسته می‌شود و برنامه پایان می‌یابد.

```
cap.release()
```

```
cv2.destroyAllWindows()
```

- آزادسازی منابع: در نهایت، شیء VideoCapture آزاد شده و تمام پنجره‌های باز بسته می‌شوند.

```
C:\> Users > Almandi > Desktop > HandDetection > src > Hand_Detection_Val_Test_Track.ipynb > from ultralytics import YOLO
+ Code + Markdown | ▷ Run All ✎ Clear All Outputs | 📄 Outline ...
```

...

```
0: 480x640 (no detections), 41.8ms
Speed: 4.7ms preprocess, 41.8ms inference, 5.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 One_VFR, 1 Six_VFR, 1 Punch_VFR, 1 Horiz_HFR, 1 Collab, 33.7ms
Speed: 8.7ms preprocess, 33.7ms inference, 9.6ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 Punch_VFR, 31.7ms
Speed: 0.2ms preprocess, 31.7ms inference, 0.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 Horiz_HFR, 1 Collab, 1 XSign, 39.6ms
Speed: 3.6ms preprocess, 39.6ms inference, 0.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 Collab, 33.3ms
Speed: 3.1ms preprocess, 33.3ms inference, 0.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 Collab, 14.3ms
Speed: 0.0ms preprocess, 14.3ms inference, 0.5ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 2 Collabs, 1 XSign, 8.7ms
Speed: 2.7ms preprocess, 8.7ms inference, 0.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 Six_VFR, 1 Horiz_HFR, 1 Collab, 1 XSign, 7.4ms
Speed: 0.0ms preprocess, 7.4ms inference, 4.1ms postprocess per image at shape (1, 3, 480, 640)
...
Speed: 0.0ms preprocess, 29.1ms inference, 5.4ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 Horiz_HFR, 13.2ms
Speed: 0.0ms preprocess, 13.2ms inference, 0.0ms postprocess per image at shape (1, 3, 480, 640)
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

۵-متريکها و ارزیابی مدل:

```
✓ [8] 162 epochs completed in 3.152 hours.  
3h Optimizer stripped from runs/detect/train/weights/last.pt, 6.2MB  
→ Optimizer stripped from runs/detect/train/weights/best.pt, 6.2MB  
  
Validating runs/detect/train/weights/best.pt...  
Ultralytics YOLOv8.2.103 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)  
Model summary (fused): 168 layers, 3,008,573 parameters, 0 gradients, 8.1 GFLOPs  

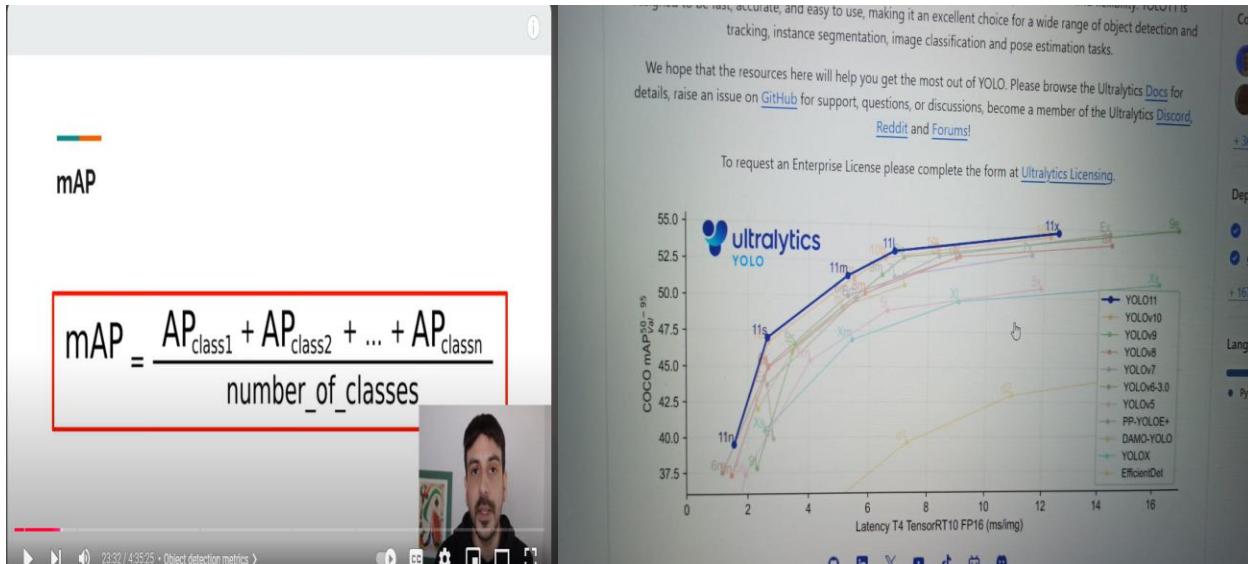

| Class     | Images | Instances | Box(P) | R     | mAP50 | mAP50-95: 100% | 8/8 [00:03<00:00, 2.39it/s] |
|-----------|--------|-----------|--------|-------|-------|----------------|-----------------------------|
| all       | 240    | 430       | 0.837  | 0.873 | 0.93  | 0.525          |                             |
| One_VFR   | 10     | 20        | 0.962  | 1     | 0.995 | 0.585          |                             |
| Two_VFR   | 16     | 32        | 0.845  | 1     | 0.925 | 0.453          |                             |
| Three_VFR | 16     | 32        | 1      | 0.936 | 0.995 | 0.58           |                             |
| Four_VFR  | 16     | 32        | 0.931  | 0.845 | 0.977 | 0.484          |                             |
| Five_VFR  | 32     | 62        | 0.464  | 0.484 | 0.476 | 0.285          |                             |
| Seven_VFR | 16     | 32        | 0.362  | 1     | 0.986 | 0.639          |                             |
| Eight_VFR | 16     | 32        | 0.815  | 0.719 | 0.837 | 0.439          |                             |
| Nine_VFR  | 16     | 32        | 0.93   | 1     | 0.995 | 0.601          |                             |
| Punch_VFR | 16     | 32        | 1      | 0.682 | 0.862 | 0.439          |                             |


| Class     | Images | Instances | Box(P) | R     | mAP50 | mAP50-95: 100% | Speed: 0.3ms preprocess, 2.1ms inference, 0.0ms loss, 3.6ms postprocess per image | Results saved to runs/detect/train |
|-----------|--------|-----------|--------|-------|-------|----------------|-----------------------------------------------------------------------------------|------------------------------------|
| all       | 240    | 430       | 0.837  | 0.873 | 0.93  | 0.525          |                                                                                   |                                    |
| One_VFR   | 10     | 20        | 0.962  | 1     | 0.995 | 0.585          |                                                                                   |                                    |
| Two_VFR   | 16     | 32        | 0.845  | 1     | 0.925 | 0.453          |                                                                                   |                                    |
| Three_VFR | 16     | 32        | 1      | 0.936 | 0.995 | 0.58           |                                                                                   |                                    |
| Four_VFR  | 16     | 32        | 0.931  | 0.845 | 0.977 | 0.484          |                                                                                   |                                    |
| Five_VFR  | 32     | 62        | 0.464  | 0.484 | 0.476 | 0.285          |                                                                                   |                                    |
| Seven_VFR | 16     | 32        | 0.362  | 1     | 0.986 | 0.639          |                                                                                   |                                    |
| Eight_VFR | 16     | 32        | 0.815  | 0.719 | 0.837 | 0.439          |                                                                                   |                                    |
| Nine_VFR  | 16     | 32        | 0.93   | 1     | 0.995 | 0.601          |                                                                                   |                                    |
| Punch_VFR | 16     | 32        | 1      | 0.682 | 0.862 | 0.439          |                                                                                   |                                    |
| Span_VFR  | 16     | 32        | 0.973  | 1     | 0.995 | 0.648          |                                                                                   |                                    |
| Horiz_HFR | 22     | 44        | 0.917  | 1     | 0.995 | 0.428          |                                                                                   |                                    |
| Collab    | 16     | 16        | 0.565  | 1     | 0.995 | 0.547          |                                                                                   |                                    |
| XSign     | 16     | 16        | 1      | 0.56  | 0.995 | 0.627          |                                                                                   |                                    |
| TimeOut   | 16     | 16        | 0.955  | 1     | 0.995 | 0.596          |                                                                                   |                                    |


```

(R)Recall Confidence Curve و mAP ابزارهای مهمی برای ارزیابی و بهبود عملکرد مدل‌های شناسایی اشیاء مانند YOLO هستند و به توسعه‌دهنگان کمک می‌کنند تا کیفیت مدل خود را در تشخیص حالات دست یا هر شیء دیگری افزایش دهند.



Mean Average Precision (mAP)

A very common metric used in object detection tasks is the *mean Average Precision* (mAP). “Mean Average” sounds a bit redundant, doesn’t it? To understand this metric, let’s go back to two classification metrics we discussed in [Chapter 3](#): precision and recall. Remember the tradeoff: the higher the recall, the lower the precision. You can visualize this in a Precision/Recall curve (see [Figure 3-5](#)). To summarize this curve into a single number, we could compute its Area Under the Curve (AUC). But note that the Precision/Recall curve may contain a few sections where precision actually goes up when recall increases, especially at low recall values (you can see this at the top left of [Figure 3-5](#)). This is one of the motivations for the mAP metric.

Suppose the classifier has a 90% precision at 10% recall, but a 96% precision at 20% recall: there’s really no tradeoff here: it simply makes more sense to use the classifier at 20% recall rather than at 10% recall, as you will get both higher recall and higher precision. So instead of looking at the precision *at* 10% recall, we should really be looking at the *maximum* precision that the classifier can offer with *at least* 10% recall. It would be 96%, not 90%. So one way to get a fair idea of the model’s performance is to compute the maximum precision you can get with at least 0% recall, then 10% recall, 20%, and so on up to 100%, and then calculate the mean of these maximum precisions. This is called the *Average Precision* (AP) metric. Now when there are more than 2 classes, we can compute the AP for each class, and then compute the mean AP (mAP). That’s it!

However, in an object detection systems, there is an additional level of complexity: what if the system detected the correct class, but at the wrong location (i.e., the bounding box is completely off)? Surely we should not count this as a positive prediction. So one approach is to define an IOU threshold: for example, we may consider that a prediction is correct only if the IOU is greater than, say, 0.5, and the predicted class is correct. The corresponding mAP is generally noted mAP@0.5 (or mAP@50%, or sometimes just AP₅₀). In some competitions (such as the Pascal VOC challenge), this is what is done. In others (such as the COCO competition), the mAP is computed for different IOU thresholds (0.50, 0.55, 0.60, ..., 0.95), and the final metric is the mean of all these mAPs (noted AP@[.50:.95] or AP@[.50:0.05:.95]). Yes, that’s a mean mean average.

Mean Average Precision (mAP)

یکی از معیارهای کلیدی برای ارزیابی عملکرد مدل‌های **YOLO** یادگیری عمیق در وظایف شناسایی اشیاء است. این معیار به ویژه در مدل‌هایی مانند **Only Look Once** که برای تشخیص اشیاء طراحی شده‌اند، کاربرد دارد.

Mean Average Precision at IoU=0.5 (mAP@0.5)

- mAP@0.5 به دقت مدل در شناسایی اشیاء با استفاده از یک آستانه Intersection over Union (IoU) برابر با ۰.۵ اشاره دارد. **IoU** معیاری است که میزان همپوشانی بین پیش‌بینی‌های مدل و برچسب‌های واقعی را اندازه‌گیری می‌کند. اگر IoU بین پیش‌بینی و برچسب واقعی بیشتر از ۰.۵ باشد، آن پیش‌بینی به عنوان صحیح در نظر گرفته می‌شود.
- این معیار به ما کمک می‌کند تا ببینیم که مدل چقدر خوب می‌تواند حالات دست را شناسایی کند و آیا پیش‌بینی‌های آن با واقعیت تطابق دارند یا خیر که ما به مقدار کلی ۰.۹۳ رسیدیم.

Mean Average Precision from 0.5 to 0.95 (mAP@[0.5::0.95])

- **mAP@[0.5::0.95]** به میانگین دقت مدل در محدوده‌های مختلف IoU (از ۰.۵ تا ۰.۹۵) اشاره دارد. این معیار معمولاً دقیق‌تر از **mAP@0.5** است زیرا شامل ارزیابی‌های مختلفی از دقت در سطوح مختلف همپوشانی می‌شود.
- این معیار به ما اطلاعات بیشتری درباره توانایی مدل در شناسایی حالات دست در شرایط مختلف می‌دهد و نشان می‌دهد که آیا مدل فقط در شرایط آسان (با IoU بالا) خوب عمل می‌کند یا در شرایط سخت‌تر نیز قابل اعتماد است که ما به مقدار کلی ۰.۵۲۵ رسیدیم.

Recall Confidence Curve

یک ابزار بصری برای تجزیه و تحلیل عملکرد یک مدل شناسایی Recall Confidence Curve اشیاء است. این منحنی نشان دهنده رابطه بین Recall و Confidence Threshold است که ما به مقدار کلی ۰.۸۷۳ رسیدیم.

Recall

• Recall (یا حساسیت) معیاری است که نشان می‌دهد چه درصدی از نمونه‌های مثبت واقعی (حالات دست) توسط مدل شناسایی شده‌اند. به عبارت دیگر، Recall نشان دهنده توانایی مدل در شناسایی تمام موارد مثبت است.

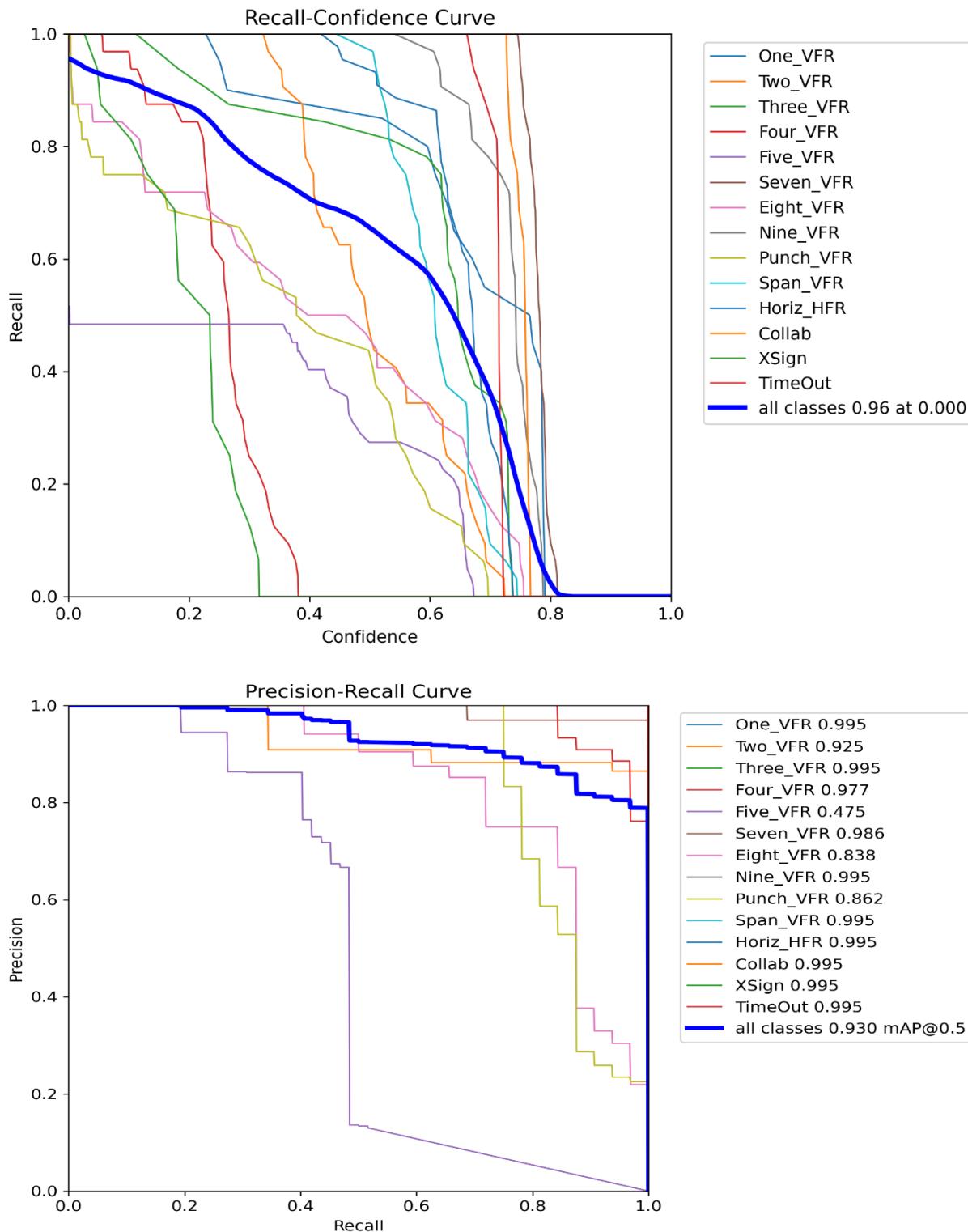
Confidence Threshold

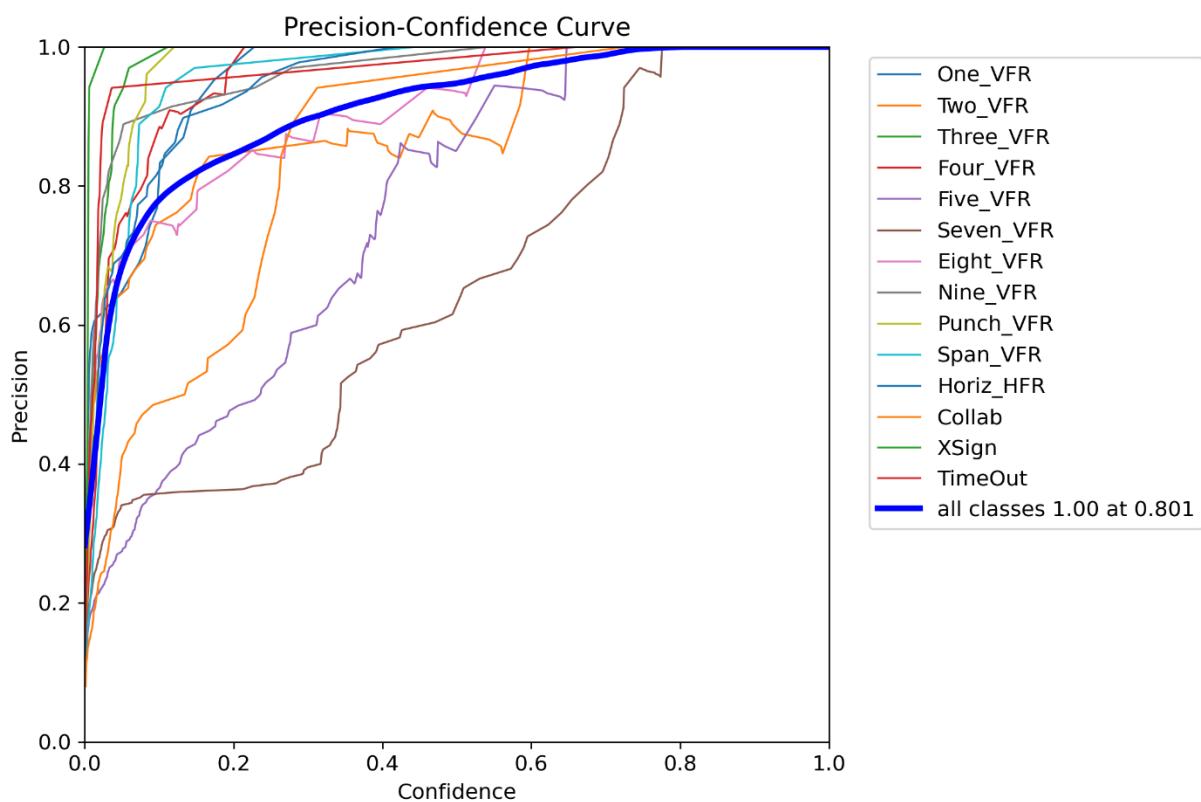
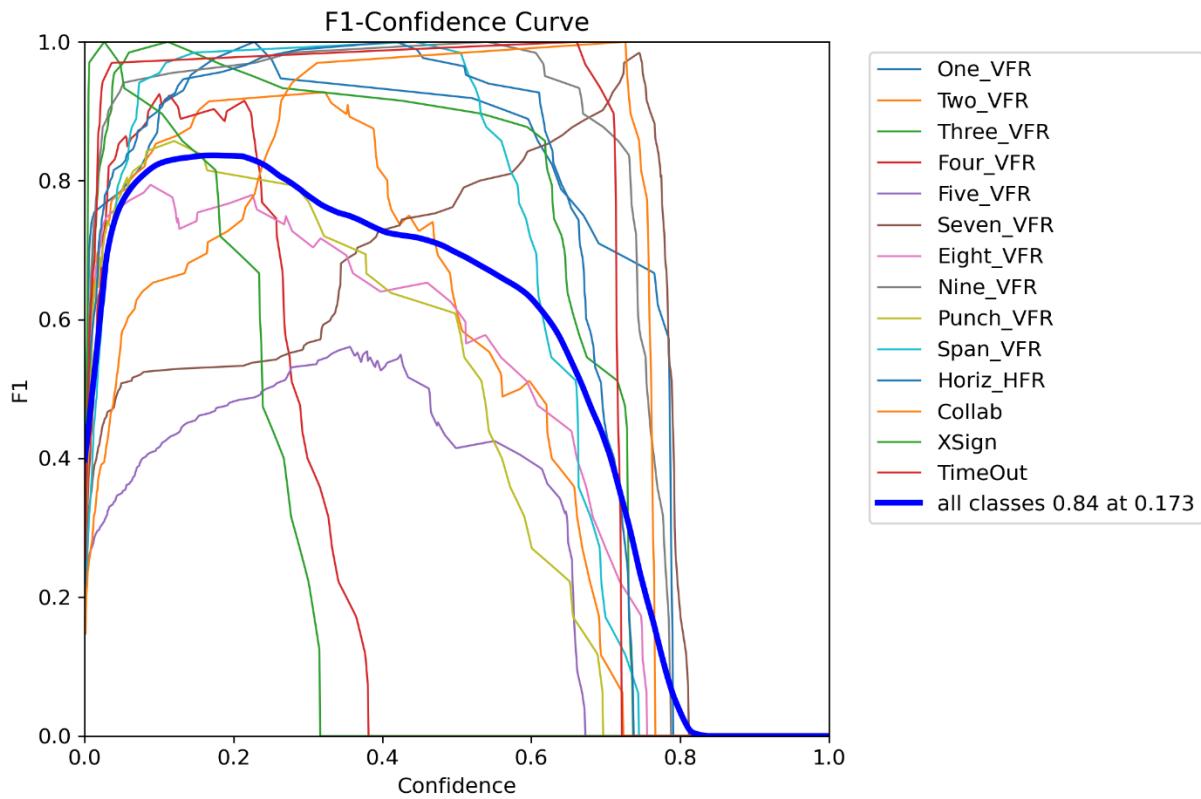
• Confidence Threshold یک مقدار آستانه‌ای است که تعیین می‌کند آیا یک پیش‌بینی باید به عنوان مثبت در نظر گرفته شود یا خیر. به عنوان مثال، اگر یک پیش‌بینی دارای نمره اطمینان بالاتر از ۰.۵ باشد، به عنوان یک شناسایی صحیح در نظر گرفته می‌شود.

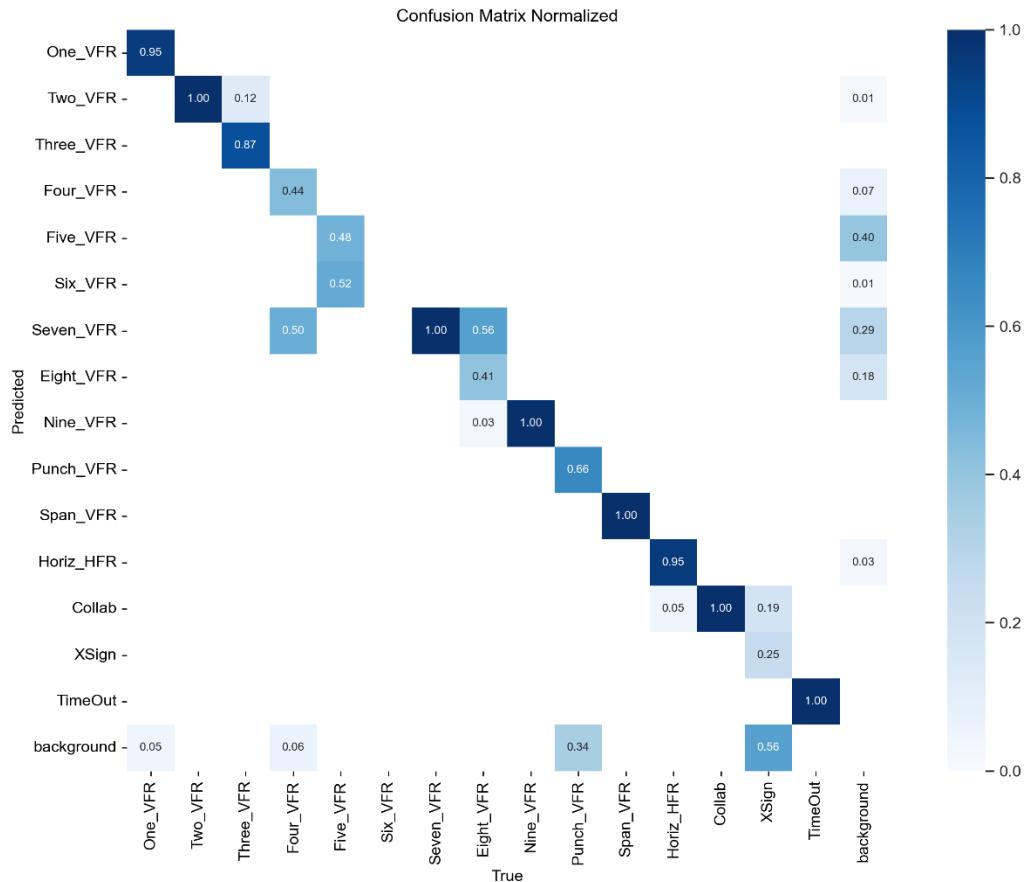
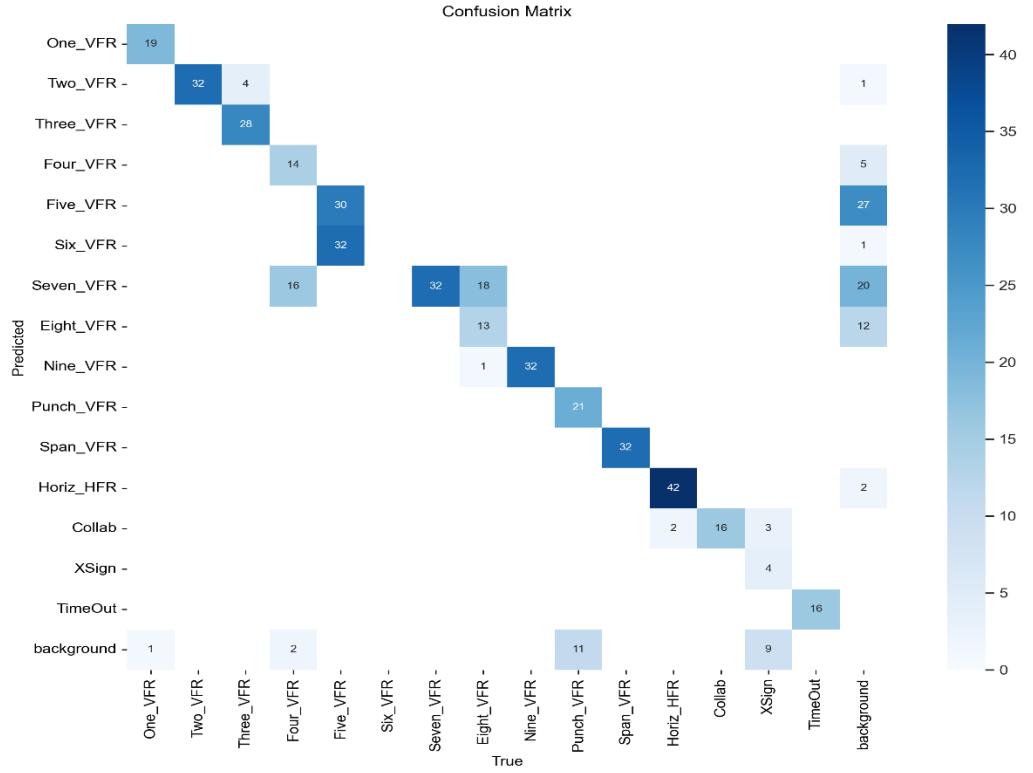
كاربرد Recall Confidence Curve

• با تغییر آستانه اطمینان، می‌توانیم Recall را محاسبه کنیم و منحنی Recall Confidence Curve را ترسیم کنیم. این منحنی به ما نشان می‌دهد که با افزایش آستانه اطمینان، چگونه تغییر می‌کند.

- این ابزار به توسعه‌دهندگان کمک می‌کند تا تصمیم بگیرند که کدام آستانه اطمینان برای اهداف خاص خود مناسب‌تر است و چگونه می‌توانند تعادل بین دقت و Recall را مدیریت کنند.



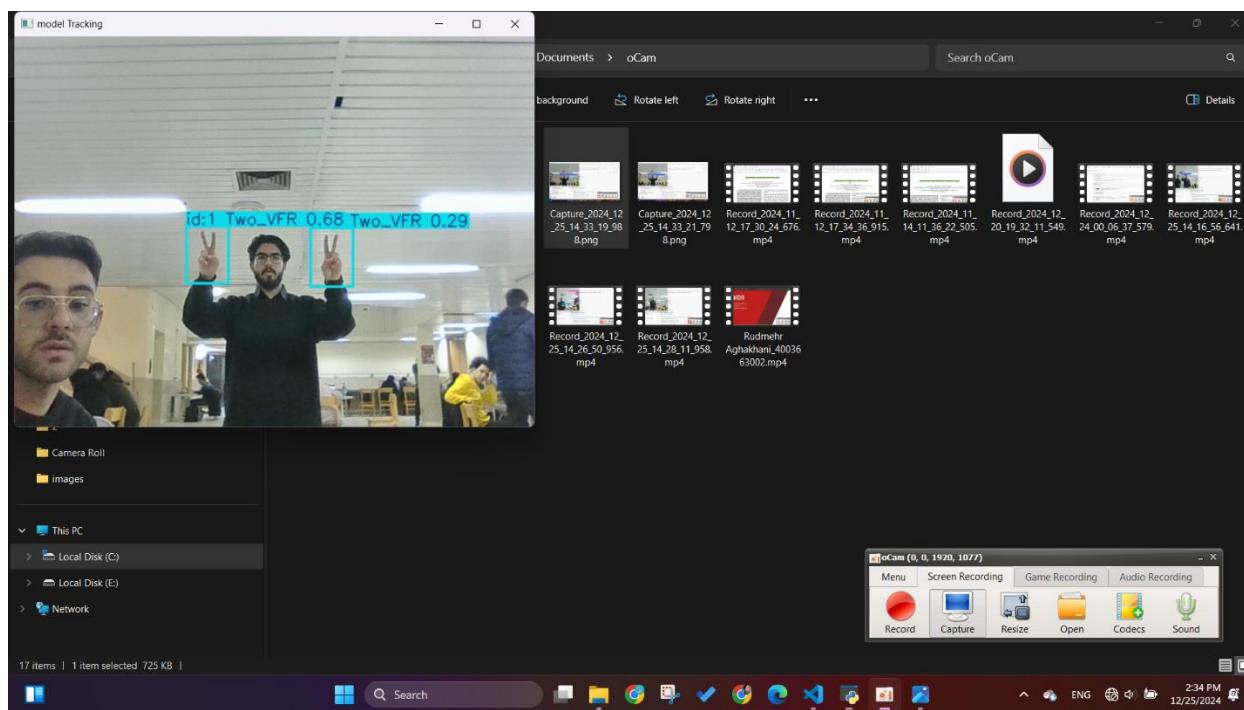
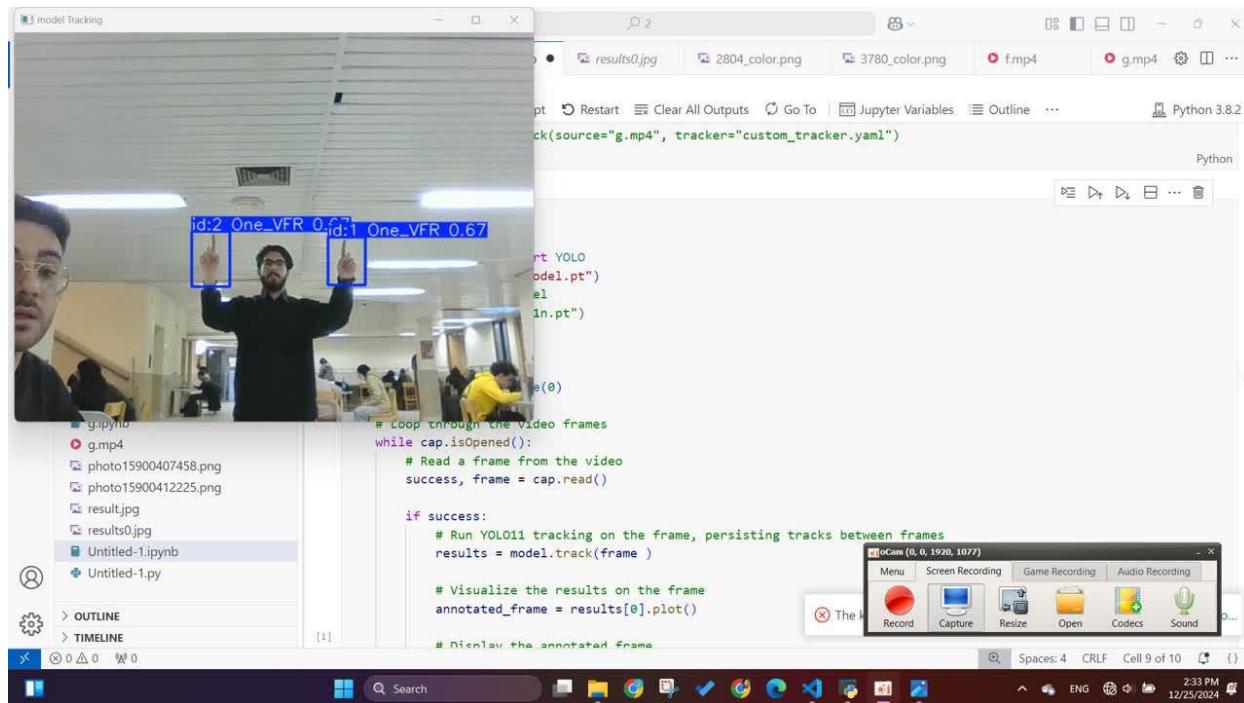


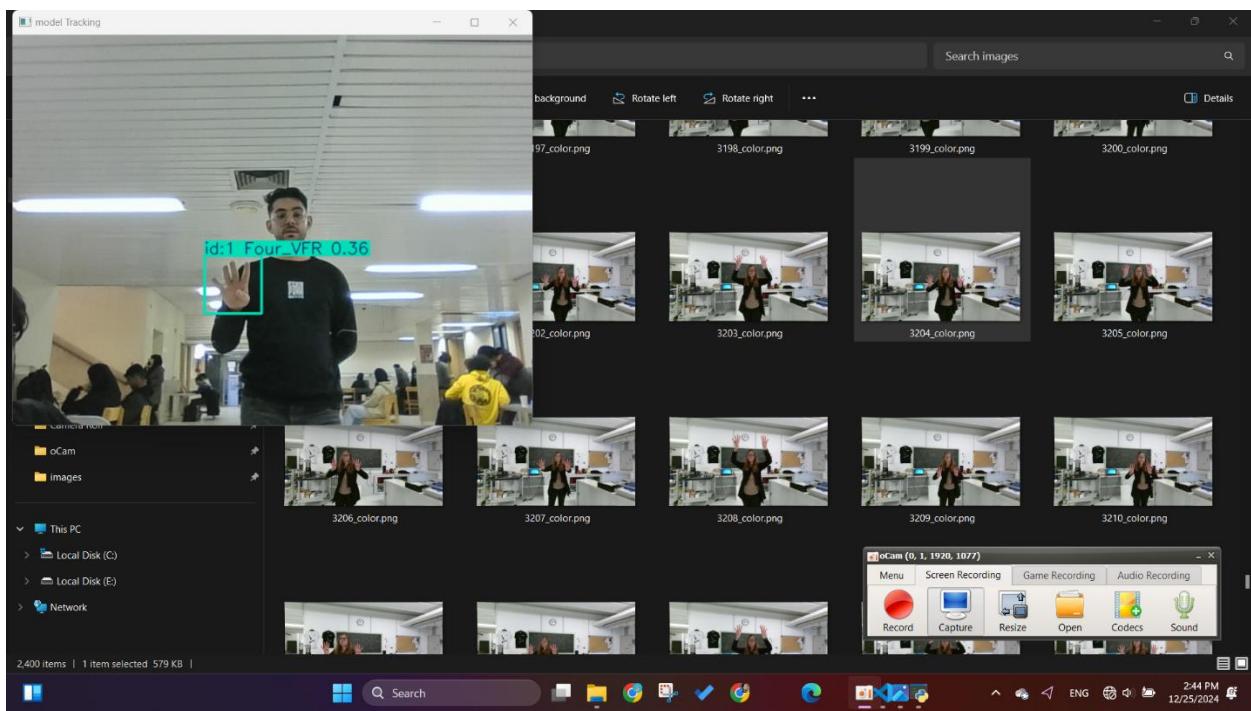
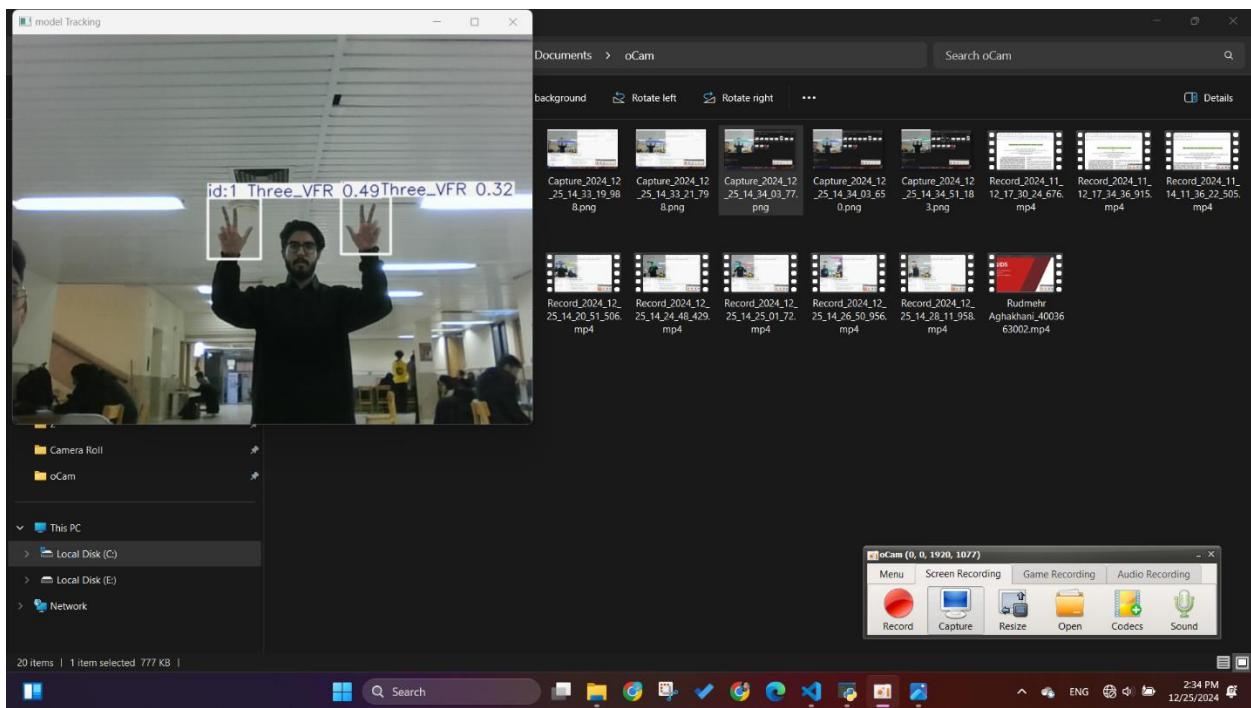


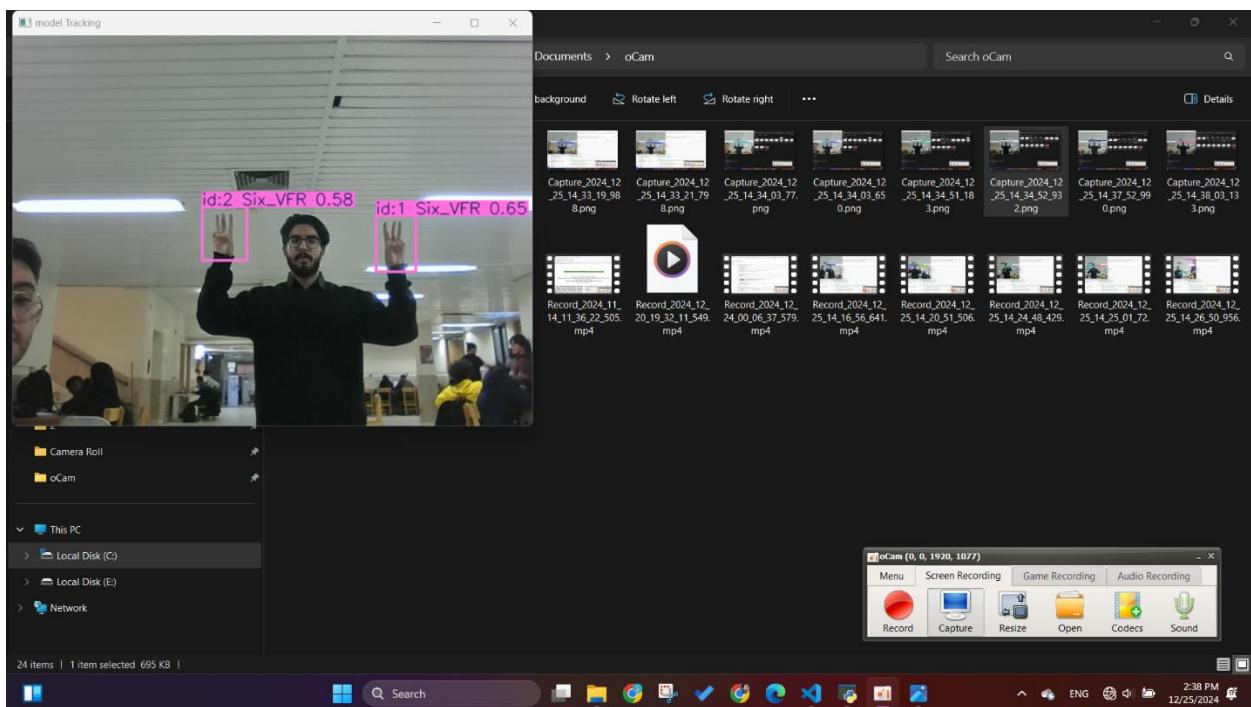
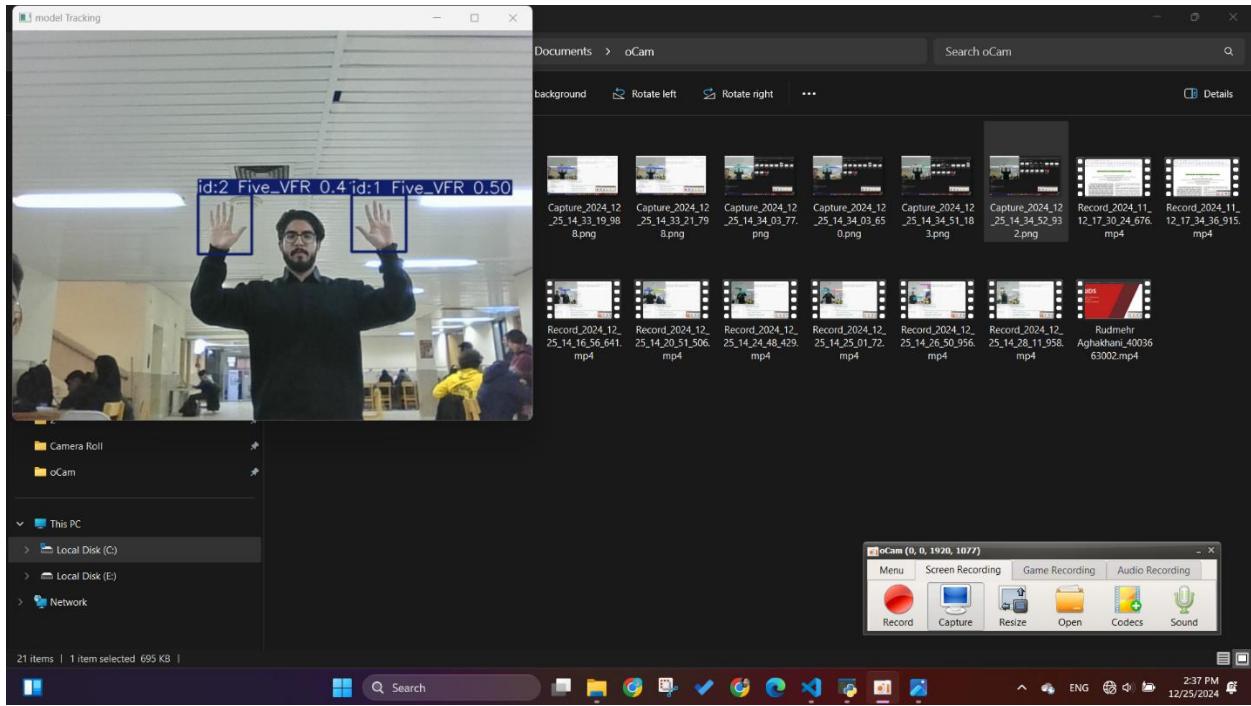
۷-نتایج:

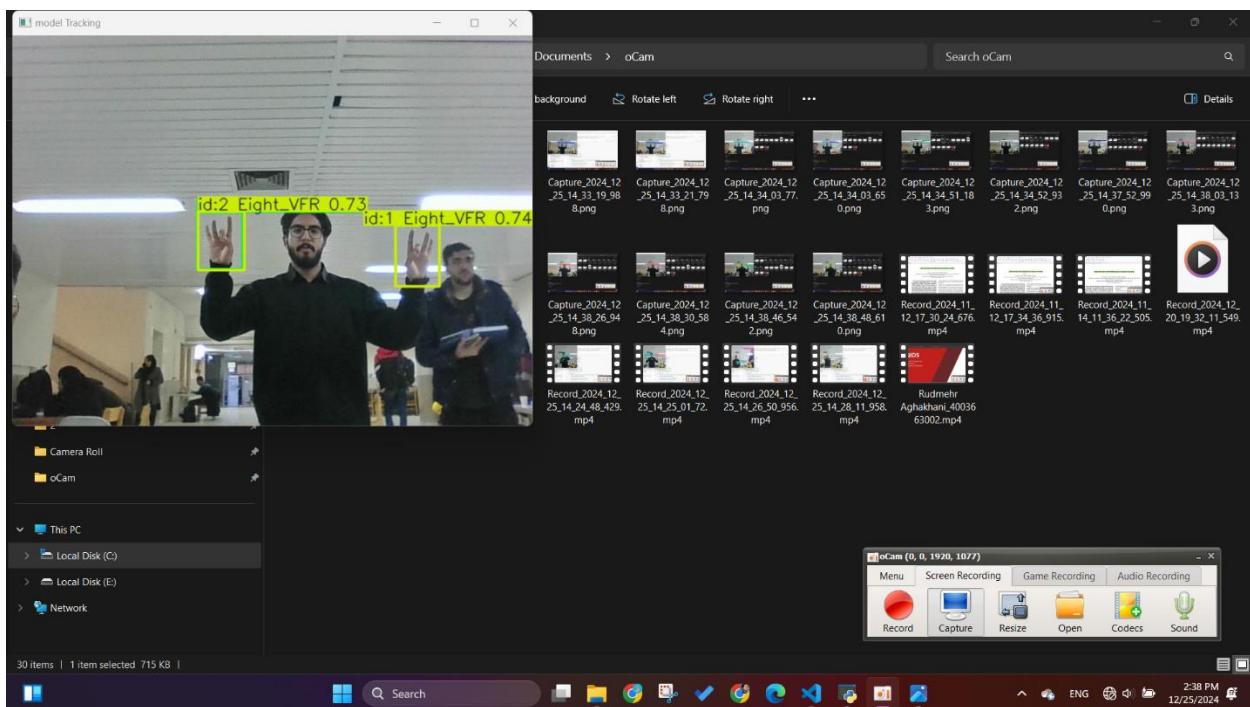
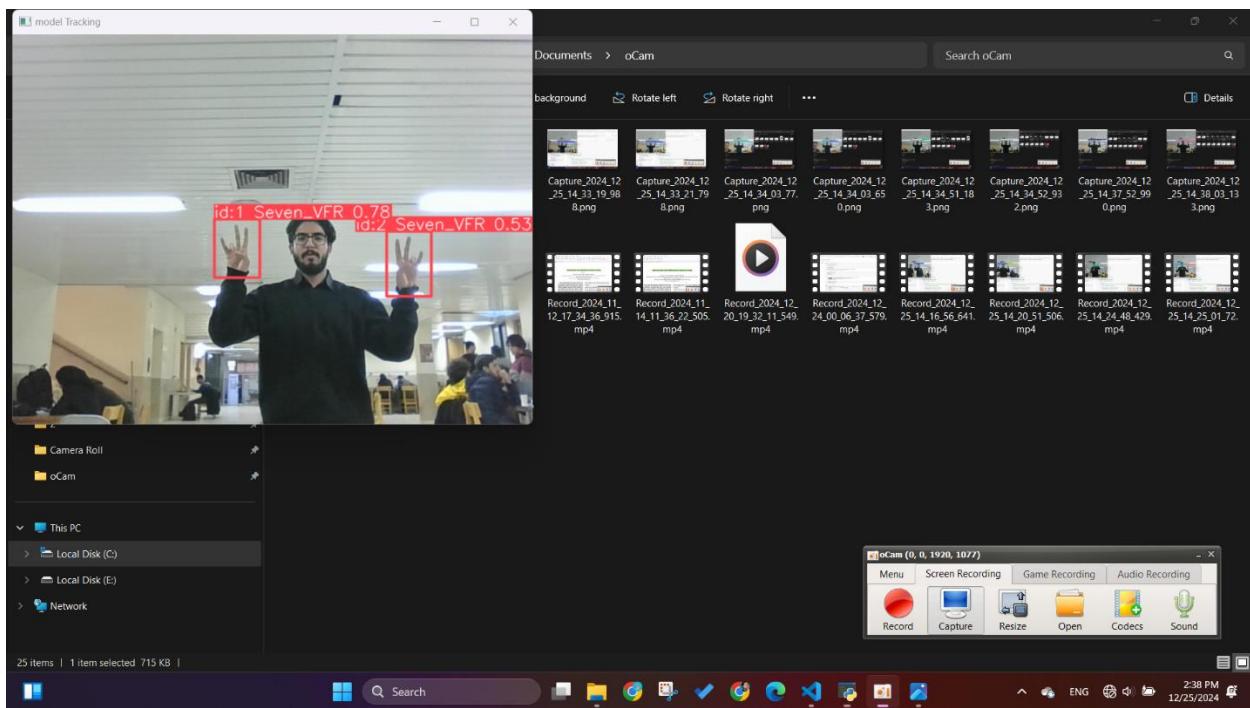
در نهایت پس از آموزش و ساخت مدل یولو ۸ نانو با mAP50 بسیار خوب(٪۹۳) و R بسیار خوب(٪۸۷) و mAP50-95 بسیار خوب(٪۵۲) به کمک وبکم آن را مورد آزمون نهایی قرار داده

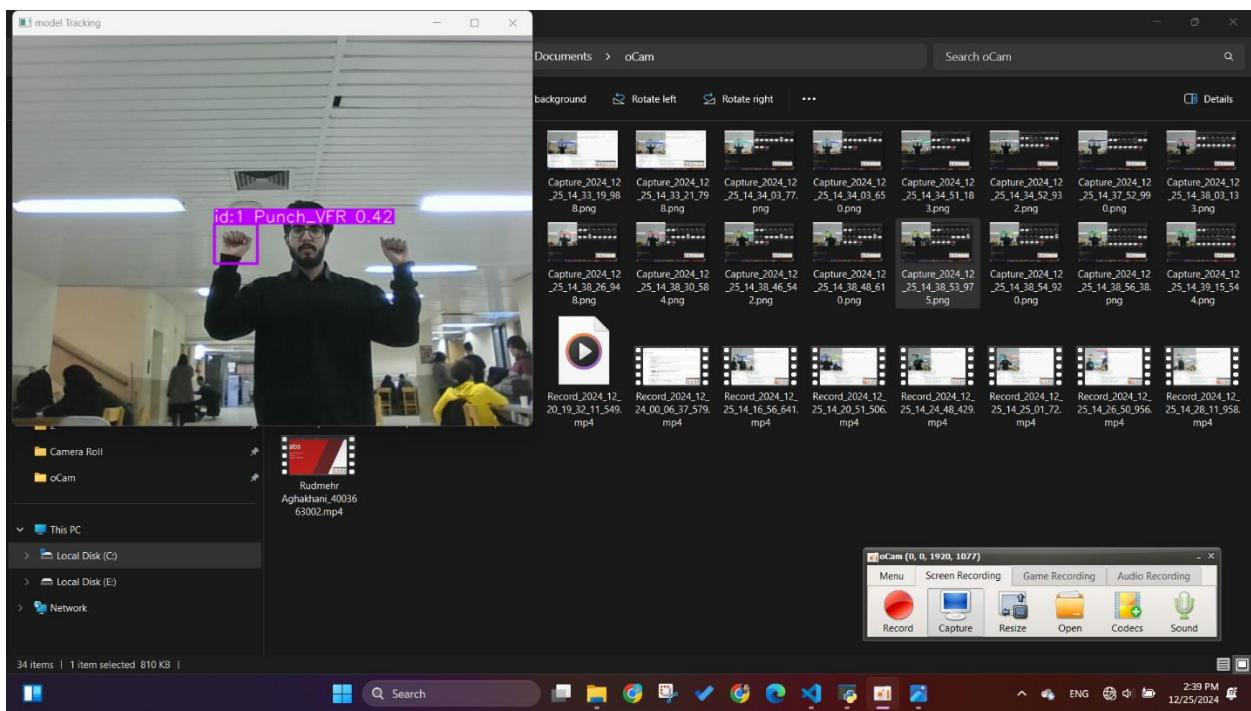
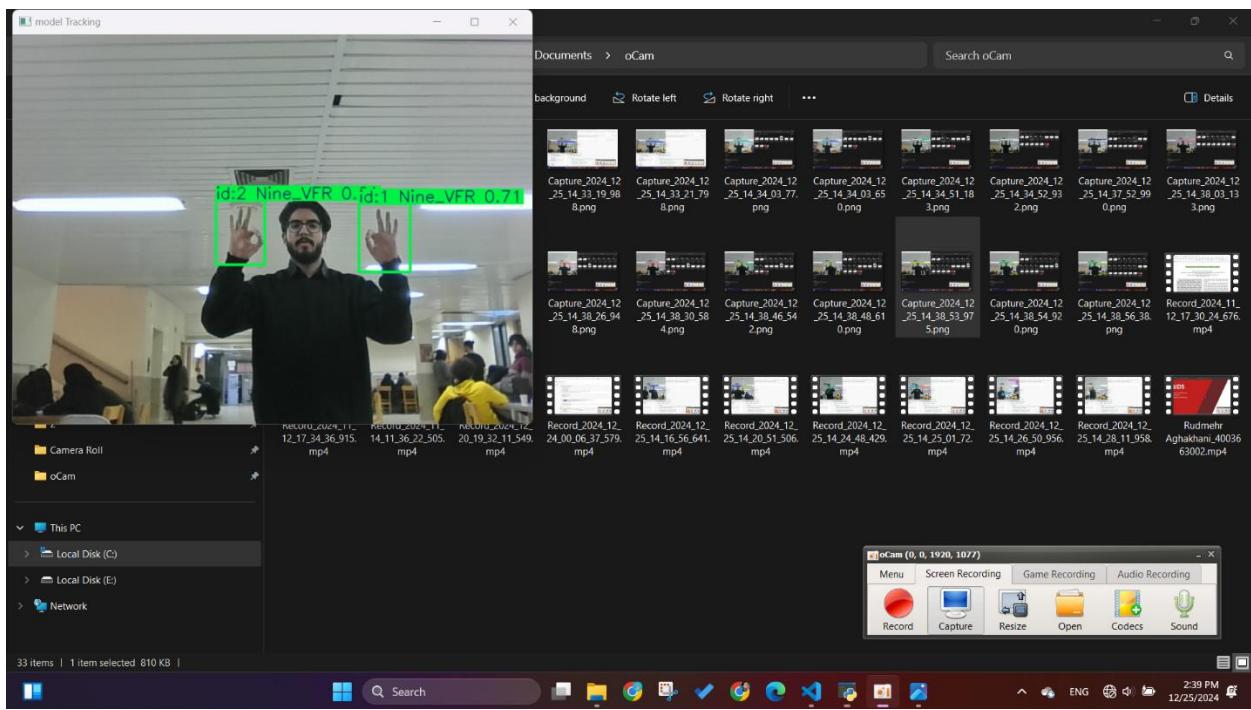
تا صحت عملکرد آن به صورت لایو مورد ارزیابی قرار گیرد:

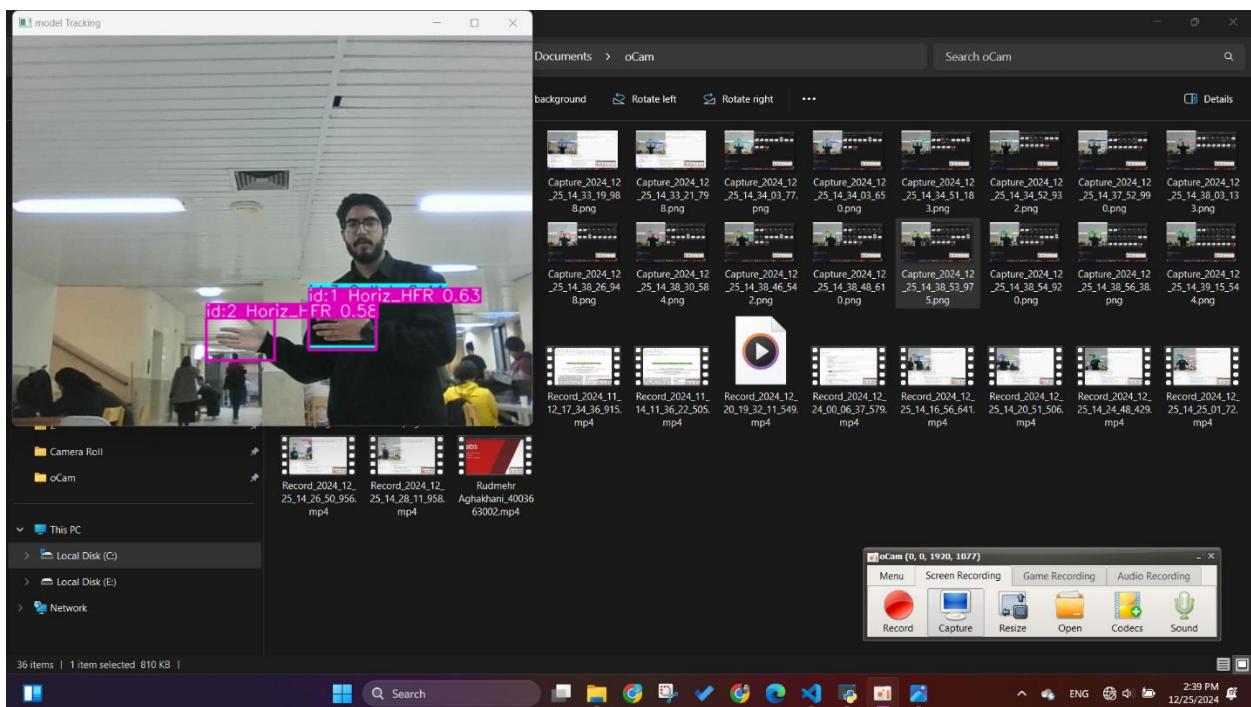
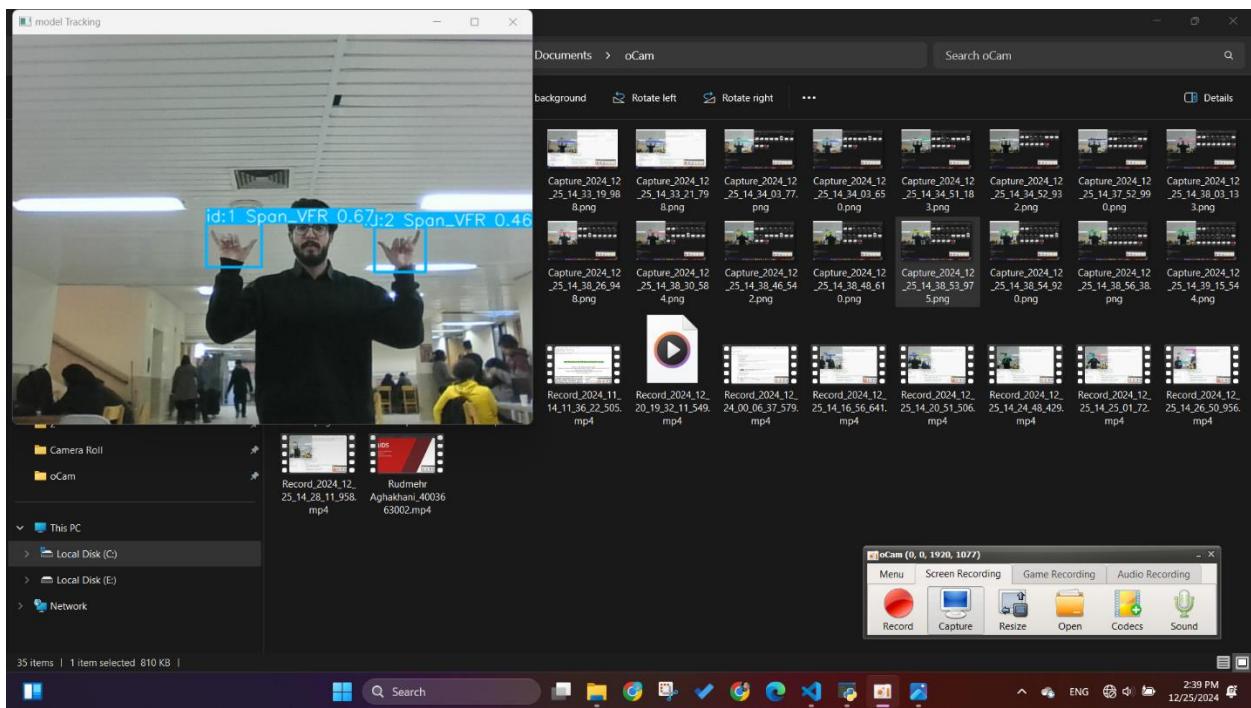


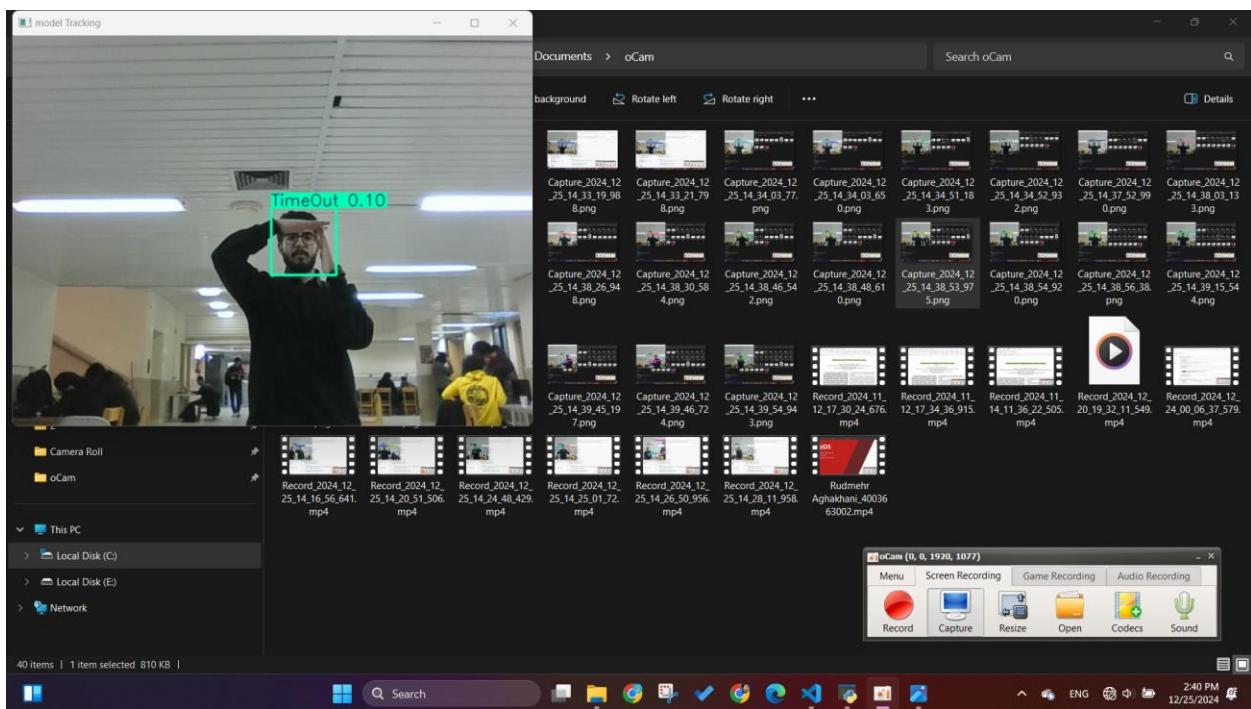
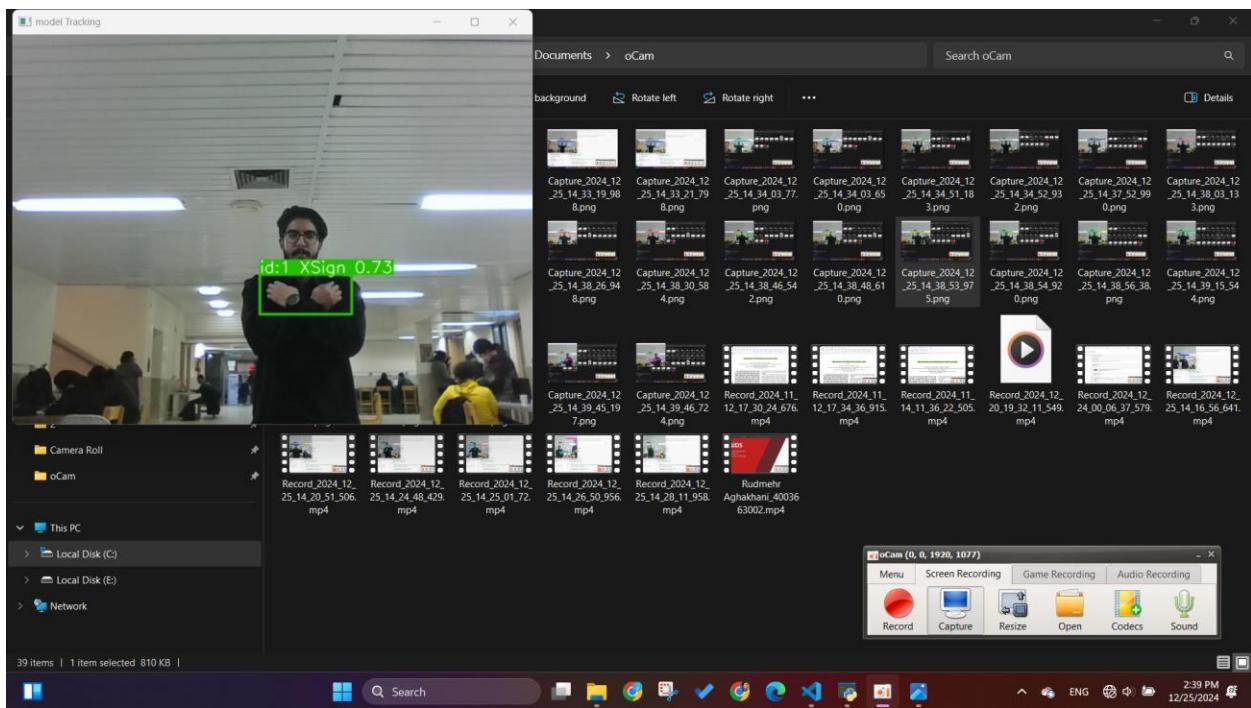


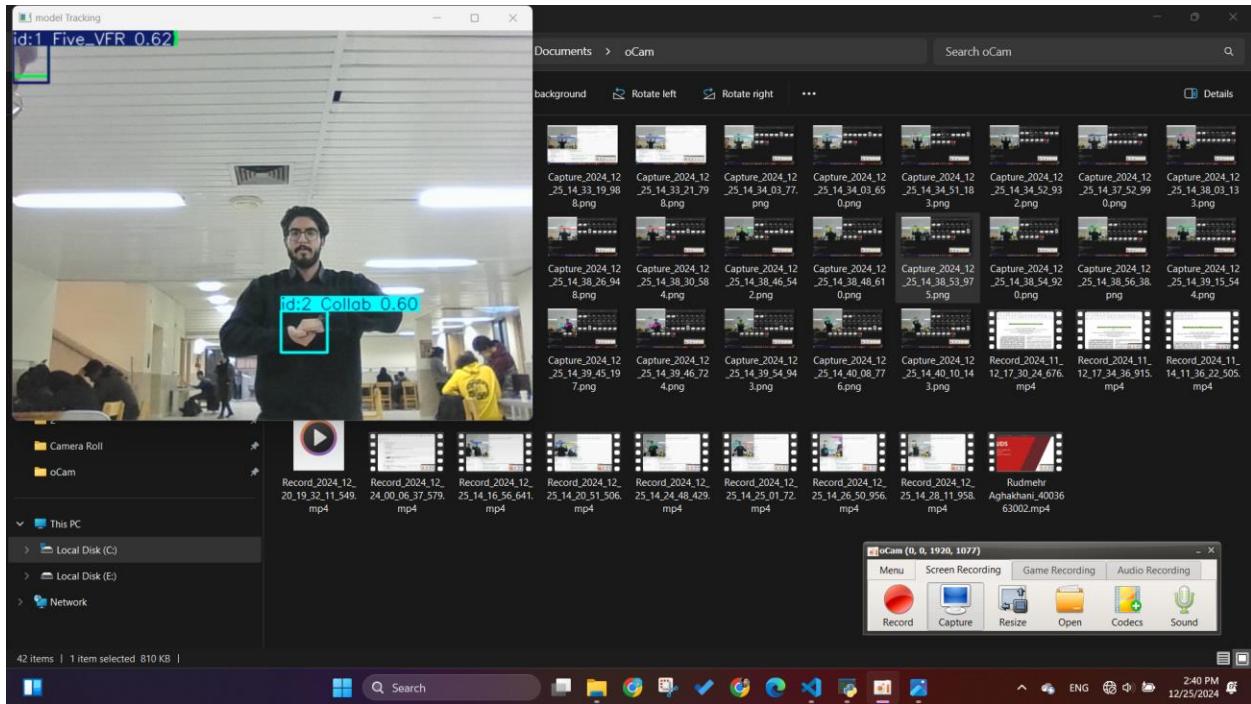












M-Amin-Kiani/HandGestureRecognition

<https://github.com/M-Amin-Kiani/HandGestureRecognition>

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

HandGestureRecognition Public

main 1 Branch 0 Tags Go to file Add file Code

About

detect the gesture of each hand with YOLO8n

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

README

HandGestureRecognition

detect the gesture of each hand with YOLO8n

(a) One_VFR	(b) Two_VFR	(c) Three_VFR	(d) Four_VFR	(e) Five_VFR
(f) Six_VFR	(g) Seven_VFR	(h) Eight_VFR	(i) Nine_VFR	(j) Punch_VFR
(k) Span_VFR	(l) Horiz_HFR	(m) Collab	(n) XSign	(o) TimeOut

Windows Type here to search 8°C Cloudy 7:17 PM 12/27/2024

٨- مراجع

- [1] <https://data.mendeley.com/datasets/ndrczc35bt/1>
- [2] <https://app.cvat.ai>
- [3] <https://www.youtube.com/watch?v=UL2cfTTqdNo&t=6352s>
- [4] <https://docs.ultralytics.com/modes/predict/>
- [5] <https://colab.research.google.com/>
- [6] https://drive.google.com/drive/folders/1kHrJFmKgruR6TQAFIT9oex11B6n_oE_G
- [7] <https://github.com/M-Amin-Kiani/HandGestureRecognition>
- [8] <https://github.com/PacktPublishing/Modern-Computer-Vision-with-PyTorch>
- [9] <https://www.rasa-ai.com/wp-content/uploads/2022/02/Aur%C3%A9lien-G%C3%A9ron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow -Concepts-Tools-and-Techniques-to-Build-Intelligent-Systems-O% E2%80%99Reilly-Media-2019.pdf>
- [10] <https://drive.google.com/drive/folders/1DJRe4IcbgJXQxKrPLdHtDQjM20vzElnj>
- [11] <https://dergipark.org.tr/en/download/article-file/2484463>