



High Impact Skills Development Program in Artificial Intelligence, Data Science, and Blockchain

Project Title: Object detection using YOLO modal

Module 09: Object detection

Name : MUHAMMAD ASLAM

Roll-no : SK23008

Training center: Skardu

Section : 02

email : maslambaig009@gmail.com

github link : <https://github.com/Aslam009/object-detection-module-project-YOLO-modal>

Project Title: Object detection using YOLO modal

Abstract

Object detection is one of the predominant and challenging problems in computer vision. Over the decade, with the expeditious evolution of deep learning, researchers have extensively experimented and contributed in the performance enhancement of object detection and related tasks such as object classification, localization, and segmentation using underlying deep models. Broadly, object detectors are classified into two categories viz. two stage and single stage object detectors. Two stage detectors mainly focus on selective region proposals strategy via complex architecture; however, single stage detectors focus on all the spatial region proposals for the possible detection of objects via relatively simpler architecture in one shot. Performance of any object detector is evaluated through detection accuracy and inference time. Generally, the detection accuracy of two stage detectors outperforms single stage object detectors. However, the inference time of single stage detectors is better compared to its counterparts. Moreover, with the advent of YOLO (You Only Look Once) and its architectural successors, the detection accuracy is improving significantly and sometime it is better

than two stage detectors. YOLOs are adopted in various applications majorly due to their faster inferences rather than considering detection accuracy. As an example, detection accuracies are 63.4 and 70 for YOLO and Fast-RCNN respectively, however, inference time is around 300 times faster in case of YOLO. In this paper, we present a comprehensive review of single stage object detectors specially YOLOs, regression formulation, their architecture advancements, and performance statistics. Moreover, we summarize the comparative illustration between two stage and single stage object detectors, among different versions of YOLOs, applications based on two stage detectors, and different versions of YOLOs along with the future research direction

Introduction

Object detection is an important field in the domain of computer vision. Various machine learning (ML) and deep learning (DL) models are employed for the performance enhancement in the process of object detection and related tasks. In the earlier time, two stage object detectors were quite popular and effective. With the recent development in single stage object detection and underlying algorithms, they have become significantly better in comparison with most of the two stage object detectors. Moreover, with the advent of YOLOs, various applications have utilized YOLOs for object detection and recognition in various context and performed tremendously well in comparison with their counterparts two stage detectors. This motivates us to write a specific review on YOLO and their architectural successors by presenting their design details, optimizations proposed in the successors, tough competition to two stage object detectors, etc. This section presents the brief introduction of deep learning and computer vision, object detection and related terminologies, challenges, stages and their role in the implementation of any object detection algorithm, brief evolution of various object detection algorithms, popular datasets utilized, and the major contributions of the review.

Object classification and localization

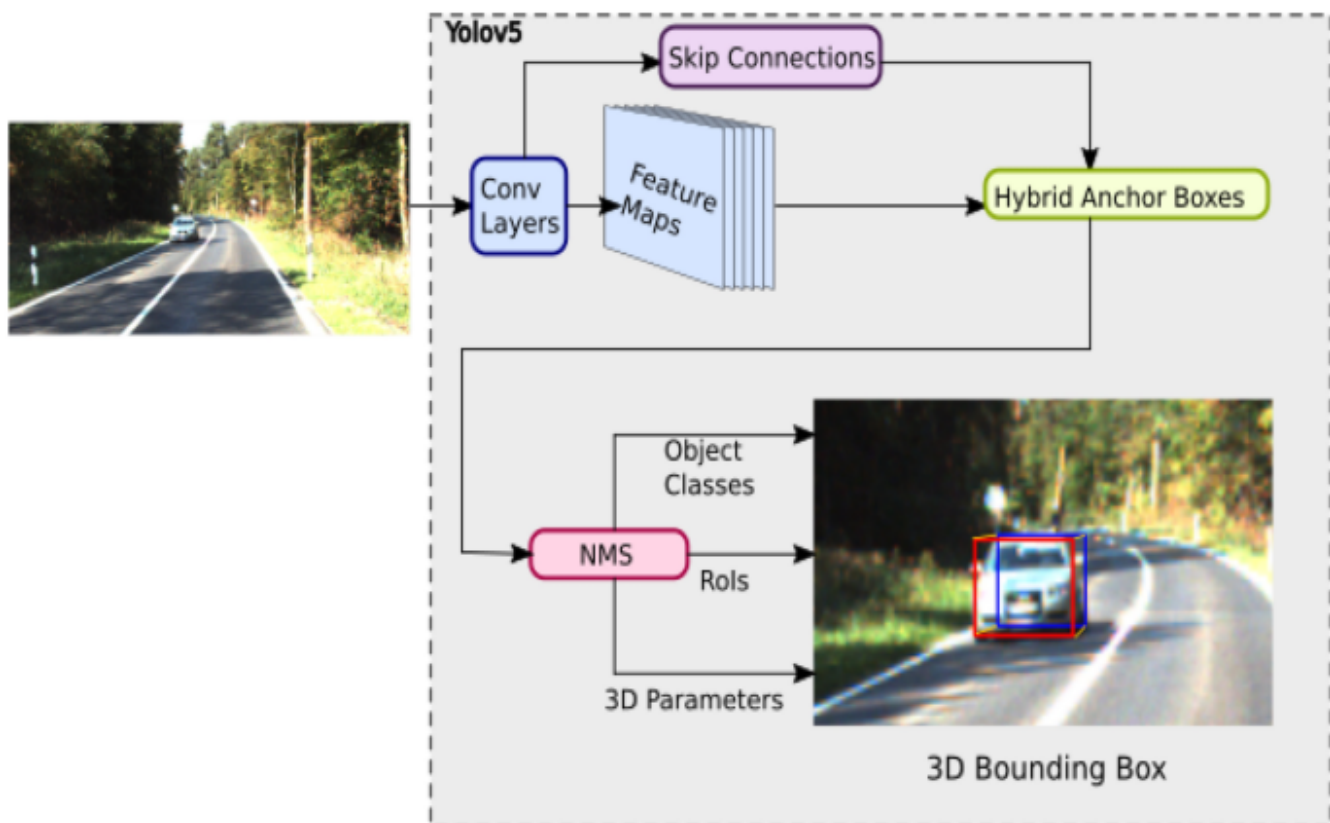
Image Classification is a task of classifying an image or an object in an image into one of the predefined categories. This problem is generally solved with the help of supervised machine learning or deep learning algorithms wherein the model is trained on a large labelled dataset. Some of the commonly used machine learning models for this task includes ANN, SVM, Decision trees, and KNN [66]. However, on the deep learning side, CNNs and its architectural successors and variants dominate other deep models for classifying images and related works. Apart from well-defined machine learning and deep learning models, one can also

witness the usage of other approaches such as Fuzzy logic and Genetic algorithms for the aforementioned tasks [19].

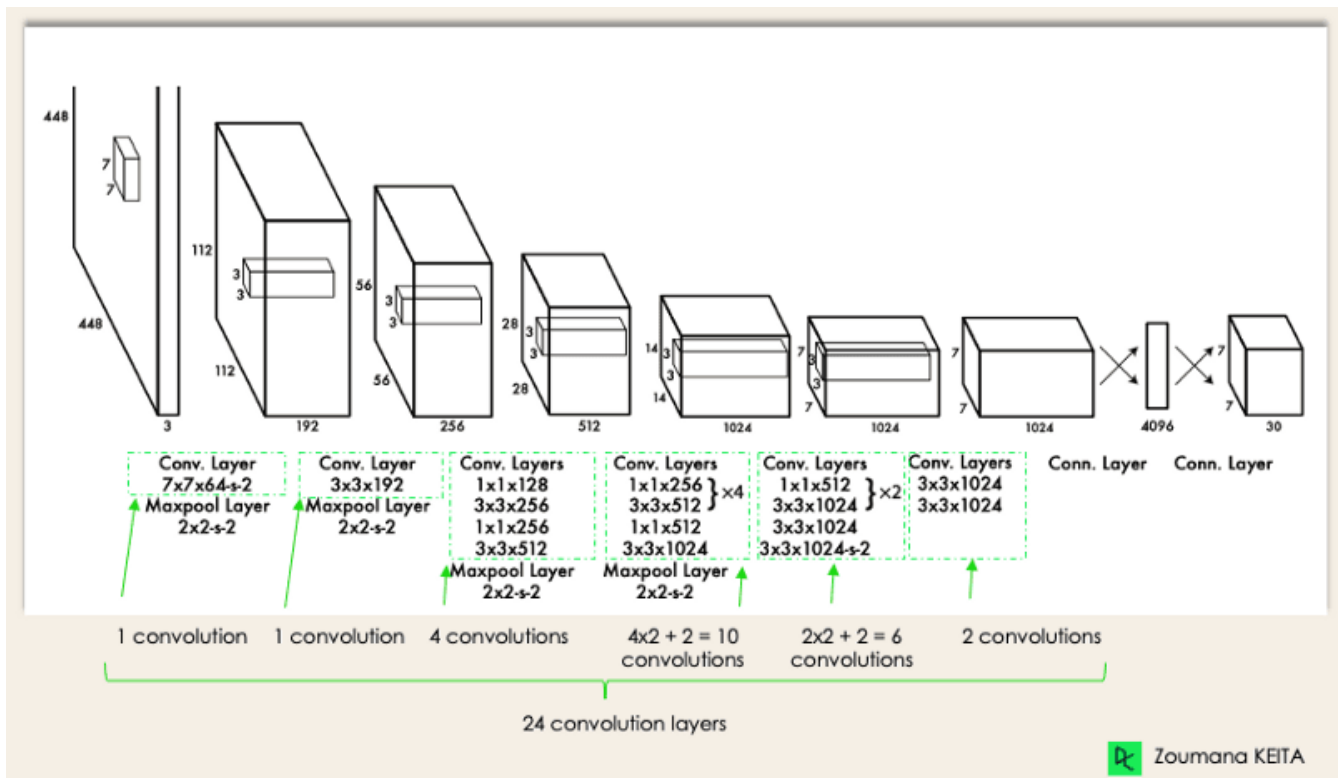
Object Localization is the task of determining position of an object or multiple objects in an image/frame with the help of a rectangular box around an object, commonly known as a bounding box. However, Image segmentation is the process of partitioning an image into multiple segments wherein a segment may contain a complete object or a part of an object. Image segmentation is commonly utilized to locate objects, lines, and curves viz. boundaries of an object or segment in an image. Generally, pixels in a segment possess a set of common characteristics such as intensity, texture, etc. The main motive behind image segmentation is to present the image into a meaningful representation. Moreover, Object detection can be considered as a combination of classification, localization, and segmentation. It is the task of correctly classifying and efficiently localizing single or multiple objects in an image, generally with the help of supervised algorithms given a sufficiently large labelled training set. Figure 1 presents the clear understanding of classification, localization, and segmentation for single and multiple objects in an image in the context of object detection.

Our image.....in progress

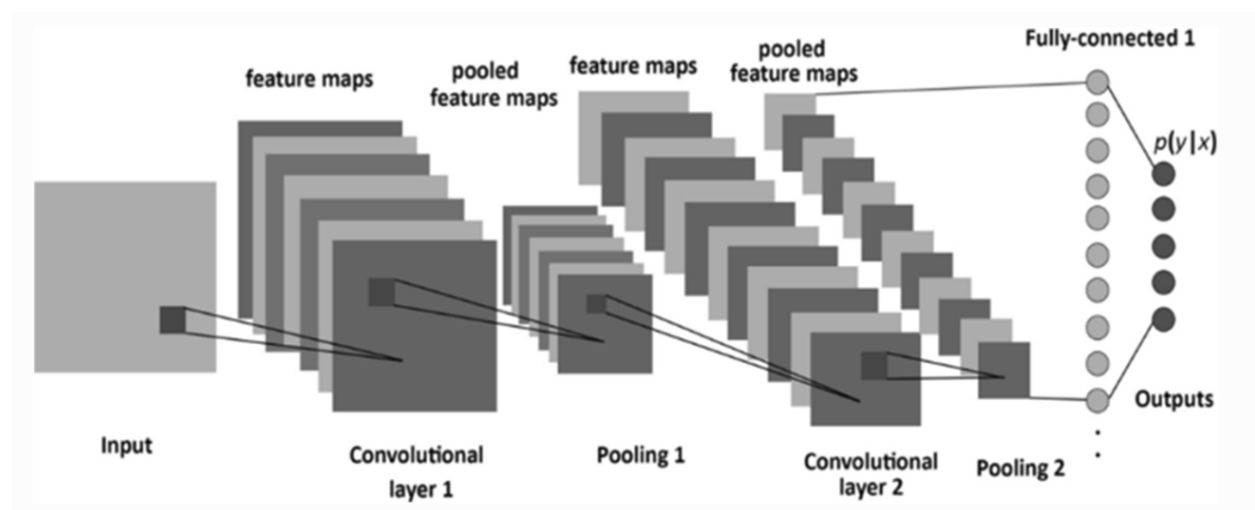
Stages in object detection



Object detection YOLO Modal Diagram



Object detection Architecture diagram



annotation techniques

Object detection using YOLO (You Only Look Once) requires annotated data, where each object in the image is labeled with a bounding box and assigned a class label. Here are some annotation techniques commonly used for YOLO object detection.

Bounding Box Annotation:

Bounding box annotation is the most common technique for object detection. Draw rectangles (bounding boxes) around each object of interest in the image.

Include the coordinates of the bounding box (usually in terms of (x, y, width, height)) and the class label for each object.

Tools like LabelImg, RectLabel, VGG Image Annotator (VIA), and Supervisely can assist in creating bounding box annotations.

Point Annotation:

In point annotation, you annotate a single point within the object's region.

You provide the (x, y) coordinates of a key point, often the object's center.

YOLO can still detect objects using point annotations, but it may be less accurate than bounding boxes.

Mask Annotation (Segmentation):

Mask annotation involves pixel-level annotation of object boundaries.

For each object, you create a binary mask that highlights the object's pixels (foreground) while masking out the background.

Segment the object using image editing software or tools like Labelbox and VGG Image Annotator.

Polygon Annotation:

Polygon annotation provides more precise object boundaries compared to bounding boxes.

You outline the object's shape using a series of connected vertices.

Each polygon annotation represents a single object and can include class labels.

Cuboid Annotation (3D Object Detection):

For 3D object detection, such as detecting objects in LiDAR data or autonomous driving scenarios, you may need cuboid annotations.

Cuboid annotations define the 3D bounding box of an object.

They typically include information about the object's position, dimensions, orientation, and class label.

Keypoint Annotation:

Keypoint annotation involves labeling specific keypoints or landmarks on an object.

Useful for scenarios like human pose estimation, facial landmarks detection, or labeling key features of objects.

Each keypoint has a class label and (x, y) coordinates.

Combination of Techniques:

In some cases, a combination of annotation techniques may be used.

For instance, you might use bounding boxes to annotate objects and keypoint annotations to specify certain features within the objects.

Multi-Class Annotation:

If your dataset contains multiple object classes, you'll need to annotate objects with their corresponding class labels.

Class labels are essential for YOLO to recognize and categorize different object types.

Verification and Quality Control:

After annotation, it's crucial to perform quality control to ensure accurate annotations.

Verify that bounding boxes, masks, or polygons align correctly with the objects in the image.

Check for any missing or mislabeled objects.

Data Augmentation:

To increase the diversity of your dataset and improve model generalization, consider applying data augmentation techniques to your annotated data.

This includes random rotation, scaling, flipping, and changes in brightness and contrast.

Data Annotation Tools:

Use annotation tools or software specifically designed for object detection tasks to streamline the annotation process and maintain consistency.

challenges faced during the object detection using YOLU modal

we have faced many challenges during skardu gitgit road turn left,right and landsliding detection using YOLU modal.the first challenges faces us is data collection and data preproceesing and second is Objects can vary in size, shape, pose, lighting conditions, and occlusion levels, making it difficult for the model to generalize and third is Detecting small objects in images can be challenging, as they may not have enough features to distinguish them from the background and last is challenge faced us is

Variability in environmental conditions such as weather, lighting, and background clutter can affect object detection accuracy.

code snippets

Download Go... Gmail My Account Sign in - Googl... Higher Educat... YouTube Android Hub 4... OnlineShopin... localhost Gmail YouTube Map

object_detection_project.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 9:29PM

Comment

+ Code + Text

```
from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

Ultralytics YOLOv8.0.20 Python-3.10.12 torch-2.0.1+cu118 CPU
Setup complete (2 CPUs, 12.7 GB RAM, 26.3/107.7 GB disk)

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

Mounted at /content/drive

```
[ ] %cd /content/drive/MyDrive/object/objectDetection

/content/drive/MyDrive/object/objectDetection
```

Download Go... Gmail My Account Sign in - Googl... Higher Educat... YouTube Android Hub 4... OnlineShopin... localhost Gmail YouTube Maps a

object_detection_project.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 9:29PM

Comment Share Settings M

+ Code + Text

Connect ^

```
%cd /content/drive/MyDrive/object/objectDetection

!yolo task=detect mode=train model=yolov8s.pt data= data.yaml epochs=10 imgsz=224 plots=True
```

/content/drive/MyDrive/object/objectDetection
Ultralytics YOLOv8.0.20 Python-3.10.12 torch-2.0.1+cu118 CPU
yolo/engine/trainer: task=detect, mode=train, model=yolov8s.yaml, data=data.yaml, epochs=10, patience=50, batch=16, imgsz=224, save=True, cache=False, device=, workers=8, pr
2023-10-01 07:45:21.759127: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-10-01 07:45:23.038793: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Overriding model.yaml nc=80 with nc=4

	from	n	params	module	arguments
0	-1	1	928	ultralytics.nn.modules.Conv	[3, 32, 3, 2]
1	-1	1	18560	ultralytics.nn.modules.Conv	[32, 64, 3, 2]
2	-1	1	29056	ultralytics.nn.modules.C2f	[64, 64, 1, True]
3	-1	1	73984	ultralytics.nn.modules.Conv	[64, 128, 3, 2]
4	-1	2	197632	ultralytics.nn.modules.C2f	[128, 128, 2, True]
5	-1	1	295424	ultralytics.nn.modules.Conv	[128, 256, 3, 2]
6	-1	2	788480	ultralytics.nn.modules.C2f	[256, 256, 2, True]
7	-1	1	1180672	ultralytics.nn.modules.Conv	[256, 512, 3, 2]
8	-1	1	1838080	ultralytics.nn.modules.C2f	[512, 512, 1, True]
9	-1	1	656896	ultralytics.nn.modules.SPPF	[512, 512, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.Concat	[1]
12	-1	1	591360	ultralytics.nn.modules.C2f	[768, 256, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.Concat	[1]
15	-1	1	148224	ultralytics.nn.modules.C2f	[384, 128, 1]
16	-1	1	147712	ultralytics.nn.modules.Conv	[128, 128, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.Concat	[1]
18	-1	1	493056	ultralytics.nn.modules.C2f	[384, 256, 1]
19	-1	1	590336	ultralytics.nn.modules.Conv	[256, 256, 3, 2]

Download Go... Gmail My Account Sign in - Googl... Higher Educat... YouTube Android Hub 4... OnlineShopin... localhost Gmail YouTube Maps a

object_detection_project.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:29 PM

+ Code + Text

Share notebook

[] %cd /content/drive/MyDrive/object/objectDetection

!yolo task=detect mode=train model=yolov8s.pt data= data.yaml epochs=10 imgsz=224 plots=True

	Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
	all	38	37	0.349	0.0278	0.0239	0.00531
	stright	38	10	0	0	0	0
	left	38	9	0.396	0.111	0.0902	0.0199
	right	38	8	0	0	0.00404	0.000945
	unexpected	38	10	1	0	0.00143	0.000374

Epoch	GPU_mem	box_loss	cls_loss	dfll_loss	Instances	Size	
2/10	0G	2.988	4.356	2.807	15	224: 100% 10/10 [00:53<00:00, 5.34s/it]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
	all	38	37	0.58	0.0556	0.0491	0.0102
	stright	38	10	0	0	0.00212	0.00081
	left	38	9	0.321	0.222	0.173	0.0348
	right	38	8	1	0	0.017	0.00448
	unexpected	38	10	1	0	0.00408	0.000798

Epoch	GPU_mem	box_loss	cls_loss	dfll_loss	Instances	Size	
3/10	0G	2.704	3.722	2.595	16	224: 100% 10/10 [00:53<00:00, 5.36s/it]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
	all	38	37	0.381	0.0868	0.0717	0.0205
	stright	38	10	0	0	0.00762	0.00198
	left	38	9	0.448	0.222	0.232	0.0646
	right	38	8	0.0751	0.125	0.0316	0.0131
	unexpected	38	10	1	0	0.0157	0.00217

Epoch	GPU_mem	box_loss	cls_loss	dfll_loss	Instances	Size	
4/10	0G	2.484	3.195	2.458	16	224: 100% 10/10 [00:55<00:00, 5.58s/it]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
	all	38	37	0.542	0.165	0.165	0.0556
	stright	38	10	0.0036	0.0	0.0000	0.0000

Download Go... Gmail My Account Sign in - Googl... Higher Educat... YouTube Android Hub 4... OnlineShopin... localhost Gmail YouTube Maps a

object_detection_project.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:29 PM

+ Code + Text

Connect

all 38 37 0.266 0.417 0.308 0.104

stright 38 10 0.599 0.9 0.673 0.21

left 38 9 0.327 0.444 0.345 0.149

right 38 8 0.138 0.323 0.198 0.0524

unexpected 38 10 0 0 0.0159 0.00432

10 epochs completed in 0.176 hours.
Optimizer stripped from runs/detect/train3/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/train3/weights/best.pt, 22.5MB

Validating runs/detect/train3/weights/best.pt...

Ultralytics YOLOv8.0.20 Python-3.10.12 torch-2.0.1+cu118 CPU

Model summary (fused): 168 layers, 11127132 parameters, 0 gradients, 28.4 GFLOPs

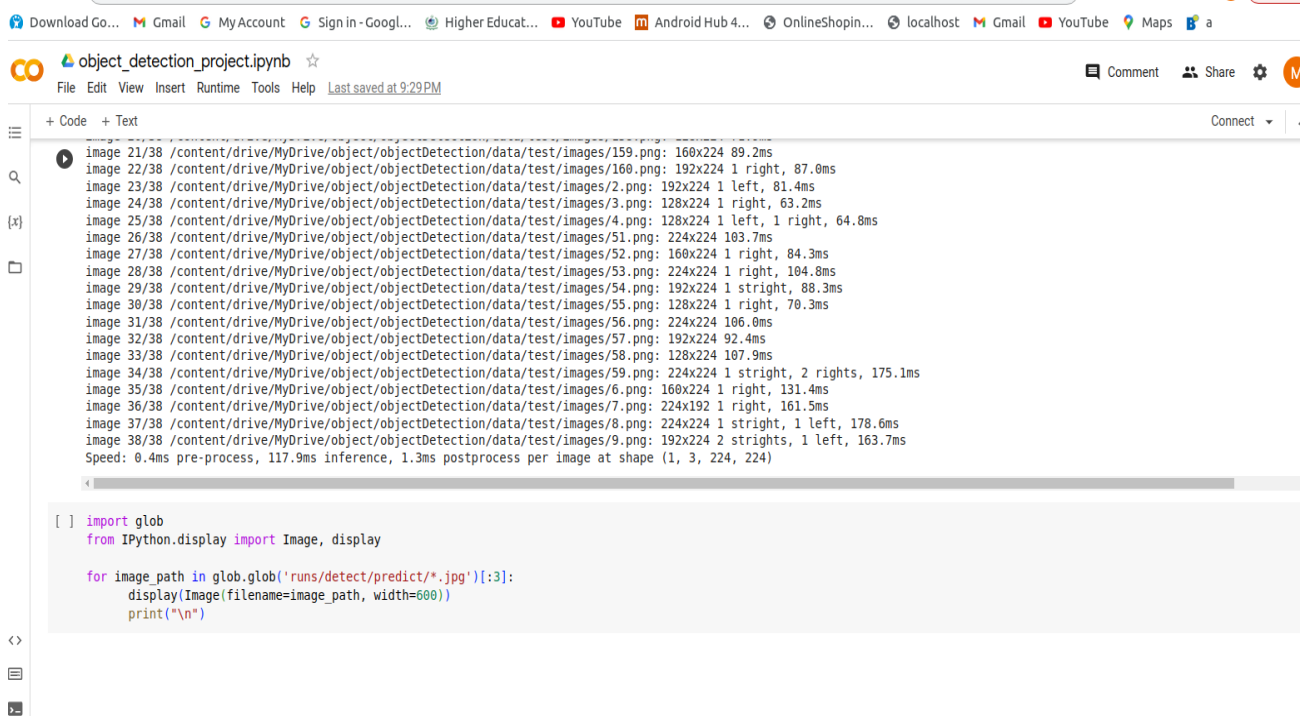
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
	all	38	37	0.265	0.416	0.309	0.105
	stright	38	10	0.596	0.9	0.673	0.209
	left	38	9	0.327	0.444	0.352	0.154
	right	38	8	0.137	0.319	0.197	0.052
	unexpected	38	10	0	0	0.016	0.00431

Speed: 0.4ms pre-process, 135.0ms inference, 0.0ms loss, 1.1ms post-process per image
Results saved to runs/detect/train3

confusion_matrix

[] !ls runs/detect/train/

args.yaml train_batch0.jpg train_batch2.jpg
results.csv train_batch1.jpg weights



Download Go... Gmail My Account Sign in - Googl... Higher Educat... YouTube Android Hub 4... OnlineShopin... localhost Gmail YouTube Maps a

object_detection_project.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 9:29 PM

Comment Share

+ Code + Text Connect

```
image 21/38 /content/drive/MyDrive/object/objectDetection/data/test/images/159.png: 160x224 89.2ms
image 22/38 /content/drive/MyDrive/object/objectDetection/data/test/images/160.png: 192x224 1 right, 87.0ms
image 23/38 /content/drive/MyDrive/object/objectDetection/data/test/images/2.png: 192x224 1 left, 81.4ms
image 24/38 /content/drive/MyDrive/object/objectDetection/data/test/images/3.png: 128x224 1 right, 63.2ms
image 25/38 /content/drive/MyDrive/object/objectDetection/data/test/images/4.png: 128x224 1 left, 1 right, 64.8ms
image 26/38 /content/drive/MyDrive/object/objectDetection/data/test/images/51.png: 224x224 103.7ms
image 27/38 /content/drive/MyDrive/object/objectDetection/data/test/images/52.png: 160x224 1 right, 84.3ms
image 28/38 /content/drive/MyDrive/object/objectDetection/data/test/images/53.png: 224x224 1 right, 104.8ms
image 29/38 /content/drive/MyDrive/object/objectDetection/data/test/images/54.png: 192x224 1 stright, 88.3ms
image 30/38 /content/drive/MyDrive/object/objectDetection/data/test/images/55.png: 128x224 1 right, 70.3ms
image 31/38 /content/drive/MyDrive/object/objectDetection/data/test/images/56.png: 224x224 106.0ms
image 32/38 /content/drive/MyDrive/object/objectDetection/data/test/images/57.png: 192x224 92.4ms
image 33/38 /content/drive/MyDrive/object/objectDetection/data/test/images/58.png: 128x224 107.9ms
image 34/38 /content/drive/MyDrive/object/objectDetection/data/test/images/59.png: 224x224 1 stright, 2 rights, 175.1ms
image 35/38 /content/drive/MyDrive/object/objectDetection/data/test/images/6.png: 160x224 1 right, 131.4ms
image 36/38 /content/drive/MyDrive/object/objectDetection/data/test/images/7.png: 224x192 1 right, 161.5ms
image 37/38 /content/drive/MyDrive/object/objectDetection/data/test/images/8.png: 224x224 1 stright, 1 left, 178.6ms
image 38/38 /content/drive/MyDrive/object/objectDetection/data/test/images/9.png: 192x224 2 strights, 1 left, 163.7ms
Speed: 0.4ms pre-process, 117.9ms inference, 1.3ms postprocess per image at shape (1, 3, 224, 224)
```

```
[ ] import glob
from IPython.display import Image, display

for image_path in glob.glob('runs/detect/predict/*.jpg')[1:3]:
    display(Image(filename=image_path, width=600))
    print("\n")
```

visualizations,

