YOLOv5: Unveiling State-of-the-Art Object Detection at 140 FPS

Zawar Khan[1]

1- Hochschule Mittweida - Dept of Mathematics
Mittweida - Germany

1 Introduction

The rapid development of Deep Learning (DL) has accelerated the progress of related methods, algorithms, and procedures in the fields of image processing and computer vision, offering a wide range of applications [**?**]. Experience the cutting-edge capabilities of $YOLOv5$, the state-of-the-art object detection model boasting an impressive speed of 140 frames per second. Backed by accurate and reproducible speed benchmarking results, $YOLOv5$ piqued my interest, prompting an in-depth assessment of its result quality and inference speed. $YOLOv5$ is one of the latest and often used versions of a very popular deep learning neural network used for various machine learning tasks, mainly in computer vision [**?**]. We will discuss in detail how we used the $FSOCO$ data set that helps Formula Student $FSAE$ teams get started with their visual perception system for driverless disciplines. What were the results and how we improve the results of $YOLOv5$? Let's delve into the details!

2 The YOLOv5 Model:

Implemented in PyTorch, the $YOLOv5$ model offers ease of understanding, training, and deployment. Architecturally akin to $YOLOv4$, a notable distinction lies in the utilization of Cross Stage Partial Network (CSP) to curtail computation costs. Training and running inferences with this model proved remarkably straightforward, and the release includes five model sizes: $YOLOv5s(smallest)$, $YOLOv5m$, $YOLOv5l$, and $YOLOv5x(largest)$. The $YOLOv5$ network consists of three main components: 1)*backbone*, 2)*neck*, and 3)*output*. First is data pre-processing tasks including mosaic data augmentation and adaptive image filling, $YOLOv5$ integrates adaptive anchor frame calculation on the input so that it can automatically set the initial anchor frame size when the dataset changes [**?**].

2.0.1 Backbone

The backbone is a convolutional neural network that aggregates and forms image features at different granularities. It mainly uses a cross-stage partial network (CSP) and spatial pyramid pooling (SPP), to extract feature maps of different sizes from the input image by multiple convolutions and pooling [**?**]. BottleneckCSP is used during inference(improving the speed) by reducing the amount

of calculations. while the SPP structure realizes the feature extraction from different scales for the same feature map and can generate three-scale feature maps, which helps improve the detection accuracy.

2.0.2 Neck

The neck consists of a series of layers of neural networks, to mix and combine image features and to pass them forward to the prediction. The feature pyramid structures of FPN and PAN are used. The FPN structure conveys strong semantic features from the top feature maps into the lower feature maps. At the same time, the PAN structure conveys strong localization features from lower feature maps into higher feature maps [**?**]. These structures strengthen the feature extraction mechanism from different network layers in Backbone fusion, improving the detection capability.

2.0.3 Output

Output is mainly used to predict targets of different sizes on feature maps.

3 Experimental dataset

Fig. 1: A sample image in the introduction.

The experimental dataset we used was the $FSOCO$ dataset consisting of manually annotated images that have been submitted by the Formula Student Driverless community. We provide ground truth data for cone detection and

support both bounding boxes and instance segmentation. My experience involved training $YOLO - v5$ on the FSOCO dataset, particularly valuable for autonomous driving applications. This dataset comprises images featuring safety cones strategically placed on roads or footpaths to redirect traffic safely. These cones vary in colors blue, large orange, orange, unknown, and yellow. The primary goal is to train models on this dataset for subsequent testing. The dataset is divided into training, testing, and validation sets, each with corresponding labeled datasets. The distribution includes 70 training images and labels, 30 testing images and labels, and a validation dataset sourced from the training set.

4 Base Model

4.0.1 Training

For the training of the model following augmentations were applied:

1. Data was divided into training, testing, and validation sets.
2. Hyperparameters set as:

   base model: $(yolov5m6.ptpre - trainedweights)$.
   train batch: 32.
   train epochs: 200.
   validation batch: 64.
   lr0: $0.01(initiallearningrate(SGD = 1E-2, Adam = 1E-3))$.
   lrf: $0.01(finalOneCycleLRlearningrate(lr0 * lrf))$.
   momentum: $0.937(SGDmomentum/Adambeta1)$.
   weight decay: $0.0005(optimizerweightdecay5e-4)$.
   warmup epochs: $3.0(warmupepochs(fractionsok))$.
   warmup bias lr: $0.1(warmupinitialbiaslr)$.
   box: $0.05(boxlossgain)$.
   cls: $0.5(clslossgain)$.
   cls pw: $1.0(clsBCELosspositive_weight)$.
   obj: $1.0(objlossgain(scalewithpixels))$.
   obj pw: $1.0(objBCELosspositive_weight)$.
   iou t: $0.20(IoUtrainingthreshold)$.
   anchor t: $4.0(anchor - multiplethreshold)$.
   fl gamma: $0.0(focallossgamma(efficientDetdefaultgamma = 1.5))$.
   hsv h: $0.015(imageHSV - Hueaugmentation(fraction))$.
   hsv s: $0.7(imageHSV - Saturationaugmentation(fraction))$.

hsv v: 0.4($imageHSV - Valueaugmentation(fraction)$).

degrees: 0.0($imagerotation(+/-deg)$).

translate: 0.1($imagetranslation(+/-fraction)$).

scale: 0.5($imagescale(+/-gain)$).

shear: 0.0($imageshear(+/-deg)$).

flipud: 0.0($imageflipup-down(probability)$).

fliplr: 0.5($imagefliplef t-right(probability)$).

Note that the $YOLOv5$ model was trained using the weights obtained after every 200 epoch.

Model completion summary: 200 epochs completed in 9.234 hours, having 276 layers, 35272008 parameters, 0 gradients, 48.9 GFLOPs. Image size were kept as 640 x 640

Class	**Images**	**Instances**	**P**	**R**	**mAP50**	**mAP50-95**
all	1387	26235	0.815	0.578	0.64	0.407
blue cone	1387	10127	0.862	0.641	0.709	0.454
large orange cone	1387	11073	0.883	0.629	0.704	0.445
orange cone	1387	915	0.889	0.706	0.778	0.551
unknown cone	1387	3248	0.869	0.667	0.725	0.466
yellow cone	1387	872	0.573	0.248	0.286	0.118

Table 1: Training Result

4.0.2 Validation

1. For the validation of the model following augmentations were applied.

   batch size: 64

   image size: 640

   confidence threshold: 0.001

   iou threshold: 0.6

   max det: 300

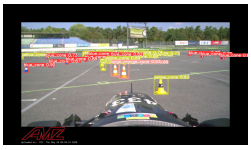
   workers: 8

   augment: $True$

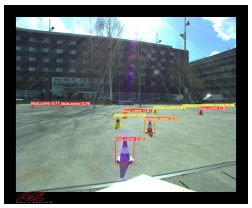
Model completion summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 GFLOPs, Speed 0.6ms pre-process, 36.1ms inference, 1.4ms NMS per image at shape $(64, 3, 640, 640)$

Class	Images	Instances	P	R	mAP50	mAP50-95
all	348	7174	0.774	0.535	0.618	0.392
blue cone	348	2851	0.795	0.571	0.629	0.401
large orange cone	348	3143	0.784	0.562	0.633	0.391
orange cone	348	223	0.871	0.601	0.697	0.486
unknown cone	348	797	0.786	0.665	0.733	0.473
yellow cone	348	160	0.635	0.275	0.396	0.208

Table 2: Validation Result



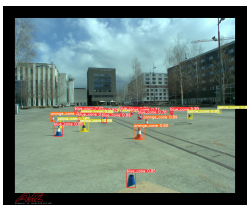
(a) 1a



(b) 1b

(c) 1c

(d) 1d



(e) 1d

Fig. 2: Test samples

4.0.3 Testing

1. For the testing of the model following augmentations were applied.

   image size: $[640, 640]$

   confidence threshold: 0.6

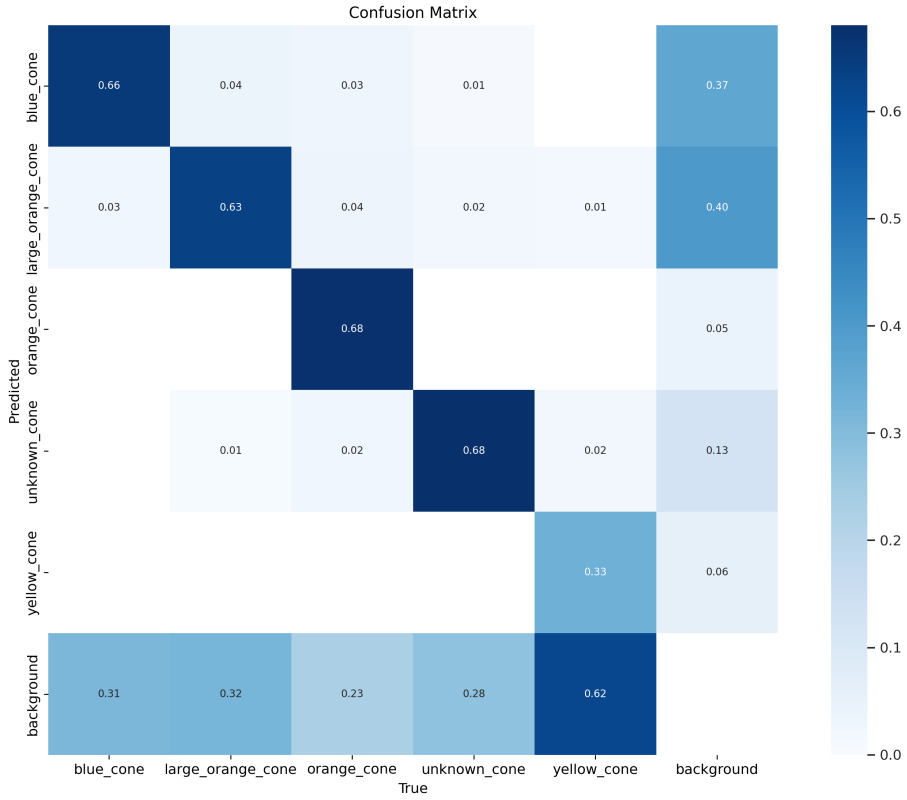
   iou threshold: 0.45

   max det: 1000

   line thickness: 3

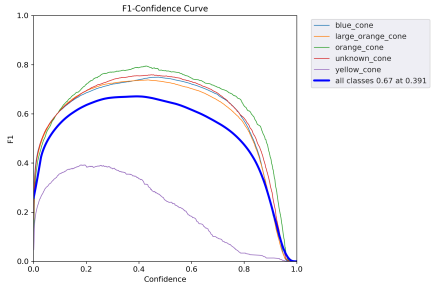
   vid stride: 1

Model completion summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 $GFLOPs$, Speed 0.4ms pre-process, 34.2ms inference, 1.0ms NMS per image at shape $(1, 3, 640, 640)$

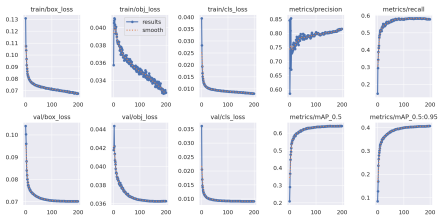


(a) 1a

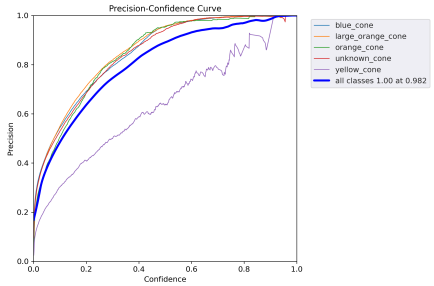
Fig. 3: Training Confusion Matrix



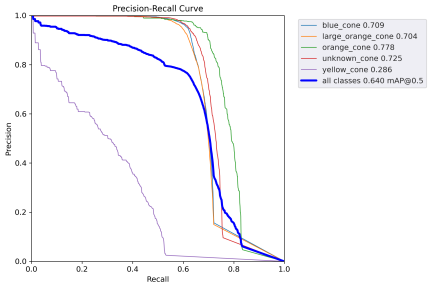
(a) Training F1 Curve



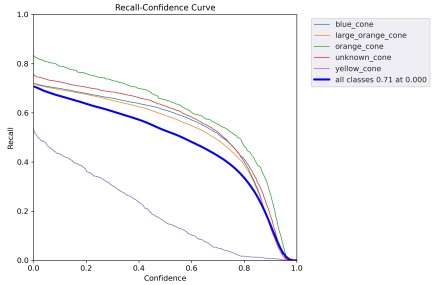
(b) Training Results



(c) Training Precission-Curve

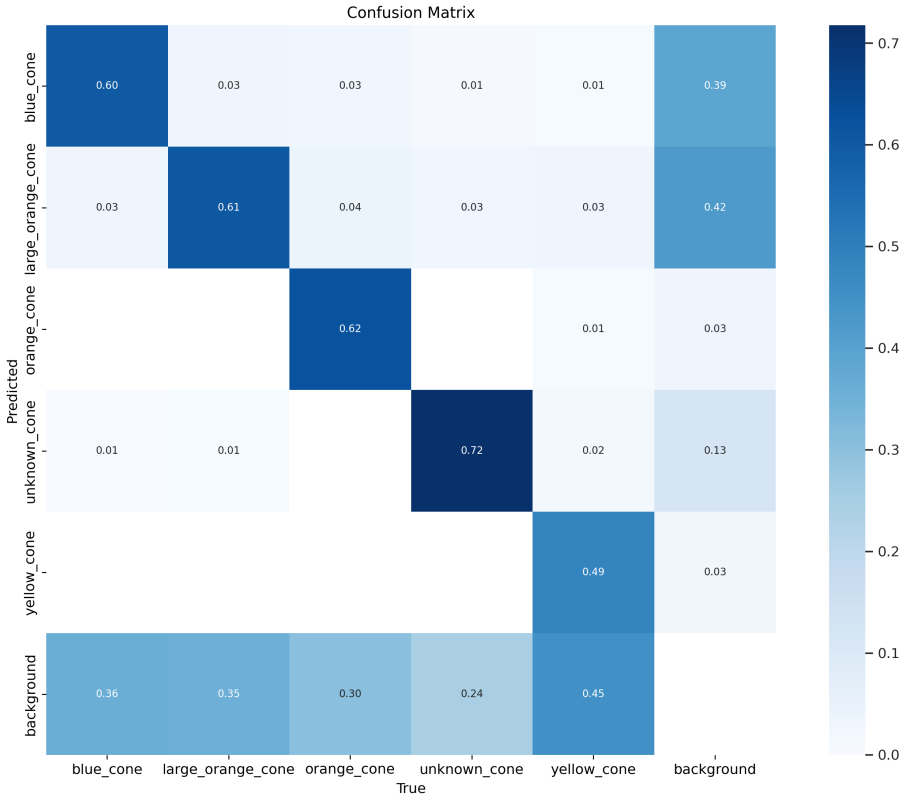


(d) Training PR-Curve



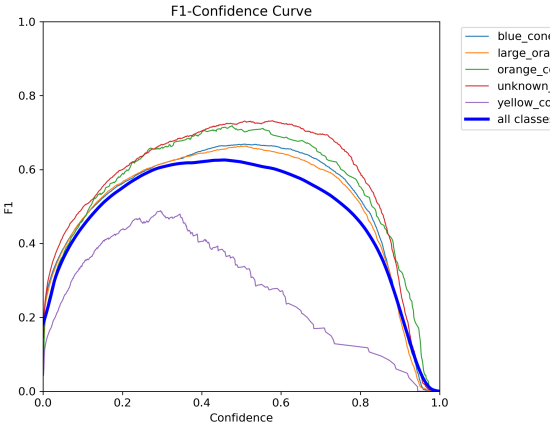
(e) Training Recall Curve

Fig. 4: Training Results

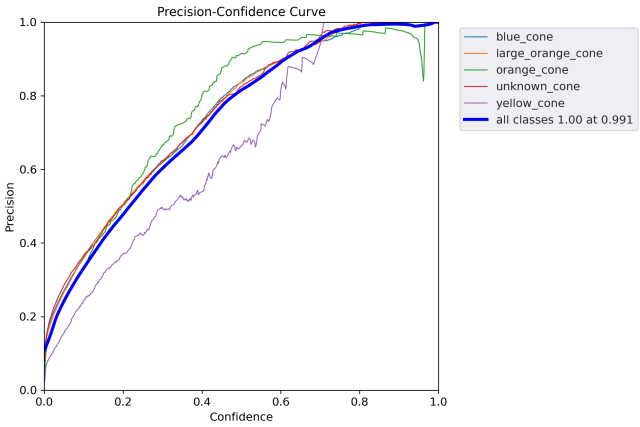


(a) 1a

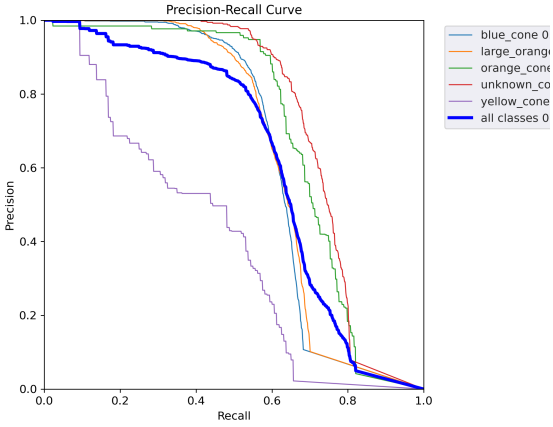
Fig. 5: Validation Confusion Matrix



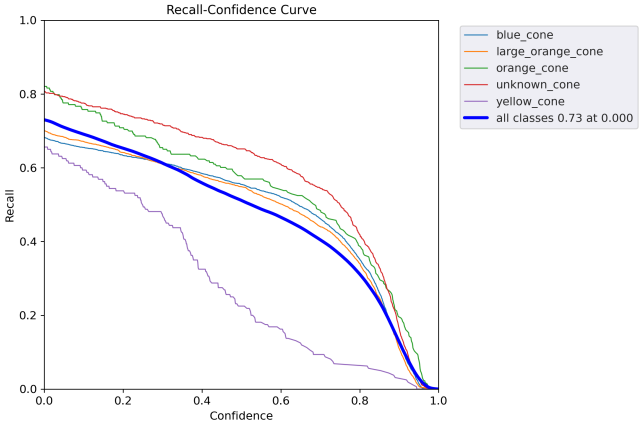
(a) validation F1 Curve



(b) validation Precission-Curve



(c) validation PR-Curve



(d) validation Recall Curve

Fig. 6: validation Results

5 Fine Tune the Model

1. For the fine tune training of the model following augmentations were applied.

   base model: $(yolov5m6.ptpre - trainedweights)$.
   train batch: 16.
   train epochs: 100.
   validation batch: 64.
   lr0: 0.00334$(initiallearningrate(SGD = 1E-2, Adam = 1E-3))$.
   lrf: 0.15135$(finalOneCycleLRlearningrate(lr0 * lrf))$.
   momentum: 0.74832$(SGDmomentum/Adambeta1)$.
   weight decay: 0.00025$(optimizerweightdecay5e-4)$.
   warmup epochs: 3.3835$(warmupepochs(fractionsok))$.
   warmup momentum: 0.59462$(warmupepochs(fractionsok))$.
   warmup bias lr: 0.18657$(warmupinitialbiaslr)$.
   box: 0.02$(boxlossgain)$.
   cls: 0.21638$(clslossgain)$.
   cls pw: 0.5$(clsBCELosspositive_weight)$.
   obj: 0.51728$(objlossgain(scalewithpixels))$.
   obj pw: 0.67198$(objBCELosspositive_weight)$.
   iou t: 0.2$(IoUtrainingthreshold)$.
   anchor t: 3.3744$(anchor - multiplethreshold)$.
   fl gamma: 0.0$(focallossgamma(efficientDetdefaultgamma = 1.5))$.
   hsv h: 0.01041$(imageHSV - Hueaugmentation(fraction))$.
   hsv s: 0.54703$(imageHSV - Saturationaugmentation(fraction))$.
   hsv v: 0.27739$(imageHSV - Valueaugmentation(fraction))$.
   degrees: 0.0$(imagerotation(+/-deg))$.
   translate: 0.04591$(imagetranslation(+/-fraction))$.
   scale: 0.75544$(imagescale(+/-gain))$.
   shear: 0.0$(imageshear(+/-deg))$.
   flipud: 0.0$(imageflipup - down(probability))$.
   fliplr: 0.5$(imagefliplef t - right(probability))$.

Note that the $YOLOv5$ model was trained using the weights obtained after every 100 epoch.

Model starting summary: 379 layers, 35299032 parameters, 35299032 gradients, 49.3 $GFLOPs$,

Model completion summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 $GFLOPs$, 100 epochs completed in 8.756 hours.

Class	Images	Instances	P	R	mAP50	mAP50-95
all	1387	26235	0.837	0.639	0.705	0.465
blue cone	1387	10127	0.912	0.713	0.788	0.526
large orange cone	1387	11073	0.913	0.713	0.796	0.525
orange cone	1387	915	0.908	0.728	0.808	0.588
unknown cone	1387	3248	0.891	0.73	0.803	0.536
yellow cone	1387	872	0.563	0.309	0.329	0.15

Table 3: Training Result

5.0.1 Validation

1. For the validation of the model following augmentations were applied.

   batch size: 64

   image size: 640

   confidence threshold: 0.001

   iou threshold: 0.6

   max det: 300

   workers: 8

   augment: $True$

Model summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 $GFLOPs$, Speed 0.4ms pre-process, 34.3ms inference, 1.1ms NMS per image at shape $(64, 3, 640, 640)$

Class	Images	Instances	P	R	mAP50	mAP50-95
all	348	7174	0.766	0.627	0.686	0.441
blue cone	348	2851	0.813	0.663	0.717	0.463
large orange cone	348	3143	0.798	0.67	0.736	0.462
orange cone	348	223	0.867	0.632	0.75	0.529
unknown cone	348	797	0.792	0.749	0.805	0.532
yellow cone	348	160	0.559	0.419	0.422	0.221

Table 4: Validation Result

5.0.2 Testing

1. For the testing of the model following augmentations were applied.

   image size: $[640, 640]$

   confidence threshold: 0.6

   iou threshold: 0.45

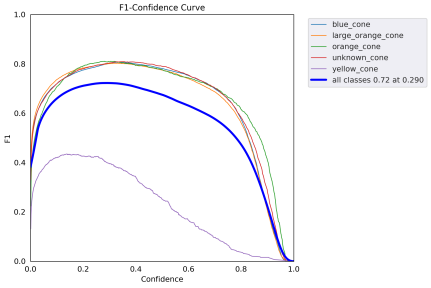
   max det: 1000

   line thickness: 3

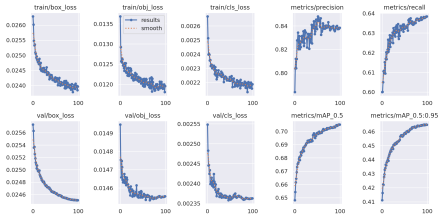
   vid stride: 1

Model completion summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 $GFLOPs$, Speed: $0.5ms$ pre-process, $35.4ms$ inference, $1.0ms$ NMS per image at shape $(1, 3, 640, 640)$

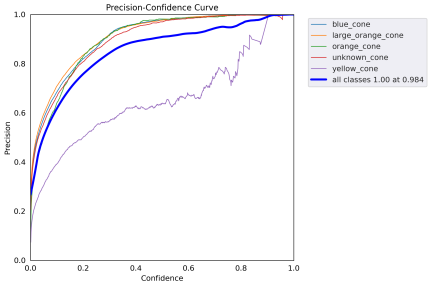
5.0.3 Fine Tune Train Results



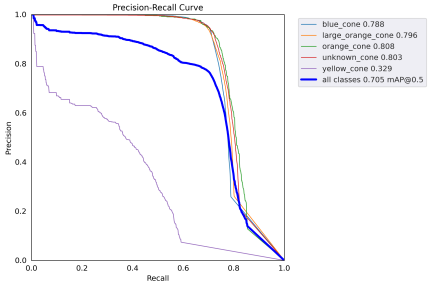
(a) F1 Curve



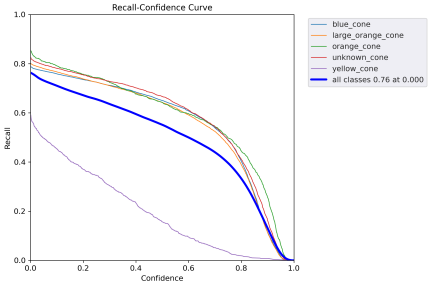
(b) Results



(c) Precission-Curve

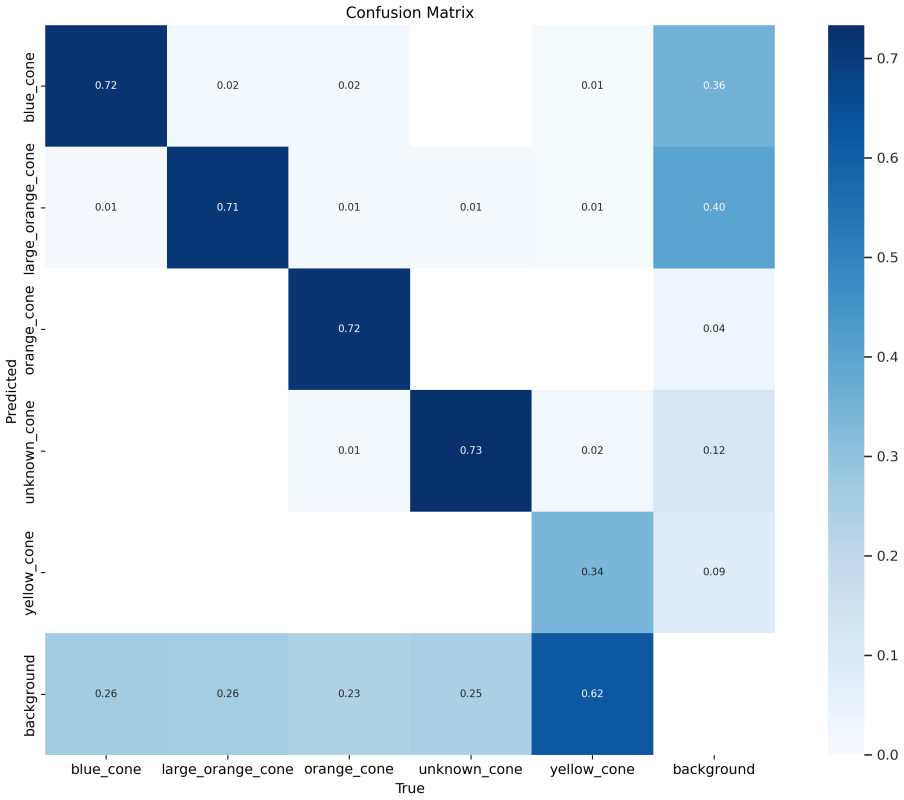


(d) PR-Curve



(e) Recall Curve

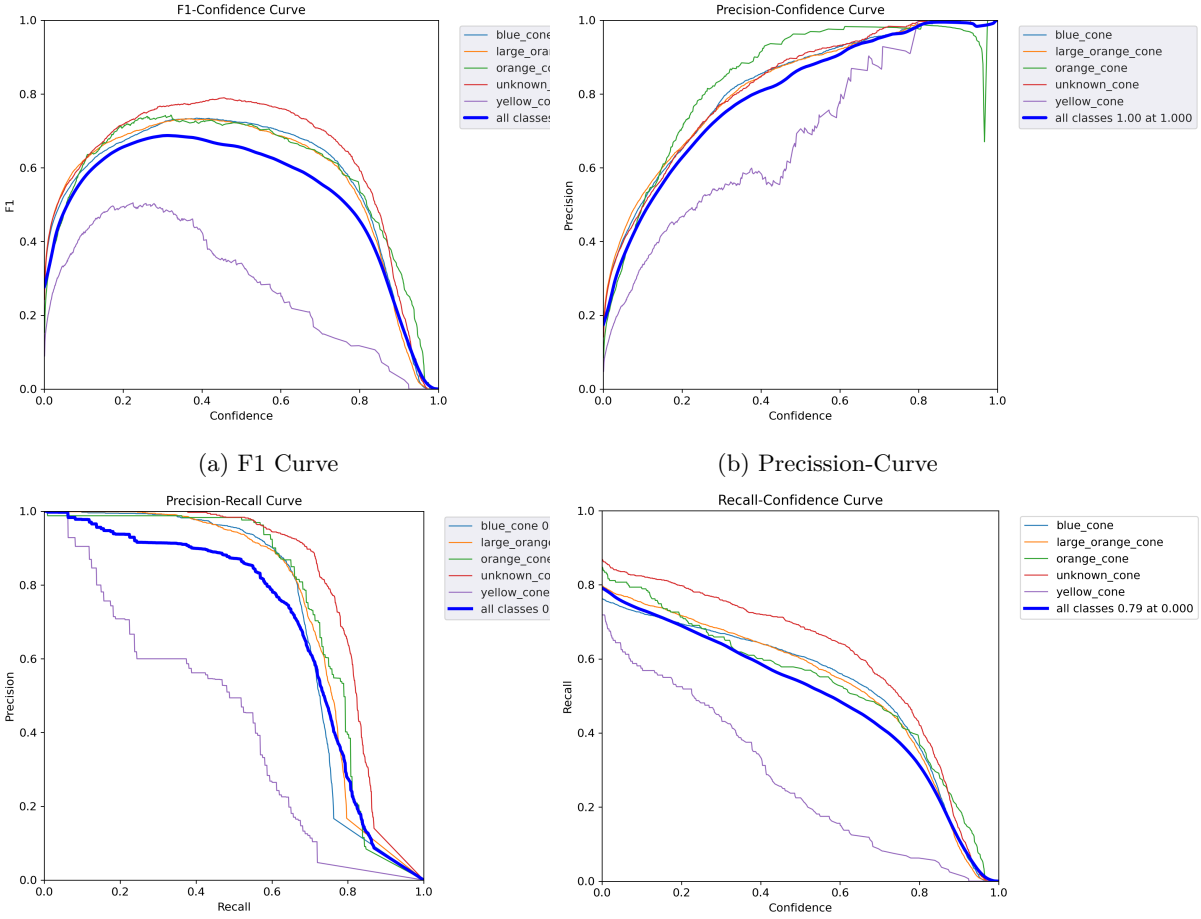
Fig. 7: Fine Tune Training Results



(a) 1a

Fig. 8: Confusion Matrix

5.0.4 *Fine Tune Validation Results*



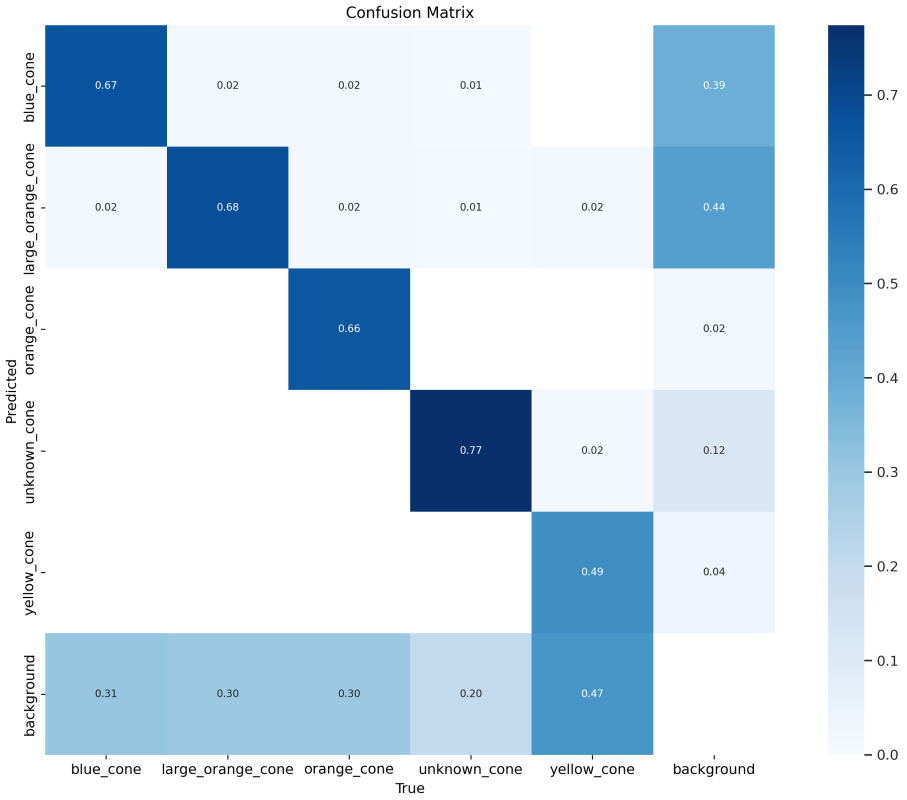
(a) F1 Curve

(b) Precission-Curve

(c) PR-Curve

(d) Recall Curve

Fig. 9: Fine Tune Validation Results



(a) 1a

Fig. 10: Confusion Matrix

6 Fine Tune With Background Images

1. Base model training had occurrence of FP ($FalsePositives$) in the model. To avoid FP, during fine-tuning of the model, 10% of background images ($Images - without - labels$) of the original training and validation were added to the training set and validation set with the following augmentations.

   base model: ($yolov5m6.ptpre - trainedweights$).

   train batch: 16.

   train epochs: 100.

   validation batch: 64.

   lr0: 0.01($initiallearningrate(SGD = 1E - 2, Adam = 1E - 3)$).

   lrf: 0.01($finalOneCycleLRlearningrate(lr0 * lrf)$).

   momentum: 0.937($SGDmomentum/Adambeta1$).

   weight decay: 0.0005($optimizerweightdecay5e - 4$).

   warmup epochs: 3.0($warmupepochs(fractionsok)$).

   warmup momentum: 0.8($warmupepochs(fractionsok)$).

   warmup bias lr: 0.1($warmupinitialbiaslr$).

   box: 0.05($boxlossgain$).

   cls: 0.5($clslossgain$).

   cls pw: 1.0($clsBCELosspositive_weight$).

   obj: 1.0($objlossgain(scalewithpixels)$).

   obj pw: 1.0($objBCELosspositive_weight$).

   iou t: 0.20($IoUtrainingthreshold$).

   anchor t: 4.0($anchor - multiplethreshold$).

   fl gamma: 0.0($focallossgamma(efficientDetdefaultgamma = 1.5)$).

   hsv h: 0.015($imageHSV - Hueaugmentation(fraction)$).

   hsv s: 0.7($imageHSV - Saturationaugmentation(fraction)$).

   hsv v: 0.4($imageHSV - Valueaugmentation(fraction)$).

   degrees: 0.0($imagerotation(+/ - deg)$).

   translate: 0.1($imagetranslation(+/ - fraction)$).

   scale: 0.5($imagescale(+/ - gain)$).

   shear: 0.0($imageshear(+/ - deg)$).

   flipud: 0.0($imageflipup - down(probability)$).

   fliplr: 0.5($imagefliplef t - right(probability)$).

Note that the $YOLOv5$ model was fine tuned using the weights obtained after every 100 epoch.

Model starting summary: 379 layers, 35299032 parameters, 35299032 gradients, 49.3 $GFLOPs$.

Model completion summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 $GFLOPs$. 100 epochs completed in 9.407 hours.

Class	**Images**	**Instances**	**P**	**R**	**mAP50**	**mAP50-95**
all	1525	26235	0.859	0.647	0.73	0.485
blue cone	1525	10127	0.938	0.731	0.817	0.546
large orange cone	1525	11073	0.936	0.733	0.832	0.551
orange cone	1525	915	0.936	0.729	0.835	0.612
unknown cone	1525	3248	0.92	0.753	0.832	0.563
yellow cone	1525	872	0.565	0.289	0.334	0.156

Table 5: Training Result With Background Images Added

6.0.1 Validation

1. For the validation of the model following augmentations were applied.

   batch size: 64

   image size: 640

   confidence threshold: 0.001

   iou threshold: 0.6

   max det: 300

   workers: 8

   augment: $True$

Model summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 $GFLOPs$, Speed 0.1ms pre-process, 29.3ms inference, 1.4ms NMS per image at shape $(64, 3, 640, 640)$.

Class	**Images**	**Instances**	**P**	**R**	**mAP50**	**mAP50-95**
all	348	7174	0.776	0.649	0.722	0.463
blue cone	348	2851	0.828	0.685	0.753	0.49
large orange cone	348	3143	0.827	0.703	0.778	0.492
orange cone	348	223	0.864	0.709	0.804	0.563
unknown cone	348	797	0.831	0.801	0.853	0.562
yellow cone	348	160	0.533	0.35	0.424	0.207

Table 6: Validation Result With Background Images Added

6.0.2 Testing

1. For the testing of the model following augmentations were applied.

   image size: $[640, 640]$

   confidence threshold: 0.6

   iou threshold: 0.45

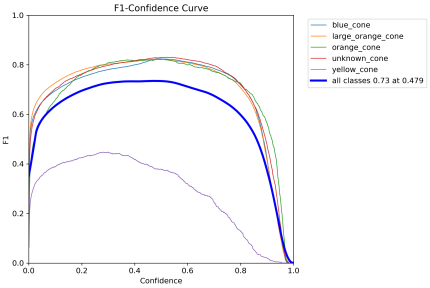
   max det: 1000

   line thickness: 3

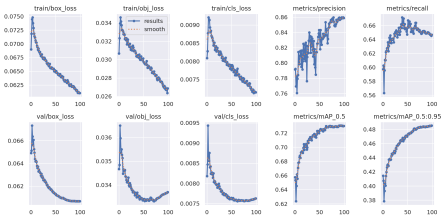
   vid stride: 1

Model summary: 276 layers, 35272008 parameters, 0 gradients, 48.9 $GFLOPs$, Speed $0.5ms$ pre-process, $41.3ms$ inference, $1.2ms$ NMS per image at shape $(1, 3, 640, 640)$.

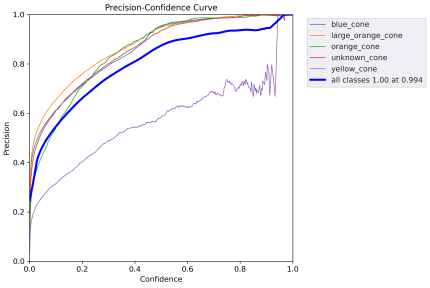
6.0.3 Fine Tune Train Results With Background Images



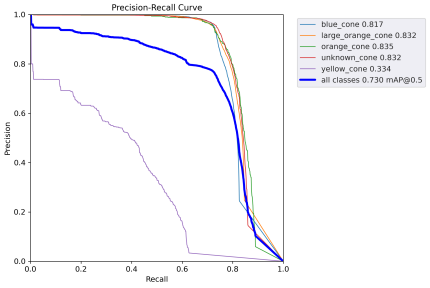
(a) F1 Curve



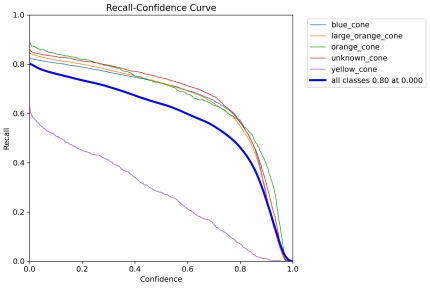
(b) Results



(c) Precission-Curve

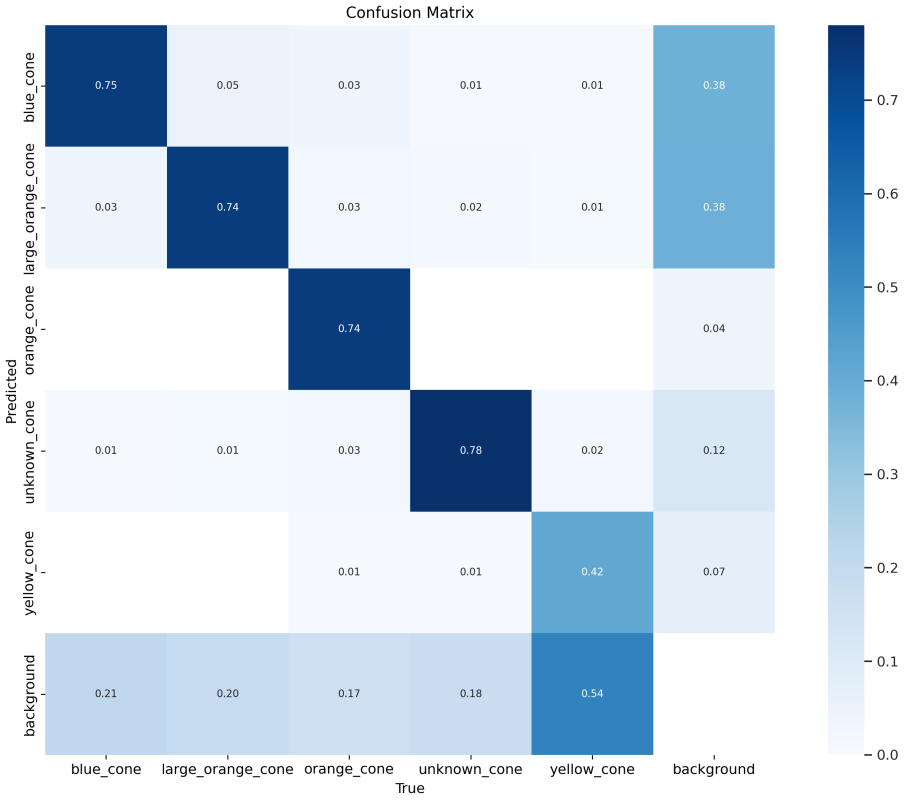


(d) PR-Curve



(e) Recall Curve

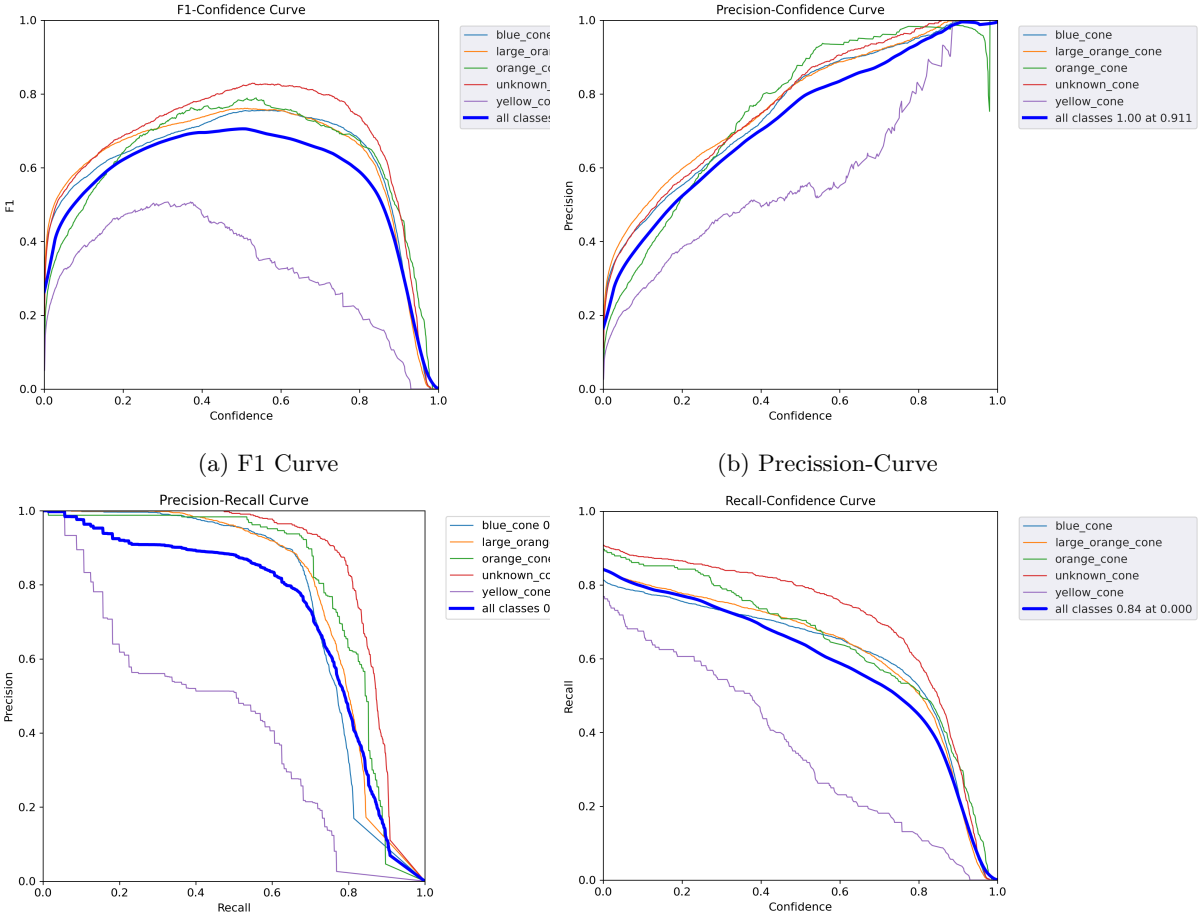
Fig. 11: Fine Tune Training Results With Background Images



(a) 1a

Fig. 12: Confusion Matrix With Background Images Added

6.0.4 Fine Tune Validation Results With Background Images



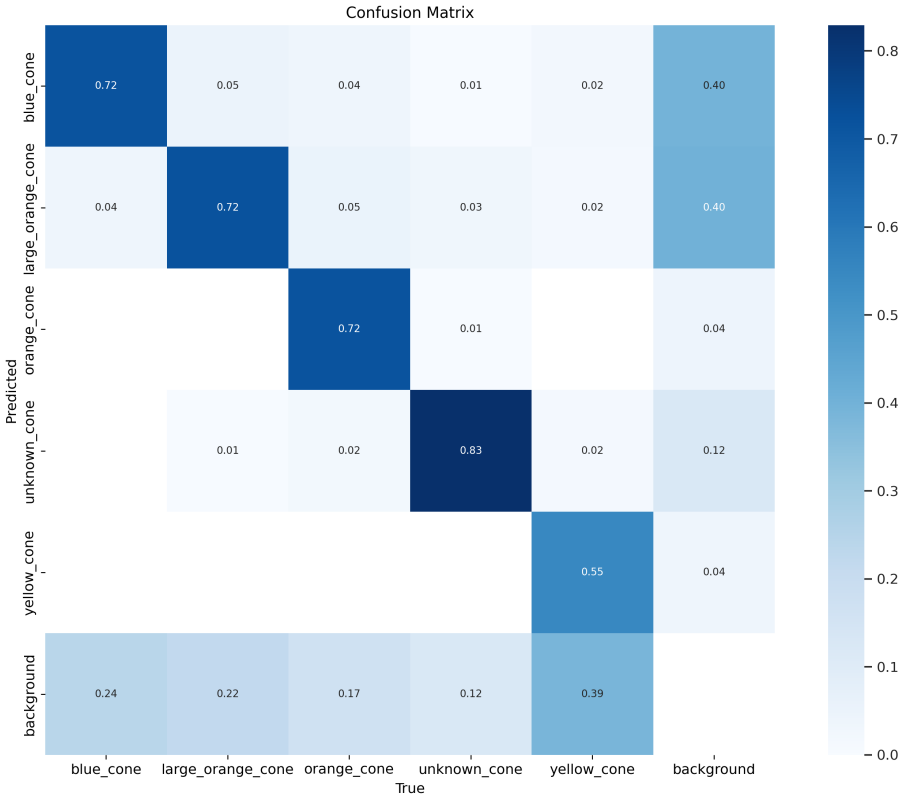
(a) F1 Curve

(b) Precission-Curve

(c) PR-Curve

(d) Recall Curve

Fig. 13: Fine Tune Validation Results With Background Images Added



(a) 1a

Fig. 14: Confusion Matrix With Background Images Added

7 Train With Background and Augmented images

7.0.1 Background

The background image collection is derived from various datasets that have been amalgamated to create a unified dataset comprising a total of 1500 images. Background images contribute to False Positives FP during the training process. The model tends to focus on background instances that are not pertinent for training. To guide the model towards a state where it dismisses these background instances, it is essential to supply it with background images that closely resemble our original dataset. In our case, these images encompass motor vehicles, roads, road signs, sky, pedestrians, buildings, etc.

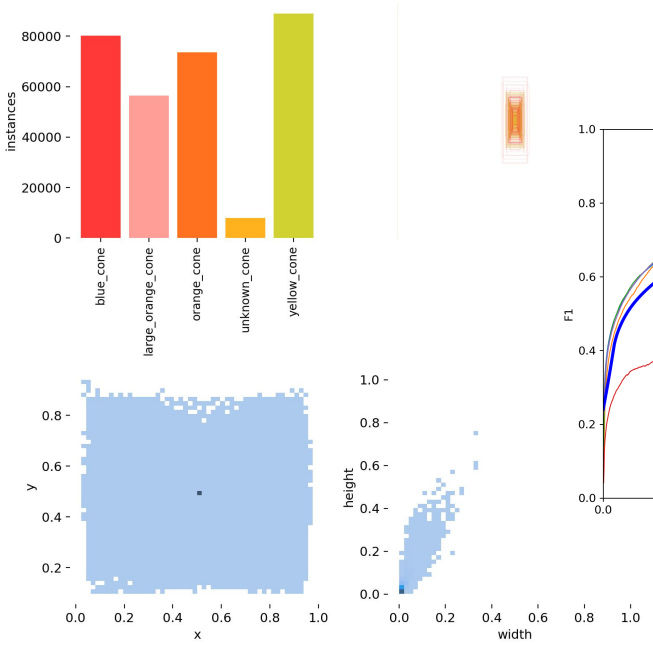
To mitigate false positives, we crafted a dataset with these objects in mind and incorporated them into the training data. Notably, we refrained from providing labels for these objects, allowing the model to perceive them as background and, consequently, learn to disregard them during learning and finetuning. Improving the overall accuracy.

7.0.2 Augmentation

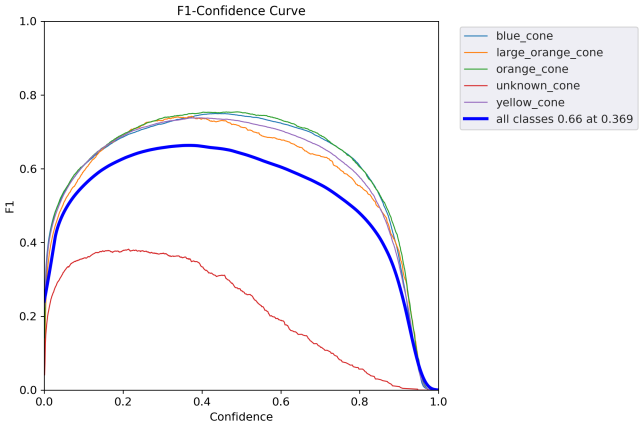
Augmentation of the dataset is a crucial aspect of preprocessing and dataset creation. We employed image augmentation because, during model training, it became evident that certain classes had more instances than others, leading to overfitting on classes with higher instance counts. To address this issue, crops were extracted from all instances, ensuring each crop was of size 640×640 as required by YOLOv5. Labels were concurrently generated, incorporating the class name and normalized coordinates within the range [0, 1]. Since all crops were of uniform size, the normalized coordinates remained consistent across all instances.

Upon observing that some classes had limited instances in the base model training, we applied image augmentation to the crops. This involved selecting a crop and rotating it at seven angles: 30°, 45°, 90°, 150°, 180°, 250°, and 270° (resulting in seven times the original number of instances). Furthermore, to augment the count, each crop underwent seven techniques such as horizontal flip, vertical flip, rotation, flips, noise addition, etc. This translated to the total instances of crops, including their rotated counterparts, multiplied by seven. Concurrently, labels were generated following a naming convention similar to their respective images, enhancing the overall instance count to prevent model overfitting on specific classes.

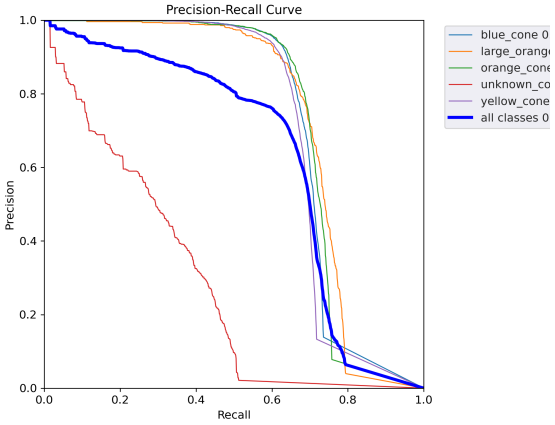
7.0.3 *Train With Augmention With and Without Background*



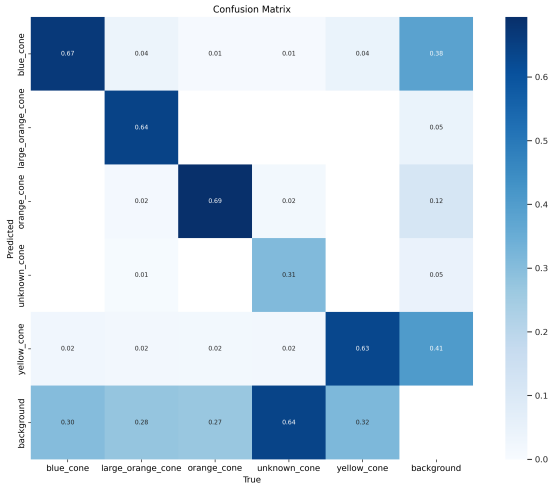
(a) Improved Instances



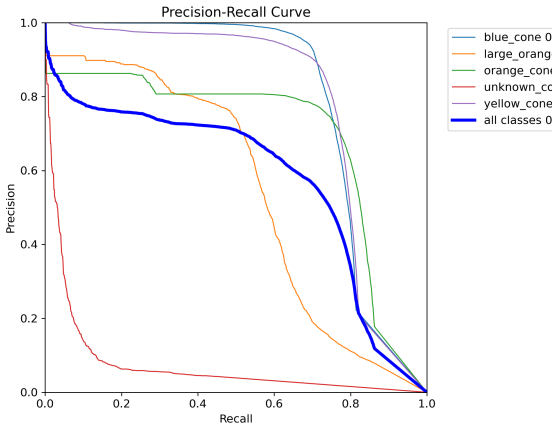
(b) F1 Curve



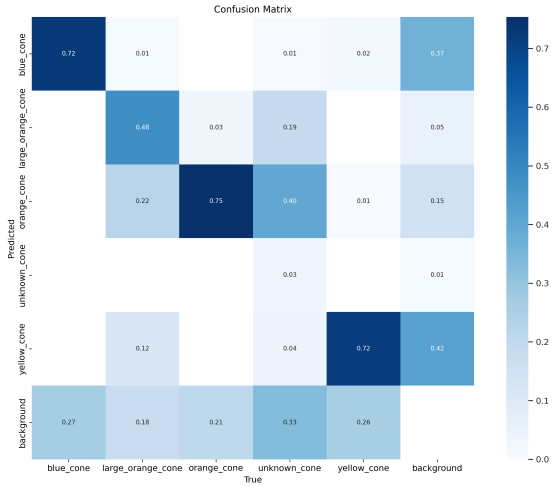
(a) PR-Curve



(b) Confusion Matrix With Augmented Images Without Background



(a) PR-Curve With Augmentation And Background



(b) Confusion Matrix With Augmented Images With Background

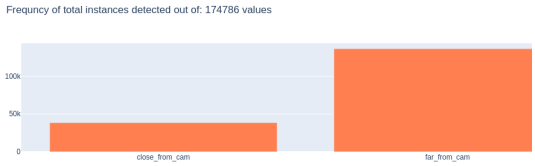
8 Distance Estimation And Depth Perspective For Analyzing Object Proximity in Autonomous Driving

In pursuit of developing a model tailored for autonomous driving applications, our primary focus was on training it, to recognize various types of traffic cones employed for vehicular lane divergence on roads. These cones encompassed five distinct types: blue, large orange, orange, yellow, and instances labeled as 'unknown.' While our model successfully predicted the presence of these cones, our principal aim was to ascertain whether the model demonstrated enhanced proficiency in detecting objects or instances in close proximity to the vehicle as compared to those farther away.

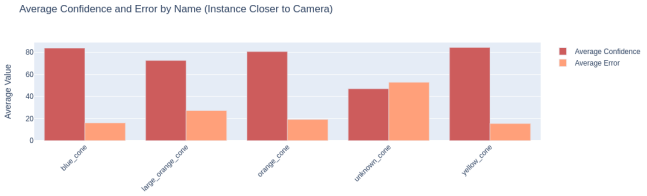
To address this objective, we leveraged our dataset for testing purposes. The model was subjected to test images featuring traffic cones, and it exhibited a high level of accuracy in detecting these objects. The output comprised six key parameters $X - axis$, $Y - axis$, $Width$, $Height$, $Confidence$, and $Class$. We subsequently implemented a Region of Interest (ROI) strategy on the image, selectively focusing on objects deemed closer to the car based on a predefined confidence threshold. Our hypothesis posited that objects in closer proximity to the vehicle would exhibit larger areas, calculated as the product of width and height. The confidence threshold was set at 3000.0, and objects surpassing this area criterion were designated with a *depthid* of 0 and a *depthname* of *closertocar*, while those falling below were assigned a *depthid* of 1 and a *depthname* of $farawayfromcar$. This data was systematically organized into a Pandas dataframe.

Acknowledging the speculative nature of our confidence hypothesis, we undertook measures to validate the accuracy of our data collection and hypothesis. We introduced the concept of $focallength$, assuming the camera's constant positioning at the *imagecenter*. Subsequently, we computed the distance between the camera and the height of the *boundingbox*, affirming the correctness of our hypothesis. Further analysis was conducted based on the probability of detecting objects closer to the car in comparison to those farther away. The rationale underlying this analysis was the anticipation that objects in closer proximity to the vehicle would be detected with a higher probability than those situated at a greater distance. link: To Analysis

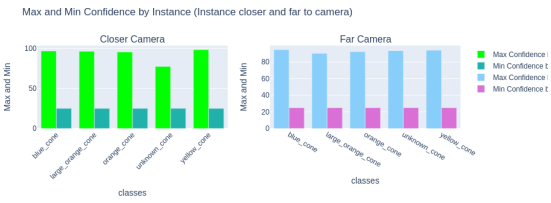
8.0.1 Analysis With ROI



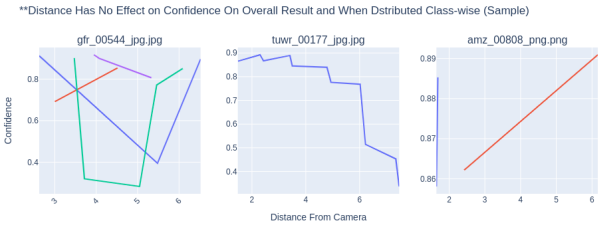
(a) Frequncy Of Total Instances



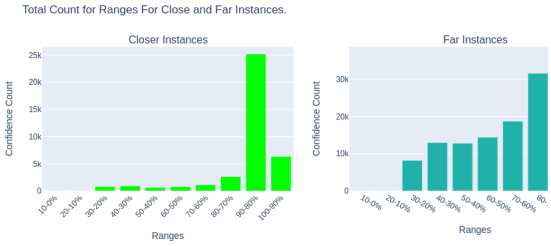
(b) Average Confidence and Error by Name (Instance Closer to Camera)Average Confidence and Error by Name (Instance Closer to Camera)



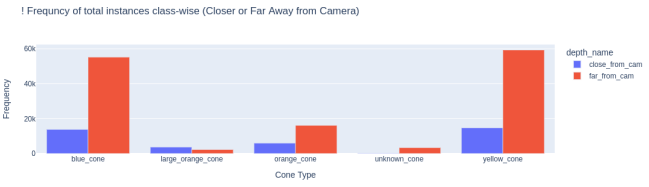
(a) Max and Min Confidence by Instance (Instance closer and far to camera)



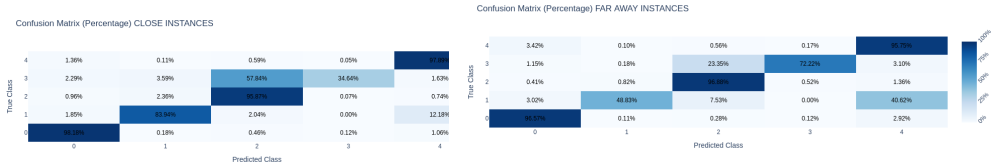
(b) Distance Has No Effect on Confidence On Overall Result and When Dstributed Class-wise (Sample)



(a) Total Count for Ranges For Close and Far Instances.



(b) Frequncy of total instances class-wise (Closer or Far Away from Camera)



(a) Confusion Matrix For Close Instances. (b) Confusion Matrix For Far Away Instances

Fig. 21: Confusion Matrix For Close And Far Away Instances

9 Yolov8

9.1 Training and Finetuning With Yolov8

The $YOLOv8$ architecture has been meticulously designed to facilitate the efficient training of object detection models, fully leveraging the capabilities of contemporary hardware. This endeavor was undertaken to conduct a comprehensive comparative analysis between $YOLOv5$ and $YOLOv8$, with the aim of evaluating their respective performance under identical hyperparameter settings. The hyperparameters employed for $YOLOv8$ encompass the original $FSOCO$ dataset, augmented images, and background data—effectively minimizing both false positives (FP) and false negatives (FN).

However, an observed limitation with YOLOv8 pertained to memory issues, specifically, the occurrence of memory leakage. This challenge stemmed from the substantial space requirements associated with the creation of *.npy* files for each image, leading to an excessive utilization of storage resources. Addressing these memory-related concerns is crucial for enhancing the overall efficiency and practical applicability of the YOLOv8 model.

9.2 Hyperparameters For Yolov8

During the training phase of the base model, several augmentations were incorporated, and the dataset was partitioned into training, testing, and validation sets. The base model underwent training using the $FSOCO$ data, background information, and augmented images from the training process. This approach was adopted to systematically assess the model's performance and generalization capabilities, providing insights into its effectiveness with the influence of additional augmented data.

1. Hyperparameters set as:

   base model: $(yolov8m.ptpre - trainedweights)$.

   train batch: 32.

   train epochs: 200.

   validation batch: 16.

lr0: 0.00334($initiallearningrate(SGD = 1E-2, Adam = 1E-3)$).
lrf: 0.15135($finalOneCycleLRlearningrate(lr0 * lrf)$).
momentum: 0.74832($SGDmomentum/Adambeta1$).
weight decay: 0.00025($optimizerweightdecay5e-4$).
warmup epochs: 3.3835($warmupepochs(fractionsok)$).
warmup momentum: 0.59462.
warmup bias lr: 0.18657($warmupinitialbiaslr$).
box: 0.02($boxlossgain$).
cls: 0.21638($clslossgain$).
dfl: 0.15($dfllossgain$).
pose: 12.0($poselossgain(pose-only)$).
kobj: 0.51728$keypointobjlossgain(pose-only)$.
label smoothing: 0.0$labelsmoothing(fraction)$.
nbs: 64($nominalbatchsize$).
hsv h: 0.01041($imageHSV-Hueaugmentation(fraction)$).
hsv s: 0.54703($imageHSV-Saturationaugmentation(fraction)$).
hsv v: 0.27739($imageHSV-Valueaugmentation(fraction)$).
degrees: 0.0($imagerotation(+/-deg)$).
translate: 0.04591($imagetranslation(+/-fraction)$).
scale: 0.75544($imagescale(+/-gain)$).
shear: 0.0($imageshear(+/-deg)$).
perspective: 0.0.
flipud: 0.0($imageflipup-down(probability)$).
fliplr: 0.5($imagefliplefl-right(probability)$).
mosaic: 0.85834($(int)disablemosaicaugmentationforfinalepochs(0todisable)$).
mixup: 0.04266.
copy paste: 0.0.
auto augment: $randaugment$.
erasing: 0.04.
crop fraction: 1.0.
optimizer: $Adamax(optimizertouse, choices = [SGD, Adam, Adamax, AdamW, NAdam, RAdan$

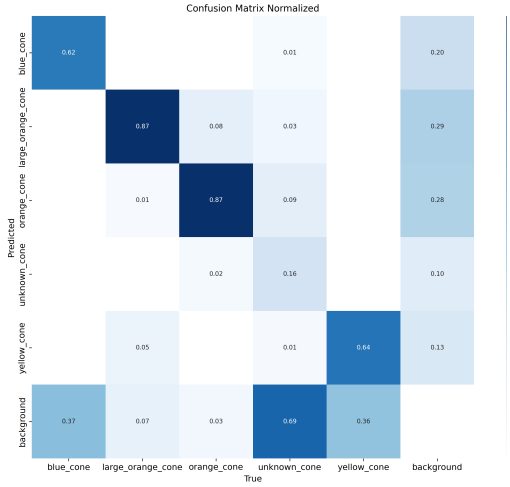
200 epochs completed in 35.957 hours. Speed 0.1ms preprocess, 2.7ms inference, 0.0ms loss, 0.7ms postprocess per image. Model summary (fused): 295 layers, 25859215 parameters, 25859199 gradients, 79.1 $GFLOPs$.

Class	Images	Instances	P	R	mAP50	mAP50-95
all	13560	31223	0.698	0.653	0.674	0.544
blue cone	13560	8248	0.881	0.632	0.769	0.48
large orange cone	13560	2772	0.56	0.874	0.806	0.774
orange cone	13560	12118	0.981	0.924	0.981	0.957
unknown cone	13560	729	0.19	0.182	0.0728	0.0331
yellow cone	13560	7356	0.879	0.651	0.74	0.475

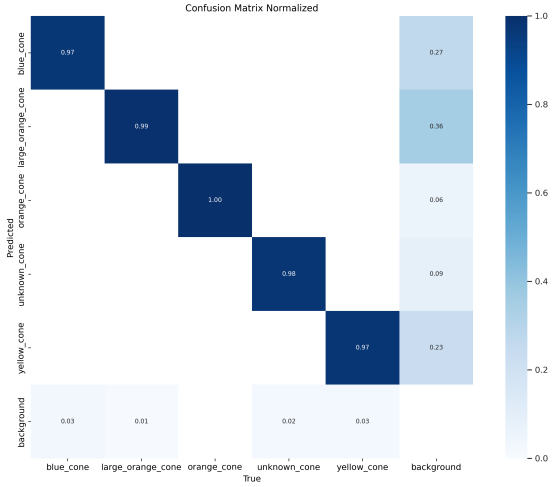
Table 7: Training Result With Background And Augmentation

Class	Images	Instances	P	R	mAP50	mAP50-95
all	13560	31223	1	0.983	0.985	0.985
blue cone	13560	8248	1	0.972	0.98	0.98
large orange cone	13560	2772	1	0.994	0.995	0.995
orange cone	13560	12118	1	1	0.995	0.995
unknown cone	13560	729	1	0.978	0.976	0.976
yellow cone	13560	7356	1	0.973	0.979	0.979

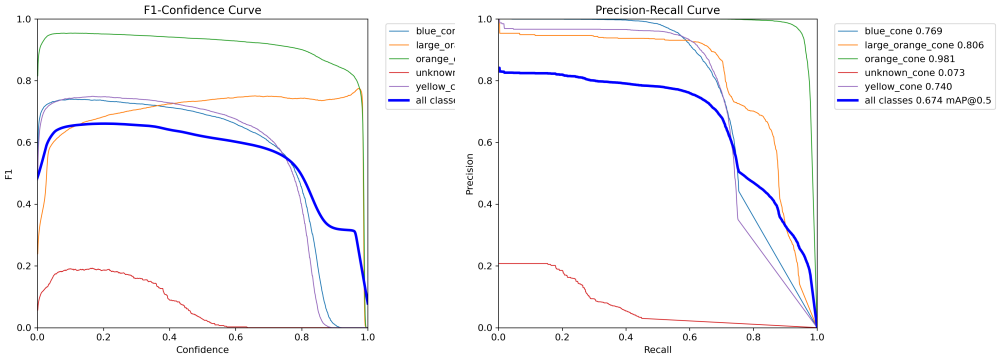
Table 8: Vaidation Result With Background And Augmentation



(a) V8 Train Base Model Confuion Matrix

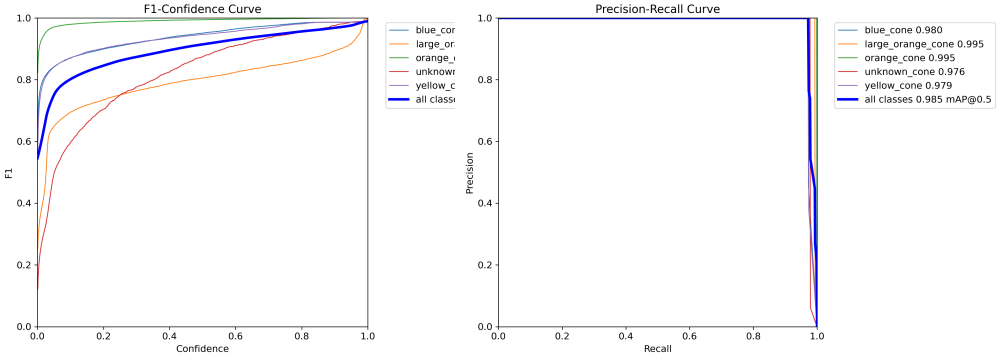


(b) V8 Validation Base Model Confuion Matrix



(a) V8 Training F1 Curve (b) V8 Training PR Curve

Fig. 23: Training F1 and PR Curves



(a) V8 Validation F1 Curve (b) V8 Validation PR Curve

Fig. 24: Validation F1 and PR Curves

10 Conclusion

In conclusion, our study focused on training a model using fsoco data, yielding noteworthy results. Notably, we observed an accuracy loss for background instances during training. To address this, we incorporated 10*percent* of background images into the training dataset and fine-tuned the model over 100 epochs. This strategic augmentation led to enhanced learning, resulting in improved model performance. Further refinement was achieved by introducing augmented crops, specifically targeting instances with limited representation. This step was pivotal in mitigating the risk of overfitting to certain classes arising from an imbalanced dataset. Post-training, a significant improvement in model efficacy was observed.

Subsequently, we conducted tests on objects positioned both closer and farther away from the car, employing the minimum distance between the center of the image and the object. This comprehensive evaluation provided insights into the model's robustness across varying object distances.Parallel experiments were conducted using YOLOv8, acknowledging its higher resource demands in terms of time and memory during training. However, the results obtained justified the investment, showcasing superior performance in both training and validation phases. Key hyperparameters, referred to as golden hyperparameters, were carefully curated and documented to ensure reproducibility. Additionally, the best-weight file resulting from this experiment has been included for reference. This meticulous approach, encompassing data augmentation, fine-tuning, and strategic testing, along with detailed documentation of hyperparameters and model weights, enhances the reliability and reproducibility of our findings, contributing to the advancement of object detection models.