Implementation and Customization of YOLOv7 : state-of-the-art for real-time object detectors.

Rakshith Ram C.A.
Indian Institute of Technology, Hyderabad
sm22mtech12003@iith.ac.in

Abstract

Object detection is one of the fundamental problems of computer vision. YOLOv7 is the current state-of-the-art real time object detectors that outperforms all the existing real-time object detectors as claimed by the authors.

First, the methodology, architecture, the novel additions and the visible improvements in the results are discussed. The further sections deal with the implementation of the code in a local machine. Training, validation and testing is performed on PASCAL VOC 2012 dataset, while extracting mAP and other metrics. Training and inference is also performed on a "Pothole" dataset, which can be used in self-driving cars as a feature.

The novel idea that can be implemented on top of this YOLOv7 model is also discussed in the further parts.

Source code is released by the authors of YOLOv7 in : https://github.com/WongKinYiu/yolov7.

The paper by the original authors can be found here : https://arxiv.org/abs/2207.02696.

1. Introduction

Real-time object detection is a very important topic in computer vision, as it is often a necessary component in computer vision systems. For example, multi-object tracking, autonomous driving, robotics, medical image analysis, etc.

YOLO stands for "You Only Look Once", it is a popular family of real-time object detection algorithms. The original YOLO object detector was first released in 2016. It was created by Joseph Redmon, Ali Farhadi, and Santosh Divvala. At release, this architecture was much faster than other object detectors and became state-of-the-art for real-time computer vision applications.

There are many different object detection models which perform well for certain use cases, but the recent release of YOLOv7, where the researcher claimed that it outperforms all known object detectors in both speed and accuracy and has the highest accuracy 56.8% AP among all known real-time object detectors. YOLOv7-E6 object detector (56 FPS V100, 55.9% AP) outperforms both transformer-based detector SWINL Cascade-Mask R-CNN by 509% in speed and 2% in accuracy, and convolutional based detector ConvNeXt-XL Cascade-Mask R-CNN by 551% in speed and 0.7% AP in accuracy, as well as YOLOv7 outperforms: YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B and many other object detectors in speed and accuracy. Moreover, YOLOv7 is trained only on MS COCO dataset from scratch without using any other datasets or pre-trained weights.

The contributions of the original paper as mentioned by the authors are summarized as follows: (1) we design several trainable bag-of-freebies methods, so that real-time object detection can greatly improve the detection accuracy without increasing the inference cost; (2) for the evolution of object detection methods, we found two new issues, namely how re-parameterized module replaces original module, and how dynamic label assignment strategy deals with assignment to different output layers. In addition, we also propose methods to address the difficulties arising from these issues; (3) we propose "extend" and "compound scaling" methods for the real-time object detector that can effectively utilize parameters and computation; and (4) the method we proposed can effectively reduce about 40% parameters and 50% computation of state-of-the-art real-time object detector, and has faster inference speed and higher detection accuracy.

2. Methodology

2.1 General architecture of YOLO models

The YOLO framework has three main components :

- Backbone
- Head
- Neck

The **Backbone** mainly extracts essential features of an image and feeds them to the Head through Neck. The **Neck** collects feature maps extracted by Extended

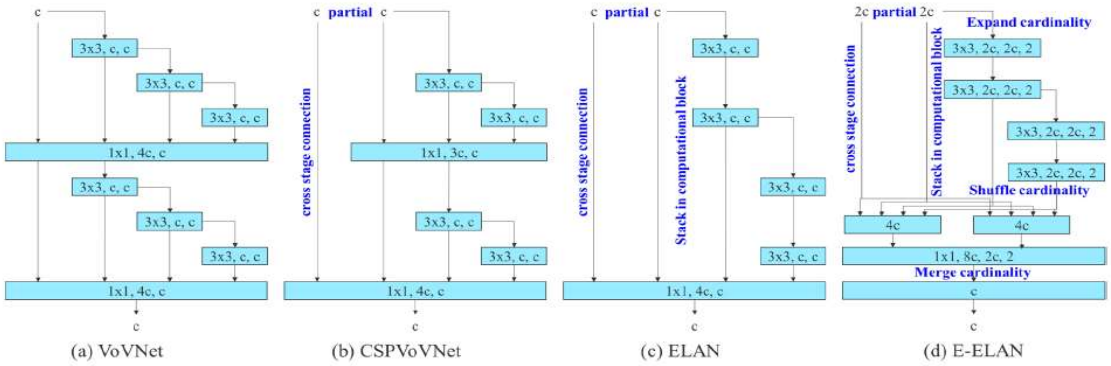


Figure 1: Extended efficient layer aggregation networks (E-ELAN)

the backbone and creates feature pyramids. Finally, the **Head** consists of output layers that have final detections.

2.2 Extended efficient layer aggregation networks

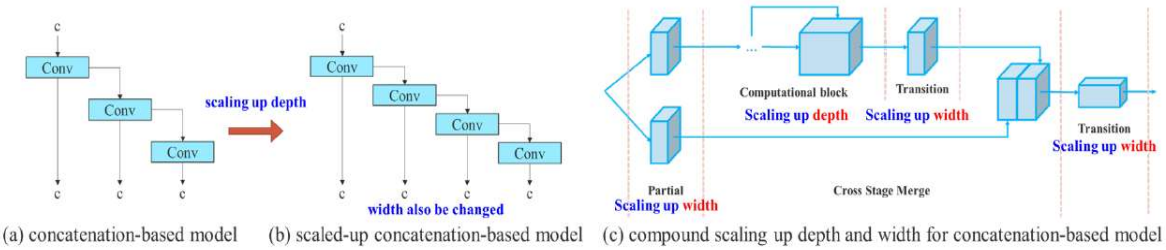
Extended efficient layer aggregation networks primarily focus on a model's number of parameters and computational density. The VovNet (CNN seeks to make DenseNet more efficient by combining all features only once in the last feature map) model and the CSPVoVNet model analyses the influence of the input/output channel ratio and the element-wise operation on the network inference speed. YOLO v7 extended ELAN and called it E-ELAN. The major advantage of ELAN was that by controlling the gradient path, a deeper network can learn and converge more effectively.

E-ELAN majorly changes the architecture in the computational block, and the architecture of the transition layer is entirely unchanged. It uses expand, shuffle, and merge techniques which enhances the learning ability of the network without destroying the original gradient path. The strategy here is to use group convolution to expand the channel and number of computational blocks, which applies the same group parameter and channel multiplier to all the computational blocks of a computational layer. Then, the feature map calculated by each computational block is shuffled and then concatenated together. Hence, the number of channels in each group of the feature maps will be equal to the number of channels in the original architecture. Finally, merge these groups of feature maps. E-ELAN also achieved the capability to learn more diverse features.

2.3 Model scaling for concatenation based models

The main motive of model scaling is to regulate a number of the model attributes and generate models comprised of various scales to increase different inference speeds. However, if these methods are applied to the concatenation-based architecture when cutting down or scaling up is performed in-depth, there will be an increase and decrease within the in-degree of a translation layer which is straight away after a concatenation-based computational block.

We can infer that can't be analyzed different scaling factors separately for a concatenation-based model but must be considered together. Take scaling-up depth as an example, such an action will cause a ratio change between the input channel and output channel of a transition layer, which can cause a decrease in the hardware usage of the model. YOLO v7 compound scaling method can maintain



(a) concatenation-based model (b) scaled-up concatenation-based model (c) compound scaling up depth and width for concatenation-based model

Figure 2: Model scaling for concatenation-based models.

the properties that the model had at the initial design and maintains the optimal structure still, this can be because while scaling the depth factor of a computational block, one should also consider calculating the change of the output channel of that block. Then, perform width factor scaling with the identical amount of change on the transition layers, this maintains the properties that the model had at the initial design and maintains the optimal structure.

2.4 Planned re-parameterized convolution

YOLOv7 proposed planned re-parameterized convolution. In this proposed planned re-parameterized model, authors had found that a layer with residual or concatenation connections, its RepConv should not have an identity connection. Under these circumstances, it can be replaced by RepConvN which contains no identity connections.

RepConv combines 3 × 3 convolutions, 1 × 1 convolution, and identity connection in one convolutional layer. After analyzing the combination and corresponding performance of RepConv and different architectures authors used RepConv without identity connection (RepConvN) to design the architecture of planned re-parameterized convolution. According to the paper, when a convolutional layer with residual or concatenation is replaced by re-parameterized convolution, there should be no identity connections.

2.5 Coarse for auxiliary and fine for lead loss

The Deep supervision technique is often used in training deep neural networks. It adds an extra auxiliary head in the middle layers of the network, and the shallow network weights with assistant loss as the guide. Here in yolov7 authors call the head responsible for the final output the lead head, and the head used to assist in the training is called the auxiliary head.

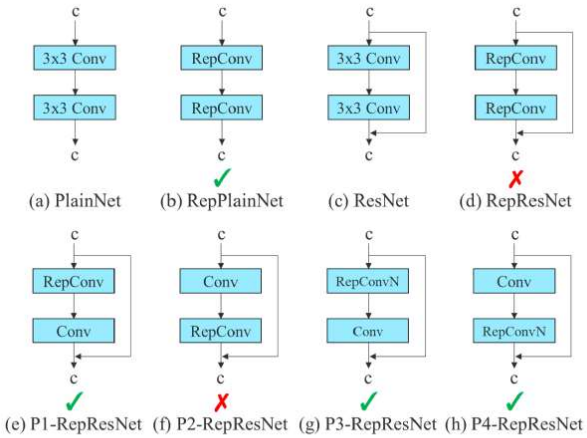


Figure 3: Planned re-parameterized model.

Lead head guided label assigner

It is mainly calculated based on the prediction result of the lead head and the ground truth, which generates a soft label through the optimization process. This set of soft labels will be used as the target training model for both the auxiliary head and lead head. According to the paper, the lead head has a relatively strong learning ability, so the soft label generated from it should be more representative of the distribution and correlation between the source data and the target.

Coarse-to-fine lead head guided label assigner

It uses the predicted result of the lead head and the ground truth to generate a soft label. However, in the process, it generates two different sets of soft labels, that is the coarse label and fine label, where the fine label is the same as the soft label generated by the lead head guided label assigner, and the coarse label is generated by allowing more grids to be treated as a positive target. This mechanism allows the importance of fine and coarse labels to be dynamically adjusted during the training process.

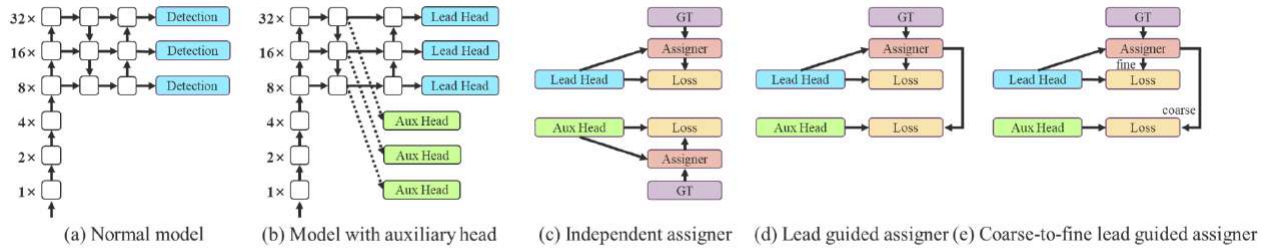


Figure 4: Coarse for auxiliary and fine for lead head label

3. Related work

Currently state-of-the-art real-time object detectors are mainly based on YOLO and FCOS. Being able to become a state-of-the-art real-time object detector usually requires the following characteristics: (1) a faster and stronger network architecture; (2) a more effective feature integration (3) a more accurate detection method (4) a more robust loss function (5) a more efficient label assignment method and (6) a more efficient training method.

In the original paper, the authors do not intend to explore self-supervised learning or knowledge distillation methods that require additional data or large model. Instead, they design a new trainable bag-of-freebies method for the issues derived from the state-of the-art methods associated with (4), (5), and (6) mentioned above.

Speed: YOLO is known for its speed, as it can process images and detect objects in real-time video streams at up to 45 frames per second, making it ideal for applications where speed is critical. In contrast, some other object detection algorithms, such as Faster R-CNN and SSD, are slower and may not be suitable for real-time applications.

Accuracy: While YOLO is known for its speed, it also achieves high accuracy in object detection. However, some PASCAL VOC (Visual Object Classes) is a widely used dataset for object detection, segmentation, and classification in computer vision. The dataset consists of more than 20,000 images, each annotated with object bounding boxes and class labels for 20 object categories, including people, animals, vehicles, and household objects.other object detection algorithms, such as Mask R-CNN and RetinaNet, may achieve higher accuracy on some benchmark datasets.

Single-stage vs. Two-stage: YOLO is a single-stage object detector, meaning it directly predicts the bounding boxes and class probabilities in a single pass through the neural network. In contrast, some other object detection algorithms, such as Faster R-CNN and R-FCN, are two-stage detectors, which involve a region proposal network to generate potential object locations before predicting the class and bounding box.

Overall, YOLO is known for its speed and simplicity, while other object detection algorithms may achieve higher accuracy or use more sophisticated techniques. The choice of algorithm depends on the specific application and the trade-off between speed and accuracy.

3. Implementation and Results

3.1 Dataset used in original paper

MS COCO (Microsoft Common Objects in Context) is a widely used dataset for object detection, segmentation, and captioning tasks. It contains more than 330,000 images, each labeled with object annotations, object segmentation masks, and captions. The dataset includes 80 object categories, such as people, animals, vehicles, and household objects, and covers a wide range of scenarios, including indoor and outdoor scenes.

Figure 5: MS COCO dataset.

The COCO dataset has been used in several challenges, including the COCO detection challenge, segmentation challenge, and captioning challenge, which have helped to push the state-of-the-art in computer vision research. The dataset is publicly available and can be downloaded from the official COCO website, making it a valuable resource for researchers and practitioners working in the field of computer vision.

3.2 Dataset used for Implementation

PASCAL VOC 2012 is the final version of the PASCAL VOC (Visual Object Classes) dataset, which is widely used for object detection, segmentation, and classification tasks in computer vision. The dataset consists of 11,540 images, each annotated with object bounding boxes and class labels for 20 object categories, including people, animals, vehicles, and household objects.

In addition to object annotations, the PASCAL VOC 2012 dataset also includes segmentation masks for some of the images, which provide pixel-level annotations of the object boundaries. The dataset also includes annotations for human pose estimation and action classification.

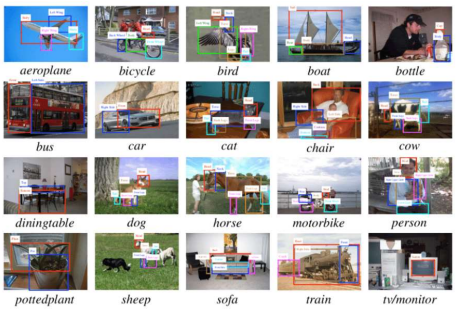


Figure 6: Pascal VOC 2012 dataset.

3.3 Inference by model trained on COCO dataset

The authors have published the weights of the model which is trained on the MS COCO dataset, in their GitHub repository: *https://github.com/WongKinYiu/yolov7*.

Below are some of the steps I have followed to clone the repository, run the model on the local system and get te inference on images, videos and webcam feed.

Downloading all the required codes :-
git clone https://github.com/WongKinYiu/yolov7.git
%cd yolov7

Installing all the requirements :-
pip install -r requirements.txt

Downloading the required weights :-
wget
https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt

Training the model :-
bash scripts/get_coco.sh

python train.py --workers 8 --device 0 --batch-size 32 --data data/coco.yaml --img 640 640 --cfg cfg/training/yolov7.yaml --weights '' --name yolov7 --hyp data/hyp.scratch.p5.yaml

Inference on an images :-
python detect.py --weights yolov7.pt --conf 0.25 --img-size 640 --source inference/images/picture.jpg

Figure 7: Inference on images by model trained on COCO dataset.

Inference on an video :-
python detect.py --weights yolov7.pt --conf 0.25 --img-size 640 --source inference/images/video.mp4

Figure 8: Snapshot of inference on video by model trained on COCO dataset.

Inference on live webcam :-
python detect.py --weights yolov7.pt --conf 0.25 --img-size 640 --source 0

Figure 9: Snapshot of inference on live webcam by model trained on COCO dataset.

Testing the model :-
python test.py --data data/coco.yaml --img 640 --batch 32 --conf 0.001 --iou 0.65 --device 0 --weights yolov7.pt --name yolov7_640_val

3.4 Training and Testing on Pascal VOC 2012 dataset

I have performed the training of the YOLOv7 model on the Pascal VOC 2012 dataset, which is comparatively smaller than the MS COCO dataset, due to constraint of time and computational resources. The training was performed on a local system which had a NVIDIA RTX A5000 Graphics Card. The training was done for 100 epochs and took approximately 7.5 hours. The picture of the same is attached below.

Training the model :-
python train.py --workers 1 --device 0 --batch-size 16 --epochs 100 --img 640 640 –hyp data/hyp.scratch.custom.yaml --name pascalVOC --weights yolov7.pt

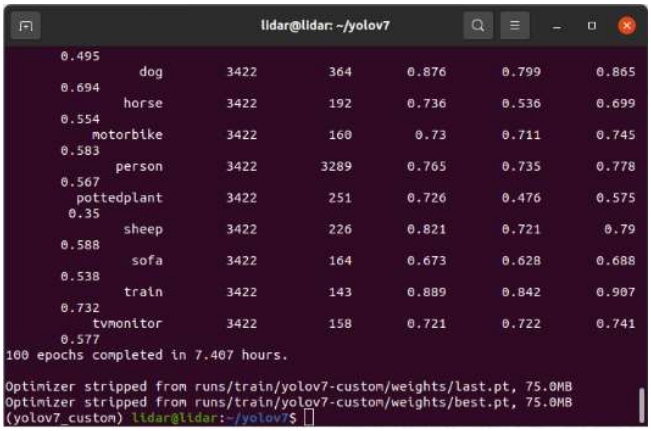


Figure 10: Training the model on Pascal VOC dataset.

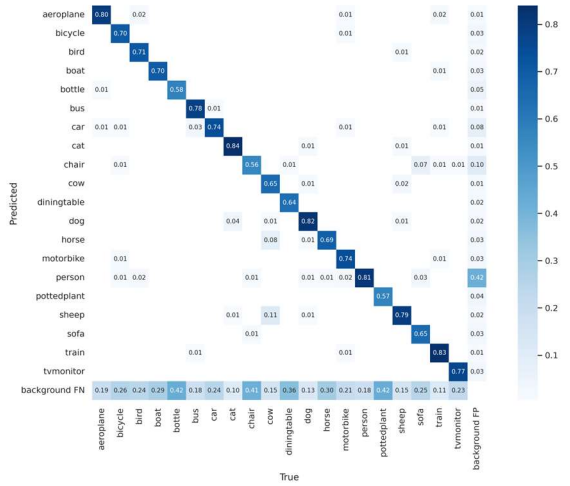


Figure 11: Confusion matrix obtained by testing the model on Pascal VOC dataset.

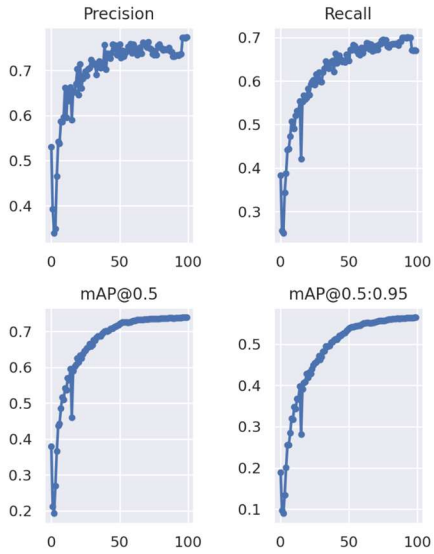


Figure 12: Precision, Recall, map@0.5 and map@0.5:0.95 v/s epochs (100 epochs).

The testing results are as shown in the corresponding images. The Confusion matrix of all 20 classes in the dataset is obtained. The prescision, recall and mAP and F1 score curves are also attached.

As observed from the results, for 100 epochs:

- mAP @ 0.5 CI = 0.741
- mAP @ 0.5 : 0.95 CI = 0.577
- F1 score reaches the highest value of 0.72 at a confidence interval (CI) of 0.432

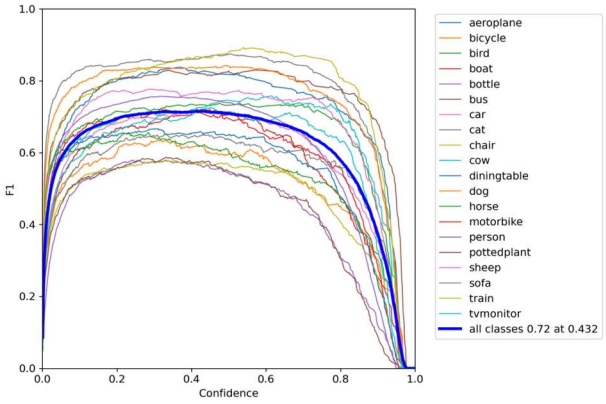


Figure 13: F1 score v/s confidence threshold.

3.5 Training, Testing and Inference on Pothole dataset

Object detection is a critical component of autonomous vehicles, as it allows the vehicle to detect and track other vehicles, pedestrians, cyclists, and other objects in the environment. Object detection is typically performed using cameras, lidar, and radar sensors, which provide different types of data that can be used to detect objects.

Here, in order to customize the detection module for autonomous vehicles, I have trained it on a custom Pothole dataset. This helps in the detection on potholes for the autonomous vehicle which is not a existing class in the MS COCO dataset.

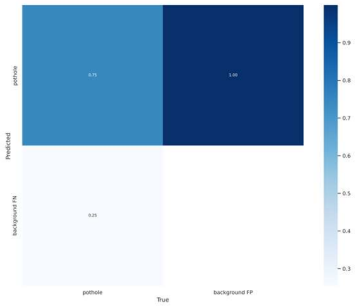


Figure 14: Confusion matrix obtained while testing.

The dataset was a publicly available one downloaded from the Roboflow platform. It is a small dataset containing of 465 training, 133 validation and 67 test images of potholes.

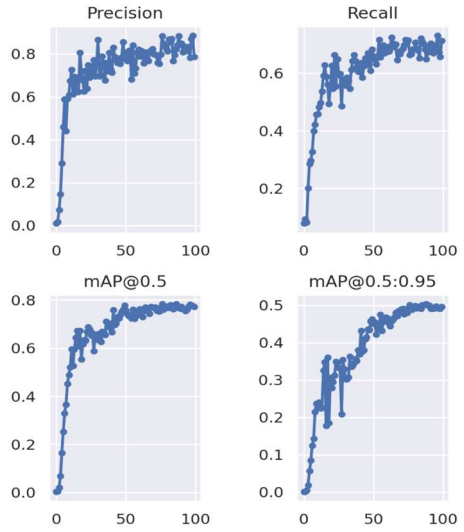


Figure 15: Precision, Recall, map@0.5 and map@0.5:0.95 v/s epochs (100 epochs).

As observed from the results, for 100 epochs:

- mAP @ 0.5 CI = 0.773
- mAP @ 0.5 : 0.95 CI = 0.496

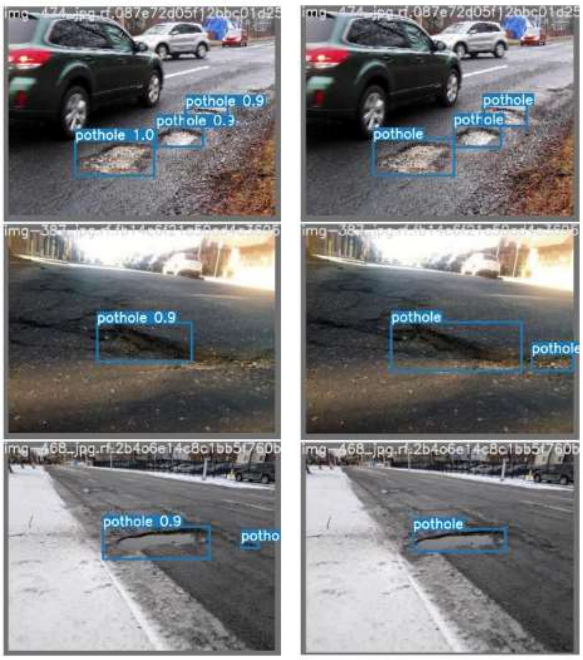


Figure 16: Ground truth v/s Predicted labels and confidence score.

3.6 Comparison with other object detectors

The above graph is a visualization of the performance of our YOLOv7 model when compared with others, as depicted by the original authors.

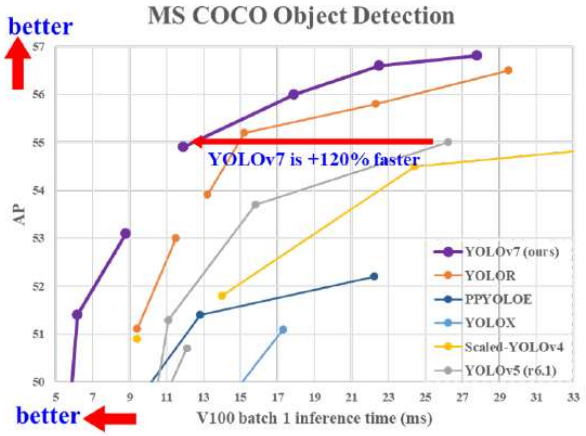


Figure 17: Comparison with other real-time object detectors.

4. Novel idea for future work.

Despite the success of YOLO models in object detection for autonomous vehicles in clear-weather visual scenes, these methods might not be employed directly due to the complex real-world weather conditions.

Domain adaptation refers to the process of adapting a machine learning model that has been trained on one domain to perform well on a different domain. In machine learning, a "domain" refers to the distribution of data that the model has been trained on. Here, for example a model trained in clear weather conditions to detect pedestrians, traffic lights etc. would not perform the same in foggy or rainy weather conditions.

Domain adaptation methods aim to address this problem by adapting the model to the new domain, without requiring additional labeled data from the target domain.

There are a variety of methods which may be used to integrate the domain adaptation methods to improve the performance of the YOLOv7 detection model. Some of them are as follows:

1. Model-based adaptation : This method involves modifying the model architecture or training process to incorporate domain adaptation techniques. For example, using adversarial training to encourage the model to learn domain-invariant features, or using a

domain classifier to identify and mitigate domain-specific features.

2. Ensembling : This method involves training multiple models on different subsets of the data and combining their predictions. Ensemble methods can improve performance in domain adaptation by reducing the impact of individual model biases and increasing model robustness.
3. Self-training : Model is trained on the labelled data from the source domain. It is then used to generate pseudo-labels for the unlabelled data from the target domain. The model is then retrained on the combined labelled and pseudo-labelled data so as to improve the models predicting accuracy.

In order to address the diverse weather conditions encountered in autonomous driving, many datasets have been generated. (References [11, 12, 13, 14]). For example, Foggy Cityscape [14] is a synthetic dataset that applies fog simulation to Cityscape for scene understanding in foggy weather.

Our further goal is to implement domain adaptation using these datasets into the YOLOv7 model using one of the abovementioned methods, so that the detection model can be more accurate in autonomous navigation applications. For example, detecting traffic signs, lights, pedestrians, lanes, potholes, speedbumps etc. even in foggy or rainy weather conditions.

5. Conclusion

In the original paper the authors propose a new architecture of realtime object detector and the corresponding model scaling method. During the research process, they found the replacement problem of re parameterized module and the allocation problem of dynamic label assignment. To solve the problem, they proposed the trainable bag-of-freebies method to enhance the accuracy of object detection. Based on the above, they have developed the YOLOv7 series of object detection systems, which receives the state-of-the-art results.

YOLOv7 is the new state-of-the-art real-time object detection model. We can use it for different industrial applications. Also, we can optimize the model, that is, converting the model to ONNX, TensorRT, etc, which will increase the throughput and run the edge devices.

I have performed the inference of images, videos and webcam stream using the original MS COCO trained weights. I have also performed the training and testing of the model with Pascal VOC 2012 dataset. For Autonomous navigation applications, I have trained the model on Pothole dataset and stated the inferences.

6. References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2016

[2] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2017

[3] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[5] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. NAS-FPN: Learning scalable feature pyramid architecture for object detection. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[6] Jocher Glenn. YOLOv5 release v6.1. https://github.com/ultralytics/yolov5/releases/tag/v6.1, 2022.

[7] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[8] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[9] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. PPYOLOE: An evolved version of YOLO. arXiv preprint arXiv:2203.16250, 2022.

[10] Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection. arXiv preprint arXiv:2107.00057, 2021.

[11] Jinlong Li , Runsheng Xu , Jin Ma , Qin Zou , Jiaqi Ma , Hongkai Yu. Domain Adaptive Object Detection for Autonomous Driving under Foggy Weather. arXiv:2210.15176v1, 2022.

[12] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. arXiv preprint arXiv:1907.07484, 2019.

[13] Quang-Hieu Pham, Pierre Sevestre, Ramanpreet Singh Pahwa, Huijing Zhan, Chun Ho Pang, Yuda Chen, Armin Mustafa, Vijay Chandrasekhar, and Jie Lin. A* 3d dataset: Towards autonomous driving in challenging environments. In IEEE International Conference on Robotics and Automation, pages 2267–2273. IEEE, 2020.

[14] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. International Journal of Computer Vision, 126(9):973–992, 2018.