Performance Analysis of You Only Look Once Object Detection Models on Custom Datasets

Jay Borade[1]*, Muddana A Lakshmi[2]

*[1]Research Scholar at GITAM (Deemed to be University), Hyderabad, India, Email: jaayb84@gmail.com
[2]CSE department, GITAM (Deemed to be University), Hyderabad, India, Email: amuddana@gitam.edu
Corresponding Author: - Jay Borade

Abstract

Object detection, localization, and classification are now thought to be common study areas. The advancement of Deep Learning has aided researchers working in the field of object detection. By decreasing the model's loss functi on, the accuracy of object detection may be boosted. Training and executing this model on a suitable GPUenabled machine can also cut down on total time. In this article, we employed the YOLO family models, namely YOLO v3, YOLO v4, and YOLO v5, and trained them using the transfer learning approach on custom datasets. These cutting-edge performers outperformed the competition on the Custom Dataset of Open Image Dataset V6 for various classes, achieving mean average precision mAP @0.5 97 %, 89 %, and 88 %, respectively. This is better than the initial trained models, which had mAP @0.5 values of 57.9 %, 43.5 %, and 33 % for YOLO v3, YOLO v4, and YOLO v5, respectively.

Keywords: Classification, Custom datasets, Deep learning, Localization, Object detection, Open Image Dataset V6, Transfer Learning, YOLO v1, YOLO v2, YOLO v3, YOLO v4.

DOI Number: 10.14704/nq.2022.20.9.NQ44153 **Neuro Quantology 2022; 20(9):1321-1327**

I. INTRODUCTION

Object Detection is a very broad domain and is a process of identifying, classifying, and localizing an object in each image [1]. Every image feed as an input undergoes through object detection pipeline which consists of phases like Informative region selection, feature extraction and classification. Object detection has its applications in many fields like security, engineering, education, and medical field etc. Object detection technique can be Generic object detection or Salient object detection.

Generic object detection deals with locating and classifying of an object in each image and label them with the bounding boxes. Generic method is further classified into region proposal-based framework and regression /classification-based framework. Salient object detection deals with highlighting most dominant object region in each image. Face detection and pedestrian detection also belongs to generic object detection techniques but belongs to special category.

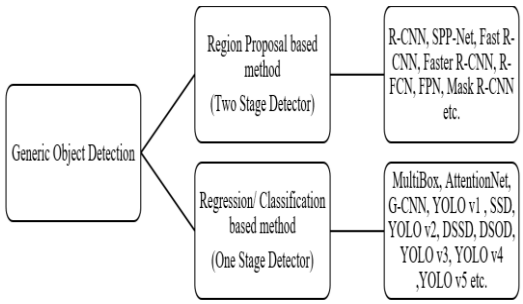

FIG. 1. CLASSIFICATION OF GENERIC OBJECT DETECTION.

Regression/Classification based methods also known as one stage detectors and popular algorithms among these categories are DNN (Deep Neural Network) based segmentation, Regression based MultiBox, Attention Net, Grid based object detector, SSD (Single Shot Detector). DNN based regression [2] generated a binary mask to test the image, it also uses bounding boxes to extract the detected object. Regression based MultiBox [3] creates a scored class-agnostic region proposals /bounding box and to predict the coordinates of these bounding boxes a unified loss is introduced in classification and the confidence of predicting.

Attention Net [4] follows the iterative classification approach to do object detection. It

points to a target object by generating quantified weak directions and it converges to an accurate object bounding box with iterative predictions. G-CNN (Grid-Convolutional Neural Network) Grid Based object detector [5] also follows iterative approach, it uses fixed grid boxes which are tightly surrounded to the objects. It trains a regressors to move and scale elements of the grid towards object iteratively. SSD Single Shot Detector [6] model has advantage over the YOLO v1 (YOLO version 1) that is to detect the small images, it uses VGG16 (Visual Geometry Group 16) as the backbone architecture. SSD model uses anchor concept from MultiBox and RPN (Region Proposal Network) whereas the final detection of the images is done through Non-Max Suppression technique. It is trained on PASCAL VOC and COCO dataset. This paper aims to compare object detection algorithms mainly YOLO v3 (YOLO version 3), YOLO v4 (YOLO version 4) and YOLO v5 (YOLO version 5) on the custom dataset using Google Colab. In section 2 various algorithms of YOLO family are discussed section 3 provides the experimental setups and dataset used. Finally, section 4 tabulates the results obtained and section 5 gives the conclusion.

II. RELATED WORK

Algorithms in Object Detection can be classified as one stage detector and two stage detector. In one stage detector algorithm object localization, detection, classification is done in single run whereas in two stage detector in first stage Region of Interest (ROI) are generated and in the second stage the actual detection is done. We will consider algorithms from YOLO family for one stage detector. The YOLO family comprises of various versions from YOLO v1 to YOLO v5 with some variants including Tiny YOLO, YOLO-Lite and YOLO –Nano etc.

A. YOLO v1

YOLO v1 [7] (You only look once) a unified, real-time object detection algorithm is a single shot object detector method to detect objects and its location from a given image in a single scan. It is faster, and runs at 45 frames per second (FPS), smaller version of same runs at 155 FPS. YOLO works on artworks and has been tested on Picasso and people art datasets. mAP (Mean Average Precision) of YOLO is double with respect to real time detectors like DPM (Deformable Part Model) and R-CNN(Region based Convolutional Neural Network).YOLO is simple, and in a single scan it creates bounding box as well as provides with class probabilities, regression is use to predict various classes of images. YOLO is best for speed but faces localization error and is less efficient to predict small objects in the image like flock of birds. YOLO does not need any complex pipeline architecture like R-CNN[8], also it has 50 % less background error with respect to Fast R-CNN [9].YOLO was tested on PASCAL VOC 2007 and PASCAL VOC 2012 dataset with 20 classes on Titan X GPU.

Training Model: - YOLO is trained on ImageNet dataset with 1000 classes while training 20 convolutional layers followed by an average pooling layer and a fully connected layer was used. Network was trained for a week and single crop top 5 accuracy of 88% achieved on ImageNet 2012 validation set which was comparable to GoogLeNet model with caffe model zoo. After training 4 convolutional layers and 2 fully connected layers with randomly initialized weights are added also the resolution of images were increased from 224 X 224 to 448 X 448. Training was done for 135 epoch on PASCAL VOC 2007 and VOC 2012 dataset.

To avoid overfitting data augmentation and dropout layer with rate=0.5 was added. Data augmentation included random scaling and translation of up to 20% of original images. It evaluates 98 BB (Bouding Box) per image with class probabilities. When YOLO is combined with Fast R-CNN it provides high performance. YOLO in combined YOLO R-CNN model takes care of background Image elimination. YOLO performs best with VOC 2007 with respect to VOC 2012. When YOLO trained on VOC 2007 and applied on artwork it outperforms the other model. R-CNN is highly overfit to PASCAL VOC. YOLO models the shape and size of the object.

B. YOLO v2

YOLO v2 [10] also known as YOLO 9000 as it detects the objects from 9000 different classes. It uses PASCAL VOC & COCO dataset. At 67 FPS YOLO v2 provides 76.8 mAP and on VOC 2007 at 40 FPS YOLO v2 provides 78.6 mAP which is more than Faster R-CNN [11] with ResNet [12] and SSD. Multiscale training facilitates with prediction of small images which was an issue with YOLO, it also works on reducing localization error and improving recall of YOLO. YOLO v2 worked well on real time system and was validated with ImageNet dataset resulting to 19.7

mAP. Training of model used COCO and ImageNet dataset together. YOLO v2 is called to be Better, Faster and Stronger.

1) Better : -

YOLO v2 is said to be Better because of following points:-

- Batch Normalization process improves convergences, eliminates use of regularization method and removes the need of the dropout layer, hence increases mAP by 2% .
- High Resolution Classifier helps to increase the mAP by 4% which is achieved by increasing the resolution from 224 X 224 to 448 X 448.
- Convolutional is done with Anchor Boxes which imporves prediction.
- Dimension Clustering using K-means algorithm with Euclidean distance is to find which anchor box fits well.
- Direct Location Prediction is done through the following equations

$$x = (t_x * w_a) - x_a \quad (1)$$

$$y = (t_y * h_a) - y_a \quad (2)$$

t_x=1 shift box right t_x=-1 shift box left
t_y= 1 shift up t_y= -1 shift down

- Fine Grained Features are created by converting a 26 X 26 X 512 feature map to 13 X 13 X 2048 feature map.
- Multiscale Training provides training on every layer.

2) Faster: -

YOLO v2 uses Darknet 19, it has 19 convolutional layers and 5 max pooling layers. Training is done for both classification and detection with different configurations which makes it a faster model.

3) Stronger:-

YOLO V2 is said to be Stronger because of following reasons

- Hierarchical Classification is achieved through directed graphs of language.
- Dataset form ImageNet and COCO is combined using WordTree.
- Joint Classification & Detection helps YOLO to learns how to identify objects in images using COCO detection data, and how to recognize a wide range of these objects using ImageNet data.

C. Fast YOLO

Fast YOLO [13] is built on YOLO v2 to support devices with limited computational power and memory for real time object detection in video. To reduce the usage of the power motion adaptive inference method is developed also for limited memory consumption 2.8 X (times) fewer parameters are considered with respect to YOLO v2 which effect IOU by nearly 2 % .It reduces the deep neural inference by an average of 38.13% and provides an average speed of nearly 3.3 X for detection with respect to YOLO v2. Fast YOLO runs on an average of 18FPS on Nvidia Jetson TX1 system. It provided two improvements, firstly provided an optimized architecture known as O-YOLO v2 and secondly a motion adaptive inference method.

Table 1. Comparison Of Number Of Parameters And Iou Of Yolo V2 And O-Yolo V2 [13]

Network Architecture	Number of parameters	IOU
YOLO v2	48.2M	67.2%
O-YOLO v2	17.1M	65.10%

Fast YOLO reduces high complexity computation and also high detection error rate which happens when sliding window method for object detection is used.

D. YOLO v3

YOLO v3 [14] is bigger and accurate than YOLO v2 and YOLO. Speed is maintain like earlier version of YOLO. For 320 X320 it runs at 22ms at 28.2 mAP. It is accurate as SSD is and is 3 times faster than SSD. It behaves well at 0.5IOU mAP as compared to RetinaNet.

1) Bounding box prediction

YOLO v2/9000 as well as YOLO v3 uses dimension clusters as anchor boxes. The network predicts 4 coordinates for each bounding box, t_x , t_y , t_w , t_h and the offset from image by (c_x , c_y) and the bounding box prior has width and height p_w , p_h .

Table 2. Comparison Of Yolo V3 And Retina Net On Titan X System.

Criteria	YOLO v3	Retina Net
On Titan X system	57.9 Ap_{50} in 51 ms	57.5 Ap_{50} in 19 ms
Performance	Similar to Retina Net but 3.8 times faster	Similar to YOLO v3 but slower

The predictions are given as

$$b_x = \sigma(t_x) + c_x \quad (3)$$

$$b_y = \sigma(t_x) + c_y \quad (4)$$

$$b_w = p_w e^{t_w} \quad (5)$$

$$b_h = p_h e^{t_h} \quad (6)$$

Where c_x and c_y are the offset of the image for which bounding box are defined by the coordinates correspond to b_x, b_y, b_w, b_h.

In Training of YOLO v3 it uses sum of squared error loss.

$\hat{t}_*$ = Ground truth value

t_* = Predicted value.

To predict objectness score it uses logistic regression; Value =1 if current bounding box overlap with ground truth more than other bounding box Value =0 otherwise. Threshold of 0 .5 is consider for the boxes which overlap somewhat with ground truth and less than .5 are neglected. For class prediction logistic classification is used instead of softmax, also loss during training binary cross entropy loss is considered. Multilabel approach is used which models the data better. YOLO v3 predict boxes at 3 different scales, features are extracted through feature pyramid network final output has a 3D tensor encoding which are Bounding Box, Objectness and Class Prediction.

For YOLO v3 experiment on COCO dataset was done and a tensor of N X N X |3*(4|1|80)| was obtained where 4 is bounding box offset, 1 is objectness prediction and 80 is class prediction. YOLO v2 has 19 convolutional layers and hence Darknet 19. YOLO v3 has 53 convolutional layers and hence Darknet 53.As multiscale predictions are used, even predicting small objects have become easier prior to other versions of YOLO. It performs little worst in case of medium and larger objects. Best part of YOLO v3 but with one issue that if the IOU threshold increases boxes will be not performed as accurate.

E. YOLO-LITE

YOLO –LITE [15] is mainly for portable devices such as cell phones or laptops without GPU. It was trained and tested on PASCAL VOC and COCO dataset and achieved mAP 33.81% on PASCAL VOC with 20 classes and 5000 images for training and 12.26% on COCO with 80 classes and 40,000 training images.. It runs with 21 FPS on a non-GPU computer and 10 FPS on the website. It has 7 layers and 482 floating point operations and is faster 3.8 X times then SSD mobile Net V1.It deals with the problems faced by YOLO and R-CNN models related to Real Time Object Detection on non-GPU platforms.

For Calculation of Average precision

$$average\ precision = \sum_{k=1}^{n} P(K)\Delta r(k) \quad (7)$$

Where P (K) here refers to the precision at threshold k while $\Delta r(k)$ refers to the change in the recall. It mainly focuses on speed and not overall physical size of the network and weights. YOLO LITE uses Tiny YOLO v2 as the base structure and after performing many R & D's the final model is developed. Darknet is used to train and test the model. Training was done on Alienware Aura R7 with Intel i7 CPU and NVidia 1070 GPU. Testing per frame was done on an XPS 13 laptop using Darkflow's live demo script. Processes like Batch Normalization and Pruning was not used.

F. YOLO-Nano

One of the biggest challenges for object detection algorithms is the deployment. When it comes to lack of space and memory utilization, object detection algorithms need proper hardware and software setup for providing accurate and efficient results. YOLO Nano [16] is the best for the mobile device/devices with less power and memory resources it provides a better efficiency with respect to Tiny YOLO v2 and Tiny YOLO v3. In earlier models developers were mainly interested in getting better performance, accuracy and speed, but in various applications a light weighted models are required high processing power and heavy models provides good results but for this heavy hardware structure is necessary and hence in order to operate on light weighted devices such as mobiles etc. light weighted models are needed.

[7] introduced Tiny YOLO family or network architecture to face this challenge. YOLO Nano consists of two major design phases: -

1) Stage 1

- ***Principled Network design prototype***

It is an initial prototype of network architecture, based on concepts of human-driven architecture. The concept is architecturally based on SSD and follows the YOLO theory. Several convolutionary layer with a 3 Dimensional tensor output consisting of Bounding Box, Objectness, and Class Prediction follow the feature representation modules in YOLO Nano.

2) Stage 2

- ***Machine Driven Design exploration***

It specifies the macroarchitecture level of the module and the microarchitecture design for the architecture of the YOLO Nano network. Via generative synthesis, certain architectural designs are achieved.

- ***YOLO Nano architectural Design***

It has unique residual projection expansion-projection (PEP) macroarchitecture, which consists of 2 Projection layer (at start and end), Expansion Layer, Depth wire convolution layer. Fully Connected Attetion macroarchitecture consists of two completely connected layers, which learn the dynamic interdependencies between channels and produce modulation weights. Macroarchitecture and Microarchitecture heterogenity allows for the unique tailoring of each part of the network architecture to achieve a very good balance between architectural and computational complexity and model expressions. When different Power supply was considered YOLO Nano worked as below:-

Table 3 Comparison Power, Inference Speed And Power Efficiency On Yolo Nano Model.

Power	Inference Speed	Power Efficiency
15W	~ 26.9 FPS	~ 1.97 images/sec/watt
30W	~48.2 FPS	~ 1.61 images/sec/watt

G. YOLO v4

YOLO v4 [17] is the latest and improvised version of YOLO family, it considers various parameters for perfection of model which includes Weighted Residual Connections (WRC), Cross Stage Partial Connections (CSP), Cross mini Batch Normalization (CmBN), DropBlock Regularization, CIoU loss etc. and achieves a 43.5% AP (65% AP_{50}) on MS COCO dataset at the real time speed of 75 FPS on Tesla V100 a conventional GPU. It addresses the problems faced by CNN model in Real Time with Conventional GPU.

The architecture of YOLO v4 is divided into three parts that is Backbone, Neck and Head. Backbone consists of CSPDarknet 53[18] model. PAN [19] and SPP [20] is used at the Neck and YOLO V3 model is present at Head portion of the architecture. YOLO v4 authors implemented methods for object detection at the backbone and detector for Bag of Freebies (BOF) and Bag of Specials (BOS). Backbone BOF has CutMix [21] and Mosaic data increase, DropBlock control, Class mark smoothing and also BOS is composed of Mish activation, Cross stage partial connections(CSP), Multi input weighted residual connections(MiWRC).Detector BOF has CIOU-loss, CmBN, DropBlock regularization, Mosaic data increase, self-adversarial preparation, eliminating grid sensitivity, using multiple anchors for a single ground reality, Cosine annealing scheduler, optimal hyperparameters, Random preparation shapes while BOS consists of mish activation, SPP block, SAM-block, PAN, path aggregation block, DIOUS-NMS. Dataset used for validation was ImageNet & MS-COCO for testing. Mixup, CutMix, Mosaic, Bluring data increase and mark smoothing regularization methods were used in BOF while the activation function LReLU, Swish and Mish was used in BOS.

H. YOLO v5

YOLO v5 [22] is a latest development in the YOLO family though by different author. It was developed in May 2020 soon after the development of the YOLO v4. YOLO v5 uses PyTorch framework and hence less complex and light weighted. It has shares similar architecture as YOLO v4 and also provides similar results with respect to accuracy when trained on COCO dataset. YOLO v5 provides high speed as overall training time and inference time utilized by it is less with respect to YOLO v4. YOLO v5 is available in various configuration like YOLO v5s, YOLO v5m, YOLO v5l, YOLO v5x where (s, m, l, x) provides the size of various models.

III. EXPERIMENTAL SETUPS

We perform three different experiment with YOLO v3, YOLO v4 and YOLO v5 on Google Colab; dataset used and the experimental setup used is explained below:-

A. Dataset

Dataset was collected from V6 open Image Dataset with three classes viz, Apple, Horse and Trousers .1000 image per class was downloaded and then various augmentation like, rotation, flip, crop, exposure were applied and dataset of 2400 images per class was created. Total 7200 images were consider in complete dataset.

B. YOLO v3 using DARKNET

In order to train the YOLO v3 on the custom dataset firstly CuDNN was configured and Darknet was cloned through github. The weights from the YOLO v3 pre-trained model from pjreddie's github was used after uploading the dataset on the Google Colab the model was trained for about 8 hours and 2500 epochs to obtained mAP of 97% with average loss 0.001 and inference time of 0.098 sec.

C. YOLO v4 using DARKNET

Firstly CuDNN was configured on Google Colab at the runtime the obtained GPU was Telsa P100-PCIE. After configuring Darknet framework was installed by cloning it from github. Weights from pre-trained model of YOLO v4 was obtained through AlexyAB's github repository. Custom dataset was then uploaded on Google Colab which includes Train Test and validate folders containing Images and their annotations. Then in the configuration file max batches were made to 6000 as the classes was 3 and learning rate used was 0.001.Model was then ready to train on the custom dataset. Training took 4hours 48min 12sec of time for 1900 epochs and mAP @0.5 obtained was 89.41% with average loss of 0.000010. Inference time for testing was nearly 0.021sec.

D. YOLO v5 using PyTorch

Firstly YOLO v5 code from ultralytics github was clone on the Google Colab and then the dependencies were installed. GPU obtained during the training was Telsa P100-PCIE. Custom dataset was uploaded and YOLO v5s model was used for training. After training for 43 min and 27 sec for 200 epochs mAP @0.5 obtained was 88% with average loss of 0.102 and Inference time for testing was 0.076 sec.

IV. RESULTS

Google Collab has the limitation on the usage of free GPU where the data can be stored on the cloud only for 12 hours and hence accordingly the experiments were designed. mAP is the commonly used measure for evaluating the performance others are available too which are mentioned in [13] We also evaluated mAP and the obtained results are tabulated below: -

Table 4. Results Obtained By Re –Training The Original Models On Custom Dataset.

Models	Framework	Dataset size	Image size	No of epochs	Training Time	Inference Time	mAP @0.5	Avg Loss
YOLO v3	Darknet	7200	416 X 416	2500	08 hours	0.098 sec	97	0.001
YOLO v4	Darknet	7200	416 X 416	1500	4hr 48min 12sec	0.021sec	89	0.00001
YOLO v5s	Pytorch	7200	416 X 416	200	43min 27 sec	0.076 sec	88	0.102

Table 5. Comparison Of Map @0.5 From Experiment And From The Original Models.

Models	Framework	mAP @0.5 from Original Models	mAP @0.5 from Experimental results
YOLO v3	Darknet	57.9	97
YOLO v4	Darknet	43.5	89
YOLO v5s	Pytorch	~33[14]	88

V. CONCLUSIONS

We retrained YOLO models on the Colab notebooks and got good results because number of classes and images used were quite less compared to those used for the actual models were better than the actual model. Dataset we've taken from Open Image dataset that provides lots of images with proper annotations regarding object identification, bounding box, classification etc. We increased the data set images by providing transformation such as turn, rotation, crop, and exposure due to which model was robustly trained. We need proper dataset to provide a generalised model with more class numbers and better accuracy. Model

performance deviates depending on the data set preparation.

REFERENCES

Z. Zhao, P. Zheng, S. Xu and X. Wu. (2019) 'Object Detection with Deep Learning: A Review, IEEE Transactions on Neural Networks and Learning Systems', Vol.30 No. 11, pp. 3212-3232.

C.Szegedy, A. Toshev and D. Erhan. (2013) 'Deep Neural Networks for Object Detection, Advances in Neural Information Processing Systems 26', Lake Tahoe, Nevada, USA, pp. 2555-2563.

D.Erhan, C. Szegedy, A. Toshev and D'Angelo. (2014) 'Scalable Object Detection using deep neural networks', IEEE Conference on Computer Vision and Pattern Recognition, Columbus, Ohio, pp.2155-2162.

D. Yoo, S. Park, J.-Y. Lee, A.S. Paek, and I. So Kweon. (2015) 'AttentionNet: Aggregating weak directions for accurate object detection', IEEE International Conference on Computer Vision, pp. 2659-2667.

M. Najibi, M. Rastegari and L.S. Davis. (2016) 'G-CNN: An iterative grid-based object detector', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2369-2377.

W. Liu, D. Anguelov, D. Erhan, C. szegedy, S. Reed, C.-Y. Fu and A.C. Berg. (2016) 'SSD: Single Shot MultiBox Detector', European Conference on Computer Vision, pp. 21-37.

J. Redmon, S. Divvala, R. Girshick and A. Farhadi. (2016) 'YOLO You Only Look Once – Unified Real-Time Object Detection', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.779-788.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. (2014) 'R-CNN: Rich feature hierarchies for accurate object detection and semantic segmentation', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.580-587.

R. Girshick. (2015) 'Fast R-CNN', IEEE International Conference on Computer Vision (ICCV), pp.1440-1448.

J. Redmon and A. Farhadi. (2017) 'Yolo9000: Better, faster, stronger', Computer Vision and Pattern Recognition (CVPR), pp. 6517–6525.

S. Ren, K. He, R. Girshick, and J. Sun. (2017) 'Faster R-CNN', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.39 No.6, pp.1137 – 1149.

K. He, X. Zhang, S. Ren, and J. Sun. (2016) 'Deep residual learning for image recognition', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778.

M.Shafiee, B. Chywl, F. Li and A. Wong. (2017) 'Fast YOLO: A Fast You Only Look Once System for Real-Time Embedded Object Detection in Video', Journal of Computational Vision, and Imaging systems, Vol.3 No.1, pp.141-143.

J. Redmon and A. Farhadi (2018). 'YOLO V3: An incremental improvement', arXiv preprint arXiv:1804.02767.

R. Huang, J. Pedoeem and C. Chen. (2018) 'YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers', IEEE International Conference on Big Data, Seattle, WA, pp. 2503 – 2510.

A. Wong, M. Famuori, M. Shafiee, F. Li, B. Chywl and J. Chung. (2019) 'YOLO Nano: A Highly Compact You Only Look Once Convolutional Neural Network for Object Detection', arXiv preprint arXiv:1910.01271.

A. Bochkovskiy, Y. Wang and Y. Liao. (2020) 'YOLO V4 Optimal Speed and Accuracy of Object Detection' , arXiv preprint arXiv: 2004.10934.

C. Wang, H. Liao, Y. Wu, P. Chen, J, Hsieh, and I. Yeh. (2020) 'CSPNet: A new backbone that can enhance learning capability of CNN'. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1571-1580.

S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. (2018) 'Path aggregation network for instance segmentation', IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8759–8768.

K. He, X. Zhang, S. Ren, and J. Sun. (2015) 'Spatial pyramid pooling in deep convolutional networks for visual recognition', IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol. 37 No. 9 , pp. 1904–1916.

S. Yun, D. Han, S. Joon, S. Chun, J. Choe, and Y. Yoo. (2019) 'Cut Mix: Regularization strategy to train strong classifiers with localizable features', IEEE International Conference on Computer Vision (ICCV), pp. 6023–6032.

Only the model [online] https://github.com/ultralytics/yolov5. (Accessed 23 April 2021).