Pothole Detection from Dash Camera Images using YOLOv5

Ashit Patel
Dept. of Computer Science
Loyola Marymount University
Los Angeles, USA
apatel33@lion.lmu.edu

Lei Huang
Dept. of Electrical and Computer Engineering
Loyola Marymount University
Los Angeles, USA
lei.huang@lmu.edu

Gustavo Vejerano
Dept. of Electrical and Computer Engineering
Loyola Marymount University
Los Angeles, USA
gustavo.vejarano@lmu.edu

Abstract

In this paper, we propose a new solution to automatically detect potholes on the road surface from dash camera images using a state-of-the-art deep learning based object detection algorithm, namely, You Only Look Once version 5 (YOLOv5). The dash camera image data were preprocessed and augmented as inputs to Convolutional Neural Network (CNN) models, which are trained to output the detected potholes with location bounding boxes. Through transferred learning, different sizes of CNN models with different layer architectures are evaluated in terms of mean Average Precision (mAP) and the number of frames per second (fps). Compared with previous work, experimental results show that our proposed solution using YOLOv5 achieved higher detection accuracy at faster detection speeds, while enabling tradeoffs between accuracy and speed with three different model size options.

Index Terms — YOLO, Object Detection, Localization, Potholes, Deep Learning, Convolutional Neural Network (CNN), Autonomous Vehicles

1. Introduction

Potholes are common road damages of varying sizes and shapes. They are usually formed by the expansion and contraction of ground water once the water has entered the ground under the pavement, and expedited by certain weather and traffic conditions. For example, potholes may sprout after rain in spring when the temperature fluctuates frequently. Potholes can be dangerous, resulting in road accidents and vehicle damage. In the United States, it is estimated that potholes account for about 3 billion dollars in car damages [12] each year. Severe accidents or damage can happen when drivers attempt or fail to avoid potholes, especially for stressed and fatigued drivers. In response, car manufacturers are continuously working on improving automated driving assistance where safety is the utmost priority. This requires detection of road conditions, so the vehicle can make autonomous decisions for safety measures, and automatic pothole detection plays an important role.

Moreover, potholes without timely treatment would accelerate further damage to the road, resulting in a higher cost of road maintenance. Timely detection and treatment of potholes have always been a priority of road service agencies.

Traditional road maintenance relies on either scheduled road surveillance or reporting calls from drivers to detect potholes. Scheduled road surveillance cannot respond to newly-formed potholes promptly. The operation consists of field data collection, identification, and classification. Currently, experienced and well-trained personnel are required to perform these tasks, resulting in high costs in time and labor. The delay could be in months or even years depending on the frequency of the schedule. On the other hand, responding to calls from drivers can be faster, but these calls are usually triggered after damage to callers' vehicles. Therefore, small potholes or those deviating from the center of driving lanes are not reported in time unless vehicle damage takes place. In addition, the manual nature of the reporting process results in inaccurate information, adding to the delay and cost.

Accurate pothole detection from autonomous vehicles will enable early detection and reporting through crowdsourcing and Internet-of-Things platforms, which could lead to a revolutionary change in road maintenance.

The pothole detection problem has been addressed with different approaches including 3D scene reconstruction, vibration-based models, and 2D image-based models [4]. Low-cost cameras and advanced image processing have inspired the development of 2D image-based models using deep learning. These models employ learning-based object detection algorithms, including You Only Look Once (YOLO) [13], Single Shot Detection (SSD) [4], and Region-based Convolutional Neural Networks (R-CNNs) [4].

Results of existing models for pothole detection in [4, 5, 6, 7, 8] indicate that YOLO solves the problem effectively in terms of detection speed and accuracy. The main challenges for reliable pothole detection in 2-D images are the various shapes and sizes of potholes. Moreover, false positives increase when there are objects similar to potholes such as patches, shadows, and water. As a result, the improvement of accuracy usually comes at the cost of computational complexity and detection time. This paper investigates a newer version of YOLOv5, proposed in [1], which achieves better precision and considerably better speed than previous solutions for pothole detection.

This paper is organized as follows. The related work is presented in Section II. Section III describes the components of our work including datasets, data preprocessing, CNN model architecture, and evaluation metrics. Section IV presents our experimental results in comparison to related work. The paper is concluded in Section V.

2. Related work

In [4], one of the solutions that addresses the problem of pothole detection uses transfer learning with Mask R-CNN, one of the region-based convolutional neural networks[14], and the backbone CNN models of ResNet101 [15] and FPN [16]. Mask R-CNN is composed of a two-step framework. The first step scans the whole image to generate proposals. The second step classifies the proposals and generates bounding boxes and masks of an object. For transfer learning, the weights were adapted from the trained model of the COCO dataset [17] and performed the training for fine-tuning with new data. The drawback of this solution is that R-CNN based models have longer prediction times. Experiments were carried out on a combination of multiple datasets like CCSAD, DLR, Japan, Sunny, and PNW. this method gave 89% precision and 93% recall.

Another approach in [4] uses transfer learning with YOLOv2 [9]. It is based on a regression algorithm that predicts classes and bounding boxes for the whole image. In YOLO, a single CNN is used for both classification and localization of an object. The YOLOv2 architecture is based on 22 convolutional layers and 5 Maxpool layers. The input image is divided into a grid cell to find the object of interest, whose center falls into a particular cell. It first generates bounding boxes with confidence scores, then followed by Non-Max Suppression (NMS), which is a process of removing bounding boxes with low object probability and large shared area. Using the same databases mentioned in the previous experiment, this method gave an average of 0.69 Intersection over Union (IoU).

A solution for the detection of potholes using YOLOv3 [10] was proposed in [5]. It estimates how much a bounding box resembles the object of interest based on logistic regression. Logistic classifiers are used since the softmax layer did not prove to be of much use for boosting performance. The performance of YOLOv3 [10] in detecting small objects has improved by several folds, but the performance is not as strong and promising as the results from YOLOv2 [9] when it comes to large and medium-sized objects. In YOLOv3 [10], 53 convolution layers are used. Three different scales are used for predicting bounding boxes. For every bounding box, an objectness score is calculated. The class label of objects in the bounding boxes is calculated using multilabel classification. The final layer produces a 3D tensor with the bounding boxes, objectness, and the class prediction encoded in it. The dataset consisted of images taken manually from a car camera using a phone and consisted of around 1500 images. A 0.49 mAP was achieved at a threshold of 0.5 IoU.

Another solution to the pothole-detection problem using YOLOv4 [11] was proposed in [6]. It aims at building a faster object detector for production systems with optimized parallel computations. YOLOv4 [11] boosts the performance over YOLOv3 [10] by using strategies such as Bag of Freebies and Bag of Specials. Bag of Specials causes a marginal overhead on the time required in the detection phase whereas Bag of Freebies improves the performance without additional time overhead. One such strategy in the Bag of Freebies employed in YOLOv4 [11] is the Complete-IoU (CIoU) loss [18], which is a loss function that considers the overlapping area, the distance between center points, and the aspect ratio, thereby achieving better convergence speed and accuracy. A strategy called Distance IoU Non-Max Suppression (DIoU NMS) [19] is used in Bag of Specials. DIoU considers IoU and an object's distance from the center, while NMS filters out the bounding boxes that improperly predict the same object and retains the one with the highest score. The CIoU and DIoU loss used in YOLOv4 [11] help attain significant performance improvement in terms of IoU. This approach gave 0.93 mAP on the dataset created manually, which is a better result when compared to previous versions of YOLO [9, 10].

The solution proposed in [8] also used YOLOv3 [10]. In this study, three architectures of YOLOv3 [10], i.e., YOLOv3, YOLOv3-tiny, and YOLOv3-spp were used. The YOLOv3 model [10] has a feature extractor of 53 layers. YOLOv3-tiny is a smaller version of YOLOv3 whileYOLOv3-spp is YOLOv3 [11] combined with Spatial Pyramid Pooling (SPP). With a dataset built from a CCTV camera mounted on the back of a car, this proposed solution gave 0.95 mAP.

The solution proposed in [7] also uses YOLOv3 [10] to solve the pothole-detection problem. The training of this model is done with full images and the probability of the class in the bounding boxes. This method has more benefits than the original methods for object detection. The YOLOv3 model [10] is fast. 45 fps can be run, and on a faster version, 150 fps can be achieved. This implies that real-time video can also be processed with latency as small as 8ms. Using 1500 images, a 82% accuracy was achieved in this experiment.

The solution proposed in this paper makes improvements over the above-mentioned approaches by using a newer YOLO version, YOLOv5 [1]. It is more efficient in terms of prediction, training time, and fps. Our YOLOv5 model is trained with the dataset from [2, 3]. This dataset is augmented using image preprocessing methods including horizontal flip and brightness/contrast adjustment.

3. Proposed solution

Our proposed solution aims to detect potholes in real-time from the images of a dash camera with high detection accuracy and fast detection speed that satisfy safety requirements for making autonomous decisions. In the following subsections, we will describe the image dataset format and data augmentation methods used, the architecture of CNN models trained, and performance metrics evaluated in our proposed solutions.

3.1. Dataset

We used the dataset compiled at the Electrical and Electronic Department, Stelllenbosch University, 2015, provided in [2] and [3]. This is the same dataset used for the solution proposed in [7]. It consists of images taken from a video of a car dash camera. The image resolution is 3680 x 2760 pixels. An example of the image from this dataset is shown in Fig. 1.

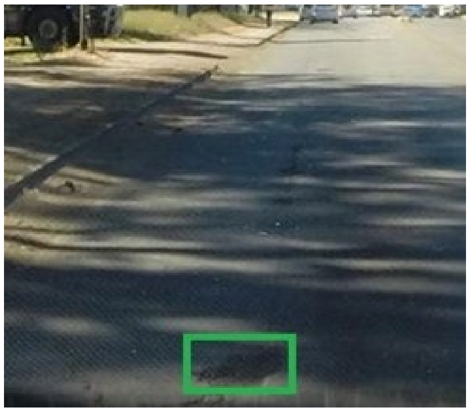

Fig. 1: Example of training data from source [2, 3]

Each image in this dataset is labeled with 5 elements per detection object: class, box center (x, y), width, and height. For example, one entry would be 0, 0.59, 0.79, 0.26, and 0.39. Here, 0 is the class label for potholes. The class label is followed by the center coordinates (x, y) of the bounding box: 0.59 and 0.79, as fractions of the width and height of the image. Finally, the width and height of the bounding box are 0.26 and 0.39 respectively, also as fractions of the width and height of the image.

3.2. Data preprocessing

The preprocessing of images from the dataset consisted of cropping and resizing.

The images were cropped for two reasons. The first reason is that potholes are not present in the bottom or top areas. These areas cover the sky and the vehicle's dashboard. The second reason is that cropping images increases the size of potholes relative to the image size. Cropping at the bottom was performed as follows. Among all images, the pothole that was closest to the bottom was identified. All images' bottom was cropped so that this pothole was 100 pixels from the new bottom. A similar process was performed to crop the top of the images. As a result, the new vertical size of the images was reduced to 37% of the original height. The left and right areas of all images were then cropped reducing the horizontal size of images to 37% of its original value to maintain a constant aspect ratio. It needs to be noted that cropping the images horizontally did remove some potholes labeled close to the left and right boarders, but the number of these potholes is smaller in comparison to the total number of potholes in the dataset. Images were then resized to 640 x 640 for training to fulfill the input requirements of the models. The labels of potholes were also adjusted accordingly.

For each training batch, we pass training data through a data loader with data augmentation. The data loader makes a variety of augmentations, such as scaling, color space adjustments, horizontal flipping, and mosaic image generation, which combines four images into four tiles of a new image with random ratios.

There were 7779 images in the augmented dataset, and the dataset was randomly split into training, validation, and testing sets with a 70:20:10 ratio.

3.3. Model architectures

The YOLOv5 model architecture is shown in Fig. 2.[1]. It consists of three main parts: model backbone, model neck, and model head.

1) Model backbone: The model backbone is mainly used to extract important features from the given input image. In YOLOv5, the Cross Stage Partial Networks (CSP) [22] are used as the backbone to extract informative features from an input image. The CSP addresses duplicate gradient problems in other larger Convolutional Network backbones resulting in fewer parameters and fewer FLOPS for comparable importance. This is important to the YOLO family, where inference speed and small model size are important.

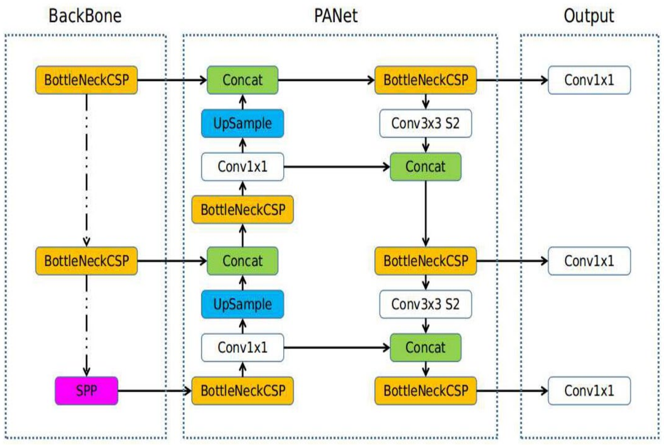

Fig. 2: YOLOv5 CNN model architecture [1]

The CSP models are based on DenseNet [23] with connected layers in convolutional neural networks. Because it is hard to back-propagate loss signals through a very deep network, the DenseNet was designed to alleviate the vanishing gradient problem, to bolster feature propagation, to encourage the network to reuse features, and to reduce the number of network parameters.

DenseNet was also edited in CSPResNext50 and CSPDarknet53 to separate the feature map of the base layer by copying it and sending one copy through the dense block and sending another straight on to the next stage. The idea with CSPResNext50 and CSPDarknet53 is to remove computational bottlenecks in the DenseNet [23] and improve learning by passing on an unedited version of the feature map.

2) Model neck: It is mainly used to generate feature pyramids. Feature pyramids help models generalize well on object scaling. It helps to identify the same object with different sizes and scales such that the model would perform well on unseen data. There are different types of feature pyramid techniques such as FPN [24], BiFPN [25], PANet [26]. In YOLOv5, PANet [26] is used to get feature pyramids.

3) Model head: It is mainly used to perform the final detection. It applies anchor boxes on features and generates final output vectors with class probabilities, objectness scores, and bounding boxes. In YOLOv5, the model head is the same as in the previous versions, YOLOv3 and YOLOv4.

3.4. Evaluation Metrics

We evaluate the performance of our proposed solution in terms of detection accuracy and detection speed. The detection accuracy is measured by mean Average Precision (mAP), while the detection speed is measured by the number of image frames per second (fps) processed during testing.

Object detection accuracy can be evaluated using binary detection performance metrics, such as precision and recall, under a certain threshold of IoU, which is a metric of localization accuracy. The mean Average Precision (mAP) is a single average precision number that combines precisions and recalls under different IoU thresholds. Generally, the higher the mAP value, the more accurate the detection is. In our experiment, mAP is bounded by the range of [0,1].

A real-time detection should achieve at least 60 fps when the trained model is deployed. With higher fps, i.e., faster detection, safer autonomous control can be implemented.

4. Experimental results

The experiments were carried out on an Nvidia RTX 3070 graphical processing unit to boost training speed performance using the CUDA library. The experiments included training the model on small, medium, and large architectures of YOLOv5 [1]. Each training was done for 500 epochs using pre-trained weights of YOLOv5 [1], which were trained on the COCO dataset [17]. The mAP was determined for each epoch, and the average mAP was recorded for different IoU threshold values: from 0.5 to 0.95 with increments of 0.05.

The initial learning rate was set to 0.005. Validation result after each epoch during training was noted for all 3 models (i.e., small, medium, and large architectures of YOLOv5).

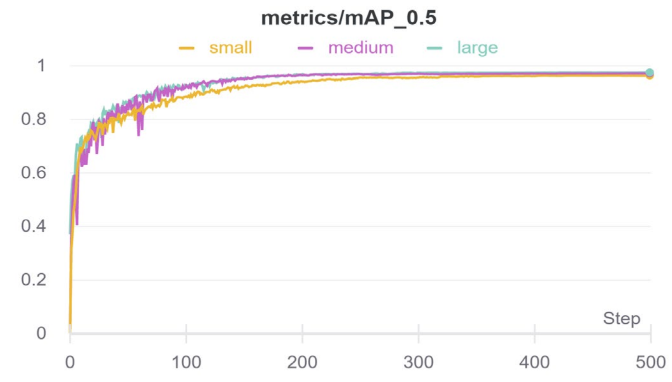

Fig. 3: Training validation results of mAP at 0.5 IoU

As shown in Fig. 3, the mAP at the 0.5 IoU threshold flattened after 250 to 300 epochs. On the other hand, the average mAP across the 0.5 to 0.95 IoU threshold values flattened after 475 to 500 epochs as shown in Fig. 4. Therefore, 500 epochs were considered a good stopping point. The training time was 4-6 hours for the small architecture, 10-12 hours for the medium architecture, and 22-24 hours for the large architecture

.

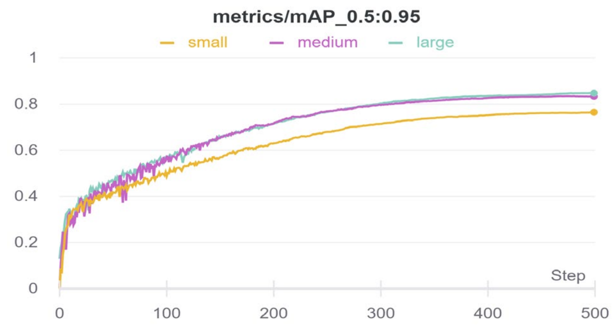

Fig. 4: Training validation results of mAP at 0.5 IoU to 0.95 IoU with increment of 0.05.

The final validation results at the end of the training are shown in Table I. It can be seen that the average mAP across the 0.5 to 0.95 IoU threshold values is less than the mAP at the 0.5 IoU threshold value for all three models. The small architecture has comparatively less mAP than the medium and large architectures, while the medium architecture has an mAP value similar to that of the larger architecture. Observing the trend of a large increase in training time and a small increase in mAP with a larger model size, we did not use the extra-large model that was also proposed in [1].

TABLE I. TRAINING VALIDATION RESULTS

Model Size	***mAP @ 0.5***	***mAP @ 0.5:0.95***	***Precision***	***Recall***
Small	0.9636	0.7648	0.9633	0.9387
Medium	0.9716	0.8332	0.982	0.9445
Large	0.9753	0.8471	0.9746	0.956

Our test results are shown and compared with those of some previous works in Table II. From this table, we can see that the small, medium and large YOLOv5 architectures delivered mAP values of 0.934, 0.933, and 0.937 respectively for an IoU threshold value of 0.5. The related works that used YOLOv3 and YOLOv4 delivered mAP values of 0.869 and 0.933. Although the improvements in mAP from YOLOv4 do not seem to be substantial, there is a significant improvement in the detection speed when comparing YOLOv5 to previous versions. YOLOv4 delivers around 60-100 fps [11] while YOLOv5 delivers 200 to 400 fps depending upon the size of the model used [1].

TABLE II. TEST RESULTS

Model	***Test set size***	***mAP @ 0.5***	***mAP @ 0.5:0.9***	***Prec***	***Rec***	***Speed (FPS)***
Small	778	0.934	0.726	0.942	0.909	400-500
Medium	778	0.933	0.778	0.951	0.912	300-400
Large	778	0.937	0.791	0.967	0.909	200-300
R-CNN [4]	6	-	-	0.898	0.928	-
YOLOv3 [5]	300	0.4971	-	0.76	0.4	-
YOLOv4 [6]	130-260	0.933	-	-	-	60-100
YOLOv3 [8]	224	0.8693	-	0.92	0.82	-

Note that we used significantly more test images (778) than the related works, which could have played an important role in achieving higher mAP values. Fig. 5 shows examples of test images where a pothole was successfully detected.

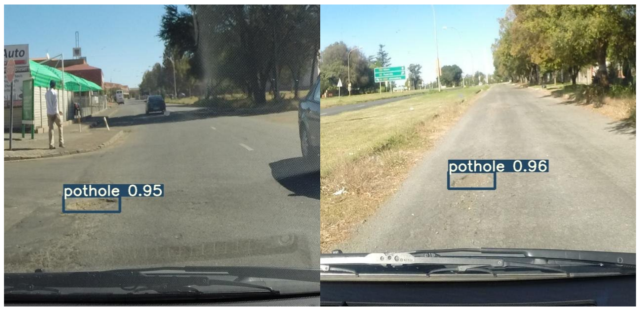

Fig. 5: Examples of correct detection of pothole in test images

There were instances in which potholes were missed or detected at incorrect locations as shown in Fig. 6. The image on the left shows improper detection of potholes on a sidewalk, and the figure on right shows an undetected pothole. However, these instances are rare given the high mAP achieved.

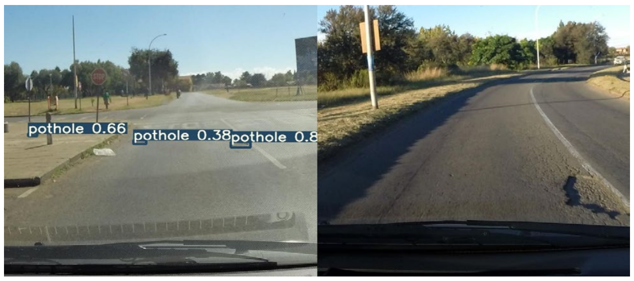

Fig. 6: Examples of improper detection in test images

5. Conclusion

Our solution to pothole detection based on YOLOv5 achieved a satisfactory detection accuracy with mAP values higher than 93% at a detection rate of 2ms per image on average. Experimental results of our proposed solution have shown improvements in both detection accuracy and detection speed compared to previous works. The improvements mainly come from both YOLOv5 model architecture improvements and extensive data augmentations.

There can be improvements in scenarios of improper detection in which potholes are detected on sidewalks or undetected. With improved detection accuracy and speed, pothole detection can be deployed for safer autonomous driving, as well as efficient road maintenance.

References

[1] Glenn Jocher et al. “ultralytics/yolov5: Initial Release”, Zenodo, 2020, http://doi.org/10.5281/zenodo.3908560

[2] S. Nienaber, M.J. Booysen, R.S. Kroon, “Detecting potholes using simple image processing techniques and real-world footage”, SATC, July 2015, Pretoria, South Africa.

[3] S. Nienaber, R.S. Kroon, M.J. Booysen , “A Comparison of Low-Cost Monocular Vision Techniques for Pothole Distance Estimation”, IEEE CIVTS, December 2015, Cape Town, South Africa.

[4] A. Dhiman and R. Klette, "Pothole Detection Using Computer Vision and Learning," in IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 8, pp. 3536-3550, Aug. 2020, doi: 10.1109/TITS.2019.2931297.

[5] D. J, S. D. V, A. S A, K. R and L. Parameswaran, "Deep Learning based Detection of potholes in Indian roads using YOLO," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 381-385, doi: 10.1109/ICICT48043.2020.9112424.

[6] P. A. Chitale, K. Y. Kekre, H. R. Shenai, R. Karani and J. P. Gala, "Pothole Detection and Dimension Estimation System using Deep Learning (YOLO) and Image Processing," 2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ), Wellington, 2020, pp. 1-6, doi: 10.1109/IVCNZ51579.2020.9290547.

[7] P. Ping, X. Yang and Z. Gao, "A Deep Learning Approach for Street Pothole Detection," 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService), Oxford, United Kingdom, 2020, pp. 198-204, doi: 10.1109/BigDataService49289.2020.00039.

[8] E. N. Ukhwah, E. M. Yuniarno and Y. K. Suprapto, "Asphalt Pavement Pothole Detection using Deep learning method based on YOLO Neural Network," 2019 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, Indonesia, 2019, pp. 35-40, doi: 10.1109/ISITIA.2019.8937176.

[9] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” 2016, arXiv:1612.08242.

[10] Redmon, Joseph, and Ali Farhadi. "YOLOv3: An incremental improvement." arXiv:1804.02767 (2018).

[11] Alexey Bochkovskiy et al.” YOLOv4: Optimal speed and accuracy of object detection,” 2020, arXiv:2004.10934

[12] The Cost of Car Damages from Potholes. Pothole Info. https://www.pothole.info/2018/04/the-cost-of-car-damages-from-potholes/

[13] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. CVPR, 2014, pp. 580–587.

[15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. CVPR, 2018, pp. 770–778.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in Proc. CVPR, vol. 1, no. 2, 2017, p. 4.

[17] T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in Proc. ECCV, 2014, pp. 740–755.

[18] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu and Wangmeng Zuo, “Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation,” 2020, preprint arXiv:2005.03572, arXiv.

[19] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye and Dongwei Ren, “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression,” 2019, preprint arXiv:1911.08287, arXiv.

[20] Adrian Rosebrock. (2016, November). Intersection over Union (IoU) for object detection. Pyimagesearch. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[21] seekFire. (2020, July). Overview of model structure about YOLOv5. Git Hub. https://github.com/ultralytics/yolov5/issues/280

[22] Chien-Yao Wang et al. “CSPNet: A New Backbone that can Enhance Learning Capability of CNN”, 2019, arXiv:1911.11929v1

[23] Gao Huang et al. “Densely Connected Convolutional Networks”, 2018, arXiv:1608.06993v5

[24] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”, 2018, arXiv:1708.02002v2

[25] Mingxing Tan et al. “EfficientDet: Scalable and Efficient Object Detection”, 2020, arXiv:1911.09070v7

[26] Shu Liu et al. “Path Aggregation Network for Instance Segmentation”, 2018, arXiv:1803.01534v4