International Journal of Advanced Research in Engineering and Technology (IJARET)
Volume 11, Issue 5, May 2020, pp. 409-419, Article ID: IJARET_11_05_043
Available online athttp://www.iaeme.com/IJARET/issues.asp?JType=IJARET&VType=11&IType=5
ISSN Print: 0976-6480 and ISSN Online: 0976-6499
DOI: 10.34218/IJARET.11.5.2020.043

REFINING YOLOV4 FOR VEHICLE DETECTION

Pooja Mahto, Priyamm Garg, Pranav Seth and J Panda
Department of Electronics & Communication Engineering,
Delhi Technological University, India

ABSTRACT

Real-time vehicle detection is a technology employed in applications like self-driving cars, traffic camera surveillance. Every year we see better and updated state-of-the-art (SOTA) object detectors, but as those are trained on general-purpose datasets (like MS COCO), we miss out on targeted model improvements for vehicular data. The aim of this paper is to improve the newly released, YOLOv4 detector, specifically, for vehicle tracking applications using some existing methods such as optimising anchor box predictions by using k-means clustering. We also carefully hand-pick and verify some key techniques mentioned in the original paper, to optimise YOLOv4 as per the requirements of our dataset (UA-DETRAC).

Our fine-tuned model is also compared with the existing models on a number of performance metrics such as - precision, recall, F1 score, mean average precision, and the average IoU. Our experimental results show that the SOTA model which already has real-time object detection capabilities can be further improved for highly targeted use cases. We urge the readers to expand the scope of the paper (and the original model) to other specific situations as well.

Key words: object detection, convolutional neural network (CNN), YOLOv4, vehicle detection, k-means clustering

Cite this Article: Pooja Mahto, Priyamm Garg, Pranav Seth and J Panda, Refining Yolov4 for Vehicle Detection, *International Journal of Advanced Research in Engineering and Technology (IJARET)*, 11(5), 2020, pp. 409-419.
http://www.iaeme.com/IJARET/issues.asp?JType=IJARET&VType=11&IType=5

1. INTRODUCTION

Object Detection is a sub-field of computer vision that is revolutionizing the way we identify things. Object detection and tracking are important and challenging tasks in many computer vision applications such as surveillance, vehicle detection and autonomous robot navigation. Object detection differs from object classification in terms of bounding box formation. We try to draw a bounding box around the specific object in detection algorithms. There can be multiple bounding boxes in an image, detecting different objects in the image.

There are two approaches to object detection: 1.) Machine learning-based approach, and 2.) Deep learning-based approach.

For the machine learning approach, there is a necessity of defining features, then using techniques for classification. Viola, Jones at al proposed the Viola-Jones algorithm using a boosted cascade of simple Haar features [1]. Scale-invariant feature transform (SIFT) [2] and Histogram of oriented gradients (HOG) [3] are also feature extraction techniques. Support Vector Machines (SVM) are used for classification.

For the deep learning approach, defining features are not important, as these perform end-to-end object detection. Convolutional Neural networks (CNN) is a class of deep neural networks, which find application mainly in image visualisation. Zhao et al provided a review of object detection using a deep learning approach [4]. Region-based CNN algorithms (R-CNN, Fast R-CNN, Faster R-CNN), Single Shot MultiBox Detector (SSD), YOLO are some deep learning-based approaches to object detection.

A standard CNN cannot be used for object detection, due to the variable length of the output layer, as there can be multiple occurrences of the same object in an image. One way to approach the problem of multiple occurrences of an object in an image is the division of the image into different regions of interest and using a CNN for classification of an object's presence in that region. But the problem with this approach is that the objects of interest tend to have different spatial locations in the image. This results in high computation due to a large number of regions of interest.

R-CNN algorithm was proposed by Girschick et al [5] as a solution to the problem of a large number of regions of interest. They proposed a method of extracting 2000 regions from the image, called as 'region proposals', using selective search, and classifying in those region proposals only instead of a large number of regions. Fast R-CNN improvised the previous R-CNN algorithm by generating a convolutional feature map by directly feeding the image to the CNN [6]. Different region proposals are identified from the feature map. This greatly reduced the time to test an image compared to the R-CNN. Shaoqing et al further developed the Fast R-CNN, by removing the selective search and allowing the network to learn the region proposals. They named it Faster R-CNN, as it takes even less time than the previous Fast R-CNN to test the image [7].

A typical object detector architecture consists of four components- the input, the backbone, the head, and the neck. The backbone is the pre-trained network taking the input image and does feature extraction. The head predicts the classes and bounding boxes of objects. The neck serves the purpose of increasing robustness by collecting feature maps from intermediate stages.

These algorithms can be trained and optimized for the problem of vehicle detection. In this paper, we have used the object detection algorithm YOLOv4 and optimised it for vehicle detection. YOLOv4 provides higher accuracy and faster results so as to implement real-time vehicle detection.

The rest of the paper is organized as follows. Section 2 provides the development of the YOLO algorithm, talking about its two latest versions, v3 and v4. Section 3 shows our proposed refinements in the YOLOv4 algorithm to optimize it for more accurate real-time vehicle detection. Section 4 and 5 focus on the implementation and the comparisons with previous techniques, respectively.

2. EXISTING WORK (YOLO)

Joseph Redmon et al formed a new algorithm in 2015 called You Only Look Once (YOLO) [8]. In YOLO, a single convolutional network predicts the classes and bounding boxes for the whole image in one run of the algorithm, unlike the region-based classification algorithms.

The input image is divided into an S x S grid, and each grid is responsible for the detection of an object. In each grid, we take m bounding boxes, and for each bounding box, the network provides an offset value for bounding box and class probability. The bounding boxes which have a class probability above a specific threshold are selected and are further used to locate the object in the image.

2.1. YOLOv3

In their paper for YOLOv3, Redmon et al showed an incremental improvement over YOLOv2 [9][10]. YOLOv3 improvised over its predecessor in various techniques. Firstly, it improved the bounding box prediction by using dimension clusters as anchor boxes. The network then uses logistic regression to predict an objectness score for each bounding box. There is a threshold for each classifier, and classes with higher scores than the threshold are assigned to the bounding box. Secondly, independent logistic classifiers replaced the softmax for better class prediction. Each box then predicts the classes the bounding box may have by multiclass classification. Thirdly, increasing the number of convolutional layers to 53, using the Darknet-53 framework, which is the backbone behind YOLOv3. The network is a hybrid of YOLOv2, Darknet-19, and consists of 3 x 3 and 1 x 1 filters with shortcut connections. Lastly, it makes predictions across 3 different scales, which is similar to the Feature Pyramid Network (FPN) [11]. It adds several convolutional layers from the base feature extractor. At each scale, 3 bounding boxes are predicted. This ultimately gives better features from early in the network, and the predictions at the last layers are benefited from the early computations.

Lecheng Ouyang et al implemented vehicle target detection based on YOLOv3 in complex scenes and showed advantages over traditional target detection algorithms in accuracy and speed [12]. They showed how YOLOv3 can be used for vehicle detection, and it gives an accuracy of 89.16% at 21fps on the VOC dataset. YOLOv3 is extremely fast, running in 22ms at 320 x 320 with 28.2 mAP. Despite being 3 times faster than RetinaNet [13], it has lower accuracy.

2.2. YOLOv4

Bochkovskiy et al proposed the YOLOv4 algorithm with significant changes from the previous version, and much better accuracy [14]. Published in April 2020 it is the latest, most advanced iteration of YOLO, and the first developed by the original authors (Redmon et al). To achieve better accuracy, they designed a deeper and complex network, where they used Dense Block. It contains multiple convolutional layers, with batch normalization, ReLU, after which convolution takes place. Further, a Dense Net is formed consisting of multiple Dense Blocks with transition layers in between [15]. Cross-Stage Partial connections (CSP) then separate the input feature maps of Dense Blocks into 2 parts [16]. One part goes directly to the input to the next transition layer, and the other part will go through the Dense Block. This lowers the computational needs as only one part is going through the Dense Block. For the backbone of the feature extraction, they used the CSPDarknet-53, which uses the CSP connections along with Darknet-53 from the previous YOLOv3. Spatial Pyramid Pooling (SPP) is used as the neck over CSPDarknet-53 as it increases the receptive field, differentiates the most significant feature and does not cause a reduction in speed [17]. In place of the Feature Pyramid Network (FPN) in YOLOv3, here they used Path Aggregation Network (PANet) [18]. For the head, they used the original YOLOv3 network.

Apart from the architecture, it consists of a training strategy to get better accuracy without the extra cost to hardware, which is termed as *Bag of Freebies*. With these, we can get better performance for 'free'. Another set of strategies which give better results at low cost, but not completely free, were termed as the *Bag of Specials*.

There are some additional improvements on design for a robust training on a single GPU like the introduction of new data augmentation methods like CutMix, Mosaic, etc., selection of hyper-parameters while applying general algorithms, and modifications to existing methods to make their design suitable for better detection and training. The creators of YOLOv4 have chosen *CSP darknet 53* backbone, *SPP additional module*, *PANet path-aggregation* neck, *Greedy NMS and YOLOv3* (anchor-based) head as the architecture of YOLOv4.

3. PROPOSED REFINEMENTS IN YOLOV4

We propose the following *additions to YOLOv4* and used a few techniques from the *Bag of Freebies/Specials* to refine vehicle detection on the UA-DETRAC Dataset:

- Anchor box optimisation using k-means clustering (ABK)
- Non-maximum suppression using distance-IoU (DIoU-NMS)
- Spatial Attention Module (SAM)
- Self-adversarial training (SAT)

Anchor boxes (or priors) are used extensively in single-layer object detection algorithms to set the initial dimensions of bounding boxes, as they are better than other unsupervised learning algorithms which are biased towards bounding boxes with large dimensions. These initial dimensions are then corrected, and resized to the ground truth box dimensions, based on some feedback from the neural network. This feedback can be provided by using the K-means clustering technique [26], which initialises the normalisation (correction) process by taking 'K' random boxes (aka means or centroids) as cluster heads. Clusters are then repeatedly assigned around the nearest centroid, and updated based on a certain *threshold* value, until convergence. This threshold can be taken in terms of the Euclidean distance or the Jaccard Index (IoU value), but the latter turns out to be more accurate. Intersection over Union is the overlap between the anchor and the ground truth box, divided by the non-overlap. If the highest IoU is greater than 50%, the anchor box is said to detect the object. Otherwise, it is said to be ambiguous or unlearned.

To ascertain the number of centroids (the k-value) we need to look at the trade-off between the ideal overlap value and the tolerable training time [9]. Increasing the number of centroids will definitely increase the IoU (overlap) but it will also substantially increase the computational overhead, as there is a linear increment in the number of convolution filters resulting in increased training time. Experimentally, we found the average IoU value to be 78.30% for K=12, (as compared to 75.83% for K=9) to be the most optimal. Subsequent increments in the K value didn't translate into meaningful results. As will be seen later, ABK yields a 4%-point increase in mean average precision (mAP@.50), at K=12, by itself.

Bounding box regression is used to optimise the loss function. Zheng et al [19], proposed using distance-based IoU loss instead of generalised IoU (GIoU) over problems of slow convergence. DIoU uses normalised distances between the ground truth (gt) and the bounding box. Non-maximum suppression (NMS) is used to ensure that an object appearing in multiple boxes in the grid is calculated only once. Using IoU as the criterion to eliminate redundant detection boxes, doesn't work well on cases with object occlusion. Hence using DIoU with NMS [20] makes the system less susceptible to occlusion because of the consideration for central distance along with overlap area.

Convolution Block Attention Module (CBAM) is an attention module for feed-forward convolutional neural networks. Here Woo et al [21], propose a simultaneous application of Max Pool and the Average Pool layer to generate a spatial attention map, by concatenating them and then convolving them. Such a Spatial Attention Module (SAM) causes adaptive feature refinement. In YOLOv4, the authors modify SAM to focus on point-wise attention, effectively removing the maximum and average pooling layers. We used SAM at down-sampling points (encoder) and up-sampling points in the detector (decoder). As will be observed later, SAM yields a 12%-point increase in average IoU, by itself.

Figure 1. Predicted Anchor Boxes

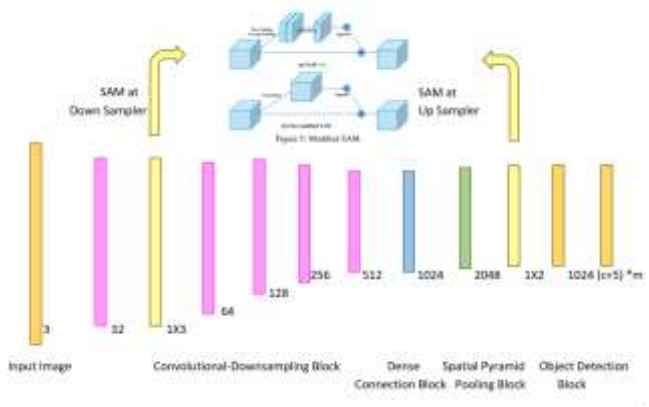

Figure 2. Refined YOLOv4 with Modified SAM

Self-Adversarial Training (SAT) is a data augmentation technique using back-propagation [22] with two iterations of forward-backwards passes. On the first backward pass, to minimize the cost function, we alter the original image instead of the network weights. Contrary to the usual method, this actually degrades the network performance, or simply put, the model performs an "adversarial attack" on itself. Now, this modified image is used to train the network on the second iteration. This way we are able to reduce overfitting and make the model more universal [14]. As will be observed later, SAT yields an 8.2%-point increase in mean average precision (mAP@.50), by itself.

4. DATASET

For training and testing, we use the UA-DETRAC Benchmark dataset [23]. We owe our debt of gratitude to the authors for publicly providing this diverse dataset on vehicle applications for academic research. It consists of 100 video sequences, shot at 25fps at a resolution of 960X540, divided in a 60-40 ratio for training and testing use cases. To avoid overfitting, both kinds of videos are taken at different locations but share similar attributes and traffic conditions. The test set is further divided into *easy*, *medium* and *hard* levels of difficulty based on the Edge Box detection rate [24]. Certain areas of low-resolution are also marked as *ignore-regions* to lower the computational overhead. Over 140,000 manually annotated frames of the dataset are each annotated, using attributes like vehicle category (Bus, Car, Van, Others), occlusion, weather (Sunny, Rainy, Cloudy, Night), truncation, and scale. The dataset specifications and sample data are shown below:

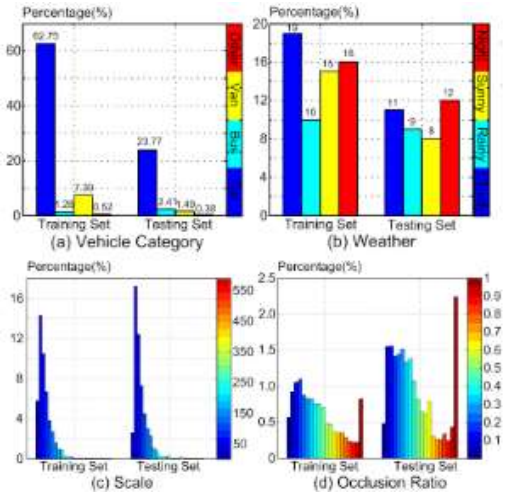

Figure 3. UA-DETRAC Classification

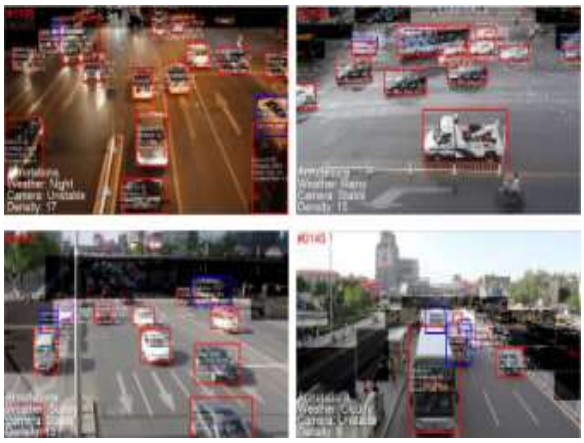

Figure 4. Annotated Frames from Dataset

The data annotation files for DETRAC-Training set were in the XML format and had to be first converted into the VOC format. Then the corresponding images are migrated into the target directory. After this, the data is converted into separate TXT formats suitable for working with YOLO.

5. EXPERIMENTATION

We test the efficacy of various improvement techniques on UA-DETRAC dataset in order to refine and tune YOLOv4 for better Vehicle Detection.

5.1. Experimental Setup

We set the hyper-parameters of YOLOv4 as follows: the training steps are 8000; the step decay learning rate scheduling strategy is adopted with initial learning rate 0.013 and multiplied with a factor 0.1 at 6400 steps and 7200 steps, respectively; The momentum and weight decay are respectively set as 0.9 and 0.0005. All architectures use a single GPU to execute multi-scale training in the batch size of 64 and mini-batch size of 64. We adopt default momentum 0.949, IoU threshold for assigning ground truth 0.213, and loss normalizer 0.07 for genetic algorithm experiments as proposed by authors of YOLOv4. We experimented with a large number of BoF, provided by YOLOv4 including data augmentation using flip, mosaic data augmentation, class label smoothing, self-adversarial training, Optimized Anchors, different kinds of IoU losses. We also conducted experiments on various BoS, including anti-aliasing, assisted excitation, NMS techniques. For all experiments, we used Google's Jupyter notebook Colab with one GPU (1xTesla K80, compute 3.7, 2496 CUDA cores, 12GB GDDR5 VRAM) for training. Therefore, techniques such as *syncBN* which are used to optimize the use of multiple GPUs were not deployed.

5.2. Methodology

We started with a comparison of the default YOLOV3 model with the default YOLOV4 model. Based on the results we shortlisted techniques from BoF and BoS which could be used to further enhance the performance of YOLOv4 for UA-DETRAC data set. From BoF apart from cut-mix and mosaic data augmentation technique and label smoothing we use flip data augmentation technique, self-adversarial training technique. From BoS, we use DIoU-NMS and SAM. We also use 12 customised anchors generated by K-mean Clustering, instead of the 9 standard Anchors of YOLOv4.

6. RESULTS AND COMPARISONS

We use the following performance metrics to compare the various models and iterations. (Assuming TP, TN, FP, FN denote the true positives, true negatives, false positives and false negatives, respectively)

6.1. Average IoU

Intersection over Union is the ratio of area common [intersection] between the ground-truth box (B_{gt}) and the predicted bounding box (B_p), and the total area by these two boxes [union]. Mathematically,

$$IoU = area(B_p \cap B_{gt}) / area(B_p \cup B_{gt}) \; 0 \leq IoU \leq 1$$

Average IoU is the IoU value averaged over the entire dataset.

6.2. Precision

Precision is a measure of what proportion of predicted positives were truly positive. Mathematically,

$$Precision = TP / (TP + FP)$$

6.3. Recall

Recall is a measure of what proportion of actual positives were classified correctly.

$$Recall = TP / (TP + FN)$$

6.4. F1 Score

F1 score [25] is the harmonic mean of recall and precision. It maintains a balance between recall and precision, as there is a trade-off between the two (one value is improved at the expense of the other). Mathematically,

$$F1\ Score = 2 \times (Precision. Recall) / (Precision + Recall)$$

6.6. Average Precision

Average precision is a measure that combines recall and precision for ranked retrieval results. For one information need, the average precision is the mean of the precision scores after each relevant document is retrieved.

6.6. Mean Average Precision (mAP [0.50])

Mean of the Average Precisions calculated over multiple classes (here 4), using single IoU threshold of 0.5.

We tested our models for 40 videos divided into 56,167 frames consisting of four classes i.e., Bus, Car, Van, Others. In comparison between the default YOLOv3 model and the default YOLOv4 model, we found that YOLOv3 outperforms YOLOv4 for the *Bus* class. And performs almost the same for the *Van* class. While analysing weather conditions we noticed YOLOv3 performs much better than YOLOv4 for night condition with an 8%-point higher mAP(@.50). After analysing the architecture of both the default models we reached to the following conclusions:

- Higher AP of the *bus* class indicates the anchor size of YOLOv3 is more suitable for detection of buses than anchor sizes of YOLOv4.
- Poor performance of YOLOv4 during the *night* is a result of the mosaic data augmentation technique used, as mosaic combines four images together. But due to

the difference in background intensity levels, the model is not able to adapt to the changes.

- Low precision indicates that YOLOV4 detected lesser TP than YOLOv3.
- Average IoU of YOLOv4 is 10% lesser than average IoU of YOLOv3 which again indicates incompatible anchor boxes.
- Low AP of the *others* class indicates that both the models were unable to detect vehicles of this class properly, as they were not well-defined and hence, detected as noise by the system. *Others* include bikes, trucks and other vehicles.

Figure 5 Detection on the Test data

In *YOLOv4+ABK*, we used 12 anchor boxes generated by K- Means Clustering Method. AP of class bus, van, others are increased by a significant amount of 10.72% , 8.35%, 1.23% respectively whereas we notice a drop in AP of class cars. Average IoU has increased by 15% and we notice a significant increase in overall the model with a 4%-point increment in mAP(@.50).

In *YOLOv4+SAT*, we use Self Adversarial Training which adds noise while training the model to make the model robust to noises in test data and adversarial attacks. YOLOv4+SAT increments the AP of class Others by a point of 17.72% and bus by 12.14% as compared to the default YOLOV4 model. Increment of 8.6%-point in mAP(@.50) and 14%-point in Average IoU is observed.

YOLOv4+SAM uses Spatial Attention Module which tells the model where to focus while training and detection, basically where to look for the features in a given image. This helps the model to overcome the intensity problem faced during the *night* condition. As a result, *average IoU* is increased by 13%-point and an increment of 0.5%-point in mAP(@0.50) is noticed.

Refined YOLOv4 uses all of the above-mentioned techniques i.e. ABK, SAT, SAM and uses DIoU NMS for non-maximal suppression which results in an AP of 85.11%, 76.57%, 59.67%, 49.25% for bus, car, van and others. High Precision of 77%, high recall of 71% and an F1-score of 74% is observed. Average IoU was found to be 65.7%. And an overall increment of 10%-point in mAP(@.50) at 67.7% is observed. **Refined YOLOv4: YOLOv4+ABK+SAT+SAM*

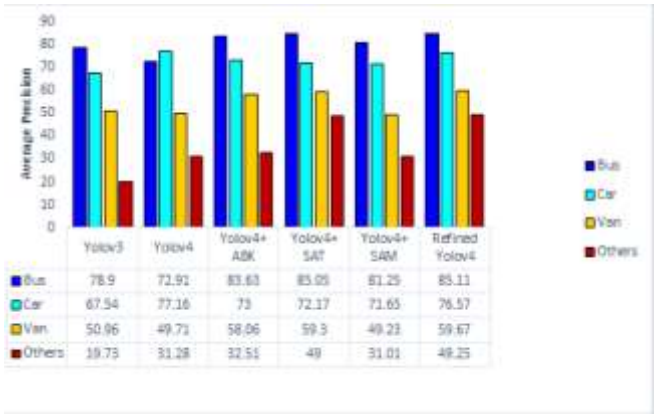

Figure 6: Influence of different YOLO models on AP of different classes

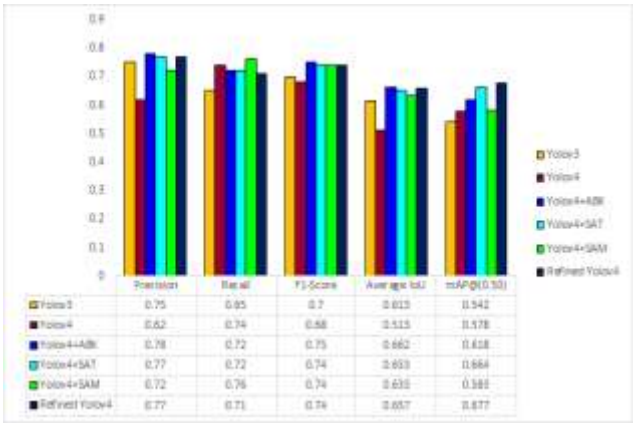

Figure 7: Influence of different YOLO models on various performance metrics.

7. CONCLUSION

In this paper, we fine-tuned the new state-of-the-art object detection algorithm YOLOv4, to specifically suit the needs of vehicle detection, using some pre-existing tweaks and techniques. Wherever possible we have experimentally verified the results and presented the same in the paper. The refined YoloV4 model boosted the performance in terms of each individual aforementioned metric. This final combined model gives benchmark results with an mAP of 67.7% (10%-point higher than base model) on the DETRAC-test dataset. We also observed YOLOv4 to be definitely faster than the previous iterations. In our calculations it was able to test around 57K frames in 25 minutes on an average, translating into a frame rate of around 38 FPS, enabling real-time computation on a single GPU. The results offer a promising scope of application of varied techniques and tweaks to optimise YOLOv4 for other specific situations (IC detection, Crack detection, face detection etc).

REFERENCES

[1] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision, 57*(2), 137-154.

[2] Lowe, David G. (1999). Object recognition from local scale-invariant features. *Proc. 7th International Conference on Computer Vision (ICCV'99)* (Corfu, Greece): 1150-1157. doi:10.1109/ICCV.1999.790410.

[3] Navneet Dalal, Bill Triggs. (2005), Histograms of Oriented Gradients for Human Detection. *International Conference on Computer Vision & Pattern Recognition (CVPR '05)*, Jun, San Diego, United States. pp.886–893, ff10.1109/CVPR.2005.177ff. ffinria-00548512f

[4] Zhao, Zhong-Qiu & Zheng, Peng & Xu, Shou-Tao & Wu, Xindong. (2019). Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*. PP. 1-21. 10.1109/TNNLS.2018.2876865.

[5] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[6] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

[7] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[8] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[9] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

[10] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[11] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125).

[12] Ouyang, L., & Wang, H. (2019, July). Vehicle target detection in complex scenes based on YOLOv3 algorithm. In *IOP Conference Series: Materials Science and Engineering* (Vol. 569, No. 5, p. 052018). IOP Publishing.

[13] T. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, (2017), "Focal Loss for Dense Object Detection," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, pp. 2999-3007, doi: 10.1109/ICCV.2017.324.

[14] Bochkovskiy, A., Wang C., Mark Liao, H. (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection". arXiv:2004.10934v1 [cs.CV] 23 Apr 2020

[15] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

[16] Wang, C., Liao, H.M., Yeh, I., Wu, Y., Chen, P., & Hsieh, J. (2019). CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *ArXiv, abs/1911.11929*.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. (2015), Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 37(9):1904–1916.

[18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. (2018), Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8759–8768.

[19] Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; and Ren, D. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. AAAI 2020

[20] Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Soft-NMS--improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision* (pp. 5561-5569).

[21] Woo, S., Park, J., Lee, J., & Kweon, I. (2018). CBAM: Convolutional Block Attention Module. *ArXiv, abs/1807.06521*.

[22] Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65-93). Academic Press.

[23] Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M. C., Qi, H., & Lyu, S. (2015). UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *arXiv preprint arXiv:1511.04136*.

[24] Zitnick, C. L., & Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *European conference on computer vision* (pp. 391-405). Springer, Cham.

[25] Sasaki, Yutaka. (2007). The truth of the F-measure. Teach Tutor Mater.

[26] Jin X., Han J. (2017) K-Means Clustering. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA

[27] Suraj Nair Aiyappa, Ramamurthy B, (2018), An Efficient Approach Towards Clustering Using K-Means Algorithm, International Journal of Civil Engineering and Technology, 9(2), pp. 705–714

[28] Sasikaladevi V and Mangai V, (2018), Colour Based Image Segmentation Using Hybrid K means with Watershed Segmentation, *International Journal of Mechanical Engineering and Technology*, 9(8), pp. 1367–1377.

[29] Ch. Ramesh, Dr. N.B. Venkateswarlu, Dr. J.V.R. Murthy, (2012), A Novel K-Means Based Jpeg Algorithm for Still Image Compression, *International Journal of Computer Engineering and Technology*, 3(1), pp. 339–354

[30] Chandra Das, Shilpi Bose, Matangini Chattopadhyay, Samiran Chattopadhyay, (2014), A Novel Distance Based Modified K-Means Clustering Algorithm for Estimation of Missing Values In Micro-Array Gene Expression Data, *International Journal of Information Technology & Management Information System*, 5(3), pp. 1–13