2020 6th International Conference on Advanced Computing & Communication Systems (ICACCS)

YOLO v3-Tiny: Object Detection and Recognition using one stage improved model

[1]Pranav Adarsh
Department of Computer Science & Engineering
Delhi Technological University
Delhi, India
pranavadarsh95@gmail.com

[2]Pratibha Rathi
Department of Computer Science & Engineering
Delhi Technological University
Delhi, India
pratibharathi96@gmail.com

[3]Manoj Kumar
Department of Computer Science & Engineering
Delhi Technological University
Delhi, India
mkumarg@dce.ac.in

***Abstract*—Object detection has seen many changes in algorithms to improve performance both on speed and accuracy. By the continuous effort of so many researchers, deep learning algorithms are growing rapidly with an improved object detection performance. Various popular applications like pedestrian detection, medical imaging, robotics, self-driving cars, face detection, etc. reduces the efforts of humans in many areas. Due to the vast field and various state-of-the-art algorithms, it is a tedious task to cover all at once. This paper presents the fundamental overview of object detection methods by including two classes of object detectors. In two stage detector covered algorithms are RCNN, Fast RCNN, and Faster RCNN, whereas in one stage detector YOLO v1, v2, v3, and SSD are covered. Two stage detectors focus more on accuracy, whereas the primary concern of one stage detectors is speed. We will explain an improved YOLO version called YOLO v3-Tiny, and then its comparison with previous methods for detection and recognition of object is described graphically.**

Keywords—Computer vision; YOLO v3; Faster RCNN; Deep learning; YOLO v3-Tiny; Object detection; image processing; Convolutional Neural Networks.

I. INTRODUCTION

In the recent few years, diverse research work happened to develop a practical approach to accelerate the development of deep learning methods. Numerous developments accomplished excellent results and followed by continuous reformations in deep learning procedures. Object localization is the identification of all the visuals in a photograph, incorporating the precise location of those visuals. By using deep learning techniques [1] [2] for object identification and localization, computer vision has reached a new zenith. Due to significant inconstancies in viewpoints, postures, dimensions, and lighting positions, it is challenging to succeed in the identification of objects perfectly. Accordingly, considerable concern has been given by researchers to this area in the past few years. There are two types of object detection algorithms. Object detection algorithms using region proposal includes RCNN [3], Fast RCNN [4], and Faster RCNN [5], etc. These techniques create region proposal networks (RPN) [6], and then the region proposals are divided into categories afterward. On the other side, object detection algorithms using regression includes SSD [7] and YOLO [8], etc. These methods also generate region proposal networks (RPN) but divide these region proposals into categories at the moment of generation. All of the procedures mentioned above have significant accomplishments in object localization and recognition. YOLO consolidates labels in diverse datasets to form a tree-like arrangement, but the merged labels are not reciprocally exclusive. YOLO9000 [9] enhances YOLO to recognize targets above 9000 categories employing hierarchical arrangement. Whereas YOLOv3 [10] uses multilabel classification, it replaces the approach of estimating the cost function and further exhibits meaningful improvement in distinguishing small targets. The arrangement of this paper is as follows. Below in section 2, background information of object detection methods is covered. It includes two stage detectors with their methodologies and drawbacks. Section 3 elaborates one stage detectors and the improved version YOLO v3-Tiny [10] [11]. Section 4 describes implementation results and comparison of object detection methods based on speed and accuracy. Finally, section 5 summarizes the conclusion.

II. BACKGROUND

In this section, we present background information. It elaborates the most representative and pioneering two-stage object detection methods with their significant contributions in object detection. First, we examine their methodologies and then explain their drawbacks.

A. HOG

HOG is a feature descriptor that is extensively applied in various domains to distinguish objects by identifying their

shapes and structures. Local object structure, pattern, aspect, and representation can usually be characterized by the arrangement of gradients of local intensity or the ways of edges. In the HOG [12] detection method, the first step is to break the source image into blocks and then distribute each block in small regions. These regions are called cells. Commonly, the blocks of image overlap each other, due to this corresponding cell may be a part of many blocks. For each pixel inside the cell, it calculates the gradients vertically and horizontally.

Drawbacks of HOG method- Due to the emergence of deep learning and its significant applications, the reasonable opinion was to displace classifiers deployed on HOG [12] methodology with a classifier based on convolutional neural network [2][13] because of its comparatively higher accuracy. But there were some problems. The computation cost of convolutional neural networks was high, and the speed is too slow. Therefore, it was challenging to run CNN based classifier on numerous patches produced by the detection approach of sliding window. This difficulty was resolved by RCNN [3]. It employs an algorithm based on object proposals termed selective search method [14]. This approach decreases the number of bounding boxes to 2000 region proposals that were supplied to the R-CNN classifier.

B. RCNN

Region based convolutional neural networks (RCNN) algorithm [3] uses a group of boxes for the picture and then analyses in each box if either of the boxes holds a target. It employs the method of selective search to pick those sections from the picture. In an object, the four regions are used. These are varying scales, colours, textures, and enclosure.

Drawbacks of RCNN method- Based on a selective search [14], 2,000 sections are excerpted per image. For every region or part of the image, we have to select features using CNN. For this, if we have 'i' number of images, then selected regions will become i×2,000. The whole method of target identification through RCNN utilizes the following three models: Linear SVM classifier for the identification of objects, CNN is employed for characteristic extraction, and a regression model is required to tighten the bounding boxes. All these three processes combine to take a considerable amount of time. It increases the running time of RCNN method. Therefore, RCNN needs almost 40 to 50 seconds to predict the result for several new images [15].

C. FAST RCNN

In place of using three different models of RCNN, Fast RCNN [4] employs one model to excerpt characteristics from the different regions. Then it distributes the regions into several categories based on excerpted features, and the boundary boxes of recognized divisions return together. Fast RCNN uses the method of spatial pyramid pooling [16] to calculate only one CNN representation for the whole image. It passes one region for each picture to a particular convolutional network model by replacing three distinct models for excerption of characteristics, distributing into divisions, and producing bounding boxes.

Drawbacks of Fast RCNN method- Fast RCNN also employ a selective search method [14] to detect concerned regions. This method is prolonged and demands a lot of time. Usually, for the detection of objects, this complete procedure needs almost two seconds for each picture. Therefore its speed is quite good in contrast to RCNN [15]. However, if we contemplate extensive real-life datasets, then the execution of fast RCNN approach is still lacked in speed.

D. FASTER RCNN

Faster RCNN [5] is a transformed variant of fast RCNN. The significant difference between both is that faster RCNN implements region proposal network (RPN) [6] [17], but fast RCNN utilizes a selective search technique for producing concerned regions. In input, RPN accepts feature maps of picture and produces a collection of object recommendations and an objectness score per recommendation in output. Usually, this approach takes ten times less time in contrast to fast RCNN approach because of RPN.

Drawbacks of faster RCNN method- To excerpt all the targets in a given picture, this procedure needs multiple passes for that particular picture. Different systems are working in a sequence therefore, the performance of the upcoming operation is based on the performance of preceding operations. This approach uses region proposal networks to localize and identify the objects in a picture. But RPNs do not contemplate the complete picture because it uses only those portions of the picture, which have high probabilities of the presence of targets [15].

Table 1: Comparison of two stage object detectors

Type	Characteristics	Computation time	Drawbacks
RCNN	To generate regions, it uses selective search. From each picture, it extracts around 2000 regions.	Forty to Fifty seconds	Time taken for prediction is large because several regions pass through CNN definitely, and it employs three distinct models for the detection of targets.
Fast RCNN	To excerpt the features, each picture passes one time through CNN. All distinct models applied in RCNN are combined collectively to	Two seconds	The method used is prolonged and time-consuming Therefore, computation time is still high.

	form a single model. It employs selective search method on the feature maps to produce result for target recognition.		
Faster RCNN	The previous approach is replaced with the region proposal networks. Therefore, this procedure works much faster compared to previous methods.	0.2 seconds	Object region proposal is time-consuming. Different types of systems are operating in sequence. Thus, the performance of entire procedure is based on the working of the preceding operations.

III. ONE STAGE DETECTORS

Two stage detectors provide adequate accuracy, but the time taken for computation is high. Therefore, to process in less time by managing sufficient accuracy, one stage detectors are proposed. Some algorithms in one stage model are SSD and the variants of YOLO. By improving the architecture of two stage models and introducing some changes such as eliminating pipeline, one stage model has achieved excellent speed. But, it has not attained sufficient accuracy at the same time. Hence, continuous changes are under process by researchers.

A. YOLO v1

By using the approach of one stage detector YOLO, an input picture is distributed to a system of S×S grids [8]. The detection of the object depends on every grid of the input image. Grid cells are employed to predict targets inside boundary boxes. Five parameters are predicted for every boundary box. These five elements are i, j, k, l, and c. In the input picture, the centre of target inside the box is denoted by 'i' and 'j' coordinates. Here, 'k', 'l', and 'c' represent height, width, and score for confidence respectively. 'c' is measured as the probability of containing the target inside boundary box.

YOLO v1 uses the Darknet framework and ImageNet-1000 dataset to train the model. It distributes the given picture to a grid of S×S cells. For every cell in the network, it computes confidence for 'n' bounding boxes. The predicted result is encoded into a tensor as $S \times S \times (n \times 5+ p)$ [8]. Here, input image is divided into total S×S sub-images. In n×5, five

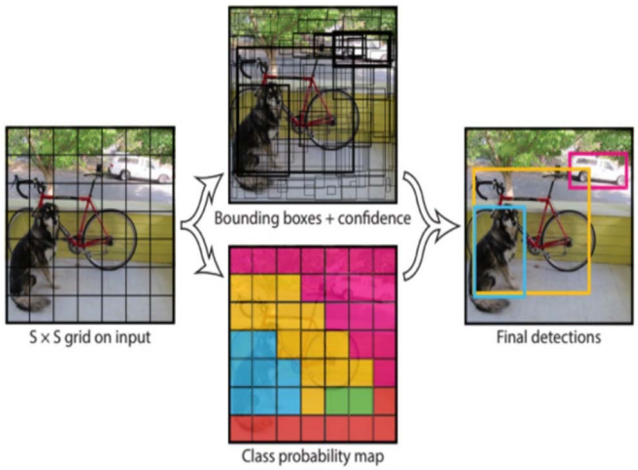

Fig.1: S×S grid representation of YOLO [8]

represents the detection of five attributes for each bounding box, which are height, weight, confidence score, and centre coordinates(x, y) of detected objects. Here, 'p' represents the probability of class. YOLO v1 also has many limitations. Therefore, the use of YOLO v1 is restricted to some extent. Limitations of YOLO version1 are based on the closeness of the objects in the picture. If the objects appear as a cluster, they could not find the small objects. If the dimensions of the object are different from the image used in training data, then this architecture found difficulty in the localization and detection of objects [15][18]. The primary concern is to locate objects in a given picture due to the error of localization.

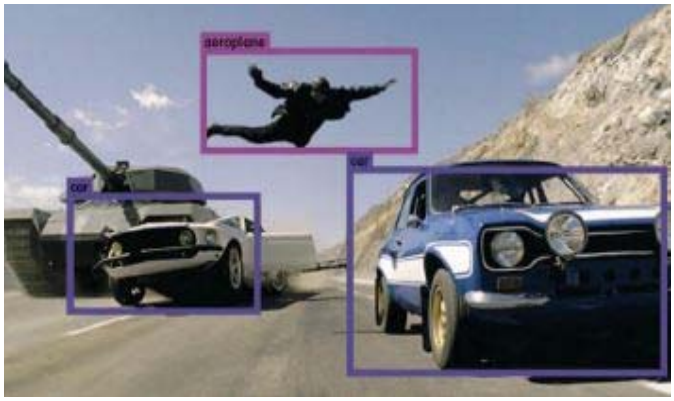

Fig.2: Drawback of Yolo v1 [8]

YOLO v1 fails sometimes, as in the above image it recognizes the man as an airplane.

B. YOLO v2

Yolo v2 supersedes Yolo by offering a great balance between running time and accuracy. For better accuracy, Yolo v2 introduces batch normalization, which helps to enhance 2 percent in map by attaching it into each layer of convolution. High resolution classifier is used to operate thoroughly by modifying its filters for giving a more extensive understanding of network time to work excellently. When it comes to the prophecy of bounding boxes, then by eliminating entirely

connected layers to anchor boxes makes decrement of map value by 0.3, but recall value is incremented by 7% [9]. Hence, it gives more potential to detect the day to day objects. Fine grained features help to identify tiny objects by reshaping the layers employing a modified approach called pass through. As we have achieved accuracy now, our goal is to maintain a balance of running time with it. For this Yolo v2 uses darknet 19, although it could use google net, but that reduces accuracy by 2%. For learning to locate targets from any visual, Yolo 9000 is employed across nine thousand categories with a 9418 node WordTree. It is certainly progress for concluding the localization and distribution.

C. YOLO v3

Object detection is used in many fields of human life, for example, health and education, etc. As all these fields are growing rapidly, so to match their requirements, one stage models also need improvement. The next advanced variant of YOLO is version 3 that uses logistic regression to compute the targetness score. It gives the score for all targets in each boundry box. YOLO v3 can give the multilabel classification because it uses a logistic classifier for each class in place of the softmax layer used in YOLO v2. YOLO v3 uses darknet 53. It has fifty-three layers of convolution. These layers are more in-depth compared to darknet 19 used in YOLO v2. Darknet-53 contains mainly 3x3 and 1x1 filters along with bypass links [8] [9] [10]. The formulas given below explain the transformation of network output for obtaining bounding box predictions. Here, d_x and d_y are center coordinates, d_w is width and d_h is the height of predicted result. Top left coordinates of the grid are m_x and m_y. Network outputs are t_x, t_y, t_w and t_h. Anchor dimensions for the box are p_h and p_w.

$$d_x = \sigma(t_x) + m_x \quad (1)$$

$$d_y = \sigma(t_y) + m_y \quad (2)$$

$$d_w = p_w e^{t_w} \quad (3)$$

$$d_h = p_h e^{t_h} \quad (4)$$

By using threshold value, a filter is applied to remove the box having class score less than the threshold value chosen. Because a score with less value represents that the box is insufficient for identifying the classes. Even after filtering by using a threshold value for the class scores, a lot of overlapping boxes still remain. When many boxes are overlapping with each other, then select only one box out of those overlapping boxes and identify the object. So, the second filter is used for choosing the desired boxes, which termed as nonmaximum suppression (NMS) [19]. It uses intersection over union (IOU) function.

$$IOU = \frac{B_1 \cap B_2}{B_1 \cup B_2} \quad (5)$$

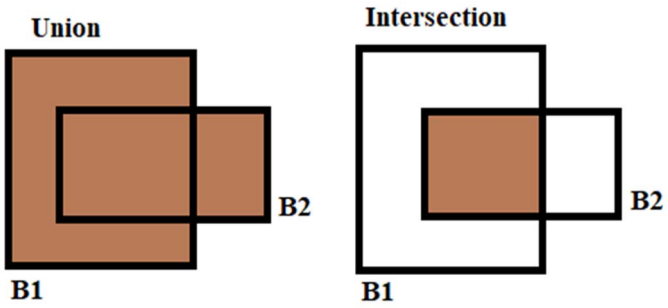

Fig. 3: Illustration depicting the definition of union and intersection

Above-left corner (a1, b1) and below right corner (a2, b2) are used to determine a box. For calculating the area of the rectangle, multiply its height (b2-b1) and its width (a2-a1). Then the coordinates (ai1, bi1, ai2, bi2) for the intersection of 2 boxes are obtained. Here, ai1 and bi1 are the highest value of a1 and b1 coordinate-position of the two boxes. Similarly, ai2 and bi2 is the lowest value of the a2 and b2 coordinate-position of the 2 boxes.

$$\text{union}(A, B) = A + B - \text{inter}(A, B) \quad (6)$$

$$\text{inter\_area} = (ai2 - ai1) \times (bi2 - bi1) \quad (7)$$

$$\text{box1\_area} = (box1[3] - box1[1]) \times (box1[2] - box1[0]) \quad (8)$$

$$\text{box2\_area} = (box2[3] - box2[1]) \times (box2[2] - box2[0]) \quad (9)$$

$$\text{union\_area} = (\text{box1\_area} + \text{box2\_area}) - \text{inter\_area} \quad (10)$$

$$\text{IOU} = \text{inter\_area}/\text{union\_area} \quad (11)$$

The advantage of YOLO v3 over YOLO v2 is that some changes are included in error function and for objects of small to a considerable size detection occurs on three scales. The multiclass problem turned in a multilabel problem, and the performance improved over small size objects.

D. SSD

SSD is a single shot detector [7] [20]. It manages an excellent balance of speed with the accuracy of result. In this, we apply for single time a CNN based model to the input picture for computing the feature map. It also employs anchor boxes similar to faster RCNN at various aspect ratios and learns the offset instead of determining the box. The processing occurs on numerous layers of CNN, where every layer functions on a varying range of scale and uses multiple feature maps. Therefore, the detection of targets of several sizes is possible. Experimentally, SSD has much better accuracy on different datasets even on inputs pictures of small size as compared to the other single stage methods.

E. SSD vs YOLO

Unlike YOLO, SSD does not divide the image into grids of random size. For every location of the feature map, it predicts the offset of predefined anchor boxes (default boxes). Relative to the corresponding cell, each box has a fixed size, proportion, and position. In a convolutional manner, all the

anchor boxes cover the entire feature map. Anchors of SSD are slightly different from the anchors of YOLO. Because YOLO makes all the predictions from a single grid, and the size of anchors used by YOLO ranges from dimensions of one grid cell to the dimensions of the entire picture. The anchors of SSD specialize its detector for distinct feasible viewpoints and dimensional ratios of its target shapes, but not enough on the size of targets. The calculation for the anchors of SSD uses a simple formula, while the anchors of YOLO are calculated by applying k-means clustering on the training data [21] [15]. SSD doesn't use confidence score, but YOLO calculates it to show the faith in predicted results. A unique background class is employed by SSD for this work. A low value of confidence score in YOLO is equivalent to the predicted output of background class in SSD. Both indicate that for the detector, the possibility of getting a target is null [18].

F. YOLO v3-Tiny

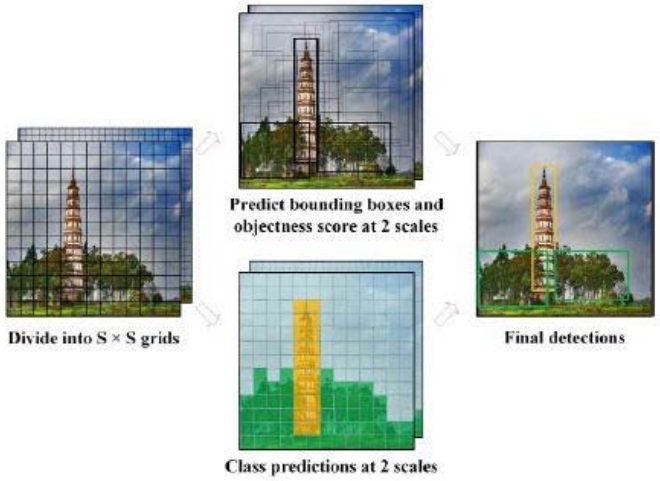

Fig. 4: Main process of YOLO v3-Tiny [11]

YOLO v3, with the decreased depth of the convolutional layer, is another version called YOLO v3-Tiny. It was proposed by Joseph Redmon [10]. Therefore, the running speed is significantly increased (approximately 442% faster than the former variants of YOLO), but detection accuracy is reduced. Darknet-53 architecture of YOLO v3 employs several 1x1 convolution layers along with 3x3 convolution layers for extracting features [22]. YOLO v3-Tiny uses pooling layer and reduces the figure for convolution layer. It predicts a three-dimensional tensor that contains objectness score, bounding box, and class predictions at two different scales. It divides a picture into S×S grid cells. For final detections, we will ignore the bounding boxes for which the objectness score is not best. For extracting features, convolution layers and max-pooling layers are utilized in the feed forward arrangement of YOLO v3-Tiny [11] [23] [24]. Prediction of bounding boxes occurs at two different feature map scales, which are 13×13, and 26×26 merged with an upsampled 13×13 feature map.

Table 2: Performance comparison of YOLO with their Tiny versions[25]

Detector	Number of n-layers	FLOPS	FPS	map value	Used set of data
YOLO v1	26.00	Not-given	45.00	63.50	VOC-data
YOLO v1-Tiny	9.00	Not-given	155.00	52.80	VOC-data
YOLO v2	32.00	62.95	40.00	48.20	COCO-data
YOLO v2-Tiny	16.00	05.42	244.00	23.60	COCO-data
YOLO v3	106.00	140.70	20.00	57.80	COCO-data
YOLO v3-Tiny	24.00	05.57	220.00	33.20	COCO-data

In Table 2, we can see that on different datasets, the tiny versions of YOLO have a lower value for map as compared to their original versions.

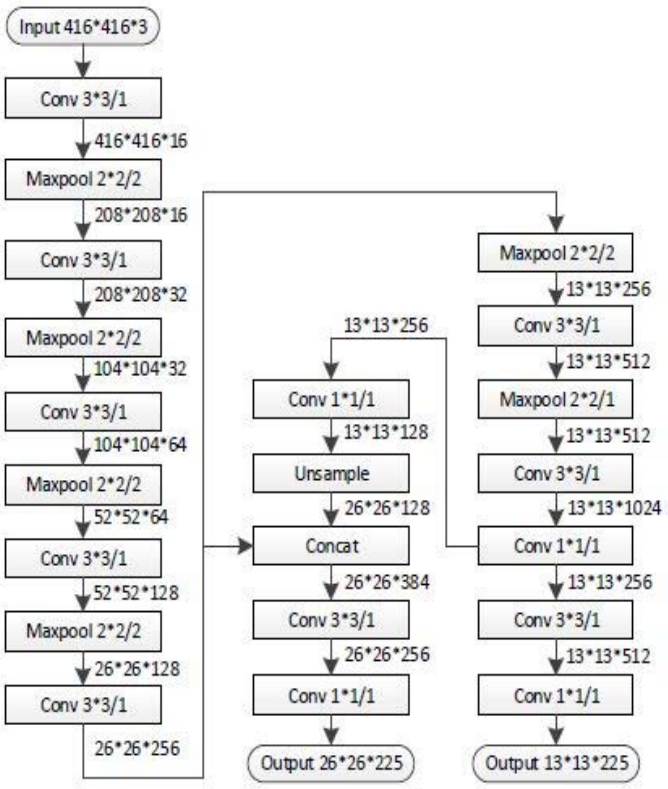

Fig.5: Architecture of YOLO v3-Tiny [11]

IV. IMPLEMENTATION RESULTS AND ANALYSIS

It is tough to make a clear comparison between different object detection methods. So, we can't give a straight decision on the best model. For many real-life applications, we make choices to create an equilibrium of accuracy with speed. Therefore, we need to be aware of other characteristics that have a significant impact on performance. For example, matching strategy and IOU threshold, ratio of positive anchor and negative anchor, training dataset, utilization of varying scale and cropped pictures for training, location loss function, pace of learning, and learning rate decay etc.

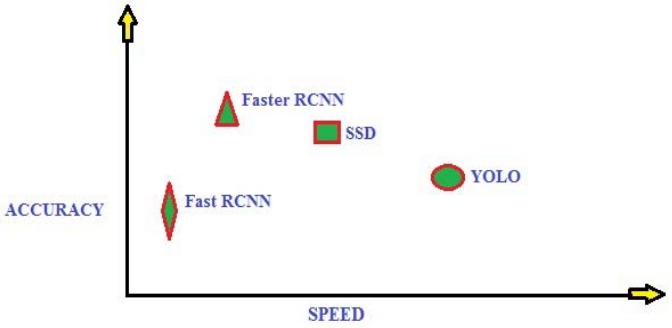

Fig. 6: Speed vs. Accuracy for object detection methods

We have thoroughly examined the characteristics of three procedures for object detection, which are faster RCNN, YOLO v3, and SSD [26]. If the requirement is accuracy (excellent quality of correctness), then faster RCNN is the best as it is much accurate compared to SSD and YOLO v3. But if accuracy is not the primary concern, we want super-fast speed, YOLO v3 takes less time compared to SSD and faster RCNN. But if at the same time requirement is of excellent accuracy and less running time, then SSD is a more favorable recommendation, as its speed is better than faster RCNN, and its accuracy is better compared to YOLO v3.

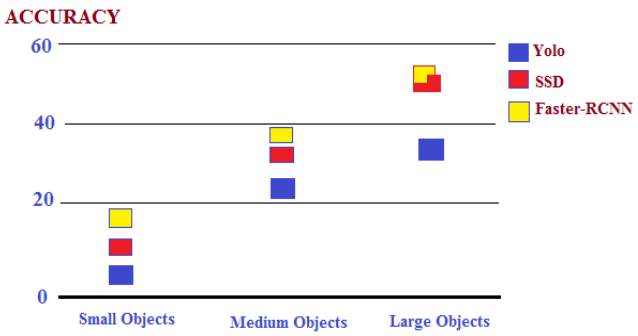

Fig.7: Accuracy comparison for different sizes of target objects

The chart given above in Fig. 7 represents the faster RCNN, YOLO, and SSD performance on targets of distinct sizes. In accuracy comparison for the big targets, SSD performs alike faster RCNN. But as the object size decreases, the gap increases [15] [18] [27].

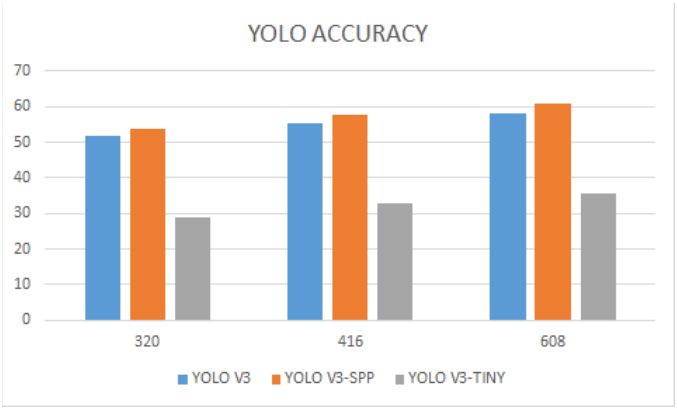

Fig.8: Accuracy comparison of YOLO algorithms

YOLO v3-Tiny is a lightweight variant of YOLO v3, which takes less running time and less accuracy when examined with YOLO v3. In Fig. 8, we compared the accuracy of different versions of YOLO algorithms for given image pixels.

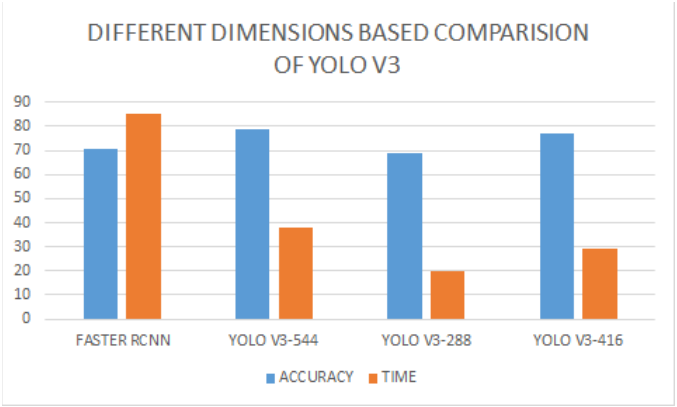

Fig.9: Accuracy and speed comparison of YOLO with faster RCNN

In Fig. 9, the accuracy and speed performance of YOLO algorithms is compared with faster RCNN [27]. As per the result, we can say that YOLO v3 takes less time compared to Faster R-CNN, and YOLO v3-tiny is faster than YOLO v3.

We have implemented YOLO v3 and YOLO v3-tiny using conda, open CV python, and TensorFlow 2.0 with the darknet 53 model, which is earlier trained on the imagenet dataset for classification. As training a model from scratch is much complicated, so we employed transfer learning on our model using VOC 2012 dataset for localization of objects. We have used a PC comprising Windows 10, intel core i5 eighth-generation quad-core CPU, NVIDIA Geforce GTX 1050 GPU, and a RAM of size 8 gigabytes. It also demands an internet of extremely high-speed. Programming Language used is python. We have taken input as preprocessed image. The output is in the form of bounding boxes, accuracy score percentage, and class label. During the training of model, ninety percent of data is applied for training, and the rest ten percent is used for the validation of our model. We trained our model for localizing twenty classes that include person, dog, and car, etc. During batch normalization, the size of each batch is sixteen and requires approximately twenty iterations

to get a valid result. The duration of the period needed for each iteration is around ten minutes. Then the trained models of YOLO v3 and YOLO v3-Tiny are applied to the pictures taken by us in our college campus to examine the execution time and accuracy (map value) of the result obtained. Speed is calculated in terms of frame per second, which is the number of images processed by the model in each second. Mean average precision is computed to estimate accuracy. After each iteration, precision and recall value is calculated, and a curve is plotted between them. The area under this curve is an average precision of that iteration. At last, we will find the mean value of all obtained average precisions to get map value. We have observed that YOLO v3-Tiny is faster compared to YOLO v3, and therefore, it is more effective in real time target localization and identification of class. However, its accuracy decreases a bit in comparison to YOLO v3.

Fig.10: Experimental result-1 of YOLO v3

Fig.11: Experimental result-1 of YOLO v3-Tiny

Fig.12: Experimental result-2 of YOLO v3

Fig.13: Experimental result-2 of YOLO v3-Tiny

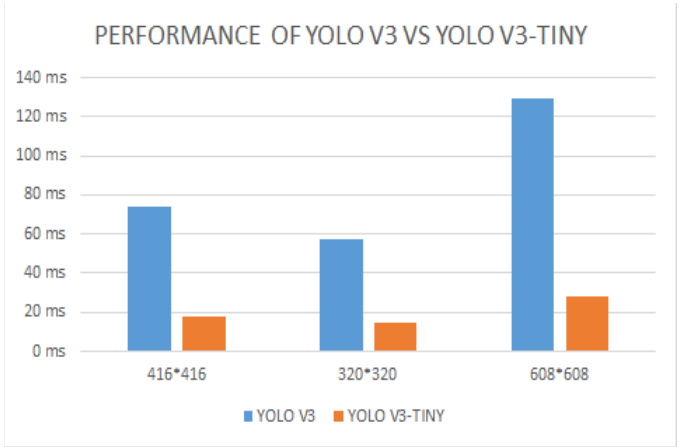

Fig.14: Speed Comparison

The above graph in Fig. 14 shows a comparison of the running time of YOLO v3 with YOLO v3-Tiny for different dimensions of images.

V. CONCLUSION

To identify and localize objects, there exist many methods with a trade-off in speed performance and accuracy of result. But yet we can't say any single algorithm is best over others. One can always select the method that suits the requirement at best. In a short period, object detection applications got much popularity and still a lot to cover in this area because of its vast scope of research[15] [18] [26]. This paper presents the comparison of various algorithms to identify and localize objects based on accuracy, time, and parameter values with varying sizes of the input image. We have identified a new methodology of single stage model for improving speed without sacrificing much accuracy. The comparison results show that YOLO v3-Tiny increases the speed of object detection while ensures the accuracy of the result. We can also extend object localization and recognition from static pictures to a video containing the dynamic sequence of images. We are planning to increase the accuracy of the localization of small targets in the future.

References

[1] A. Krizhevsky, I. Sutskever, and G. E.Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *NIPS*, 2012, doi: 10.1201/9781420010749.

[2] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra, and J. M. Z. Maningo, "Object Detection Using Convolutional Neural Networks," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2018-October, no. October, pp. 2023–2027, 2019, doi: 10.1109/TENCON.2018.8650517.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.

[4] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[6] P. Dong and W. Wang, "Better region proposals for pedestrian detection with R-CNN," 30th Anniv. Vis. Commun. Image Process., pp. 3–6, 2016, doi: 10.1109/VCIP.2016.7805452.

[7] W. Liu, D. Anguelov, D. Erhan, and C. Szegedy, "SSD: Single Shot MultiBox Detector," ECCV, vol. 1, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real- time object detection," IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[9] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[10] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv Prepr., 2018.

[11] S. Ding, F. Long, H. Fan, L. Liu, and Y. Wang, "A novel YOLOv3-tiny network for unmanned airship obstacle detection," IEEE 8th Data Driven Control Learn. Syst. Conf. DDCLS, pp. 277–281, 2019, doi: 10.1109/DDCLS.2019.8908875.

[12] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," IEEE CVPR, vol. 1, pp. 886–893, 2005, doi: 10.1109/CVPR.2005.177.

[13] C. Szegedy, W. Liu, Y. Jia, and P. Sermanet, "Going Deeper with Convolutions," CVPR, 2015, doi: 10.1108/978-1-78973-723-320191012.

[14] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," Int. J. Comput. Vis., vol. 104, no. 2, pp. 154–171, 2013, doi: 10.1007/s11263-013-0620-5.

[15] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," IEEE Trans. Neural Networks Learn. Syst., vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," ECCV, pp. 346–361, 2014, doi: 10.1023/B:KICA.0000038074.96200.69.

[17] R. Nabati and H. Qi, "RRPN : RADAR REGION PROPOSAL NETWORK FOR OBJECT DETECTION IN AUTONOMOUS VEHICLES," IEEE Int. Conf. Image Process., pp. 3093–3097, 2019.

[18] L. Jiao et al., "A Survey of Deep Learning-Based Object Detection," IEEE Access, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/access.2019.2939201.

[19] D. Wang, C. Li, S. Wen, X. Chang, S. Nepal, and Y. Xiang, "Daedalus: Breaking Non-Maximum Suppression in Object Detection via Adversarial Examples," arXiv Prepr., 2019.

[20] C. Ning, H. Zhou, Y. Song, and J. Tang, "Inception Single Shot MultiBox Detector for object detection," IEEE Int. Conf. Multimed. Expo Work. ICMEW, no. July, pp. 549–554, 2017, doi: 10.1109/ICMEW.2017.8026312.

[21] Z. Chen, R. Khemmar, B. Decoux, A. Atahouet, and J. Y. Ertaud, "Real time object detection, tracking, and distance and motion estimation based on deep learning: Application to smart mobility," 8th Int. Conf. Emerg. Secur. Technol. EST, pp. 1–6, 2019, doi: 10.1109/EST.2019.8806222.

[22] D. Xiao, F. Shan, Z. Li, B. T. Le, X. Liu, and X. Li, "A Target Detection Model Based on Improved Tiny-Yolov3 Under the Environment of Mining Truck," IEEE Access, vol. 7, pp. 123757–123764, 2019, doi: 10.1109/access.2019.2928603.

[23] Q. C. Mao, H. M. Sun, Y. B. Liu, and R. S. Jia, "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications," IEEE Access, vol. 7, pp. 133529–133538, 2019, doi: 10.1109/ACCESS.2019.2941547.

[24] W. Fang, L. Wang, and P. Ren, "Tinier-YOLO: A Real-time Object Detection Method for Constrained Environments," IEEE Access, vol. 8, pp. 1935–1944, 2019, doi10.1109/ACCESS.2019.2961959.

[25] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers," *IEEE Int. Conf. Big Data, Big Data*, pp. 2503–2510, 2019, doi: 10.1109/BigData.2018.8621865.

[26] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR*, vol. 2017-Janua, pp. 3296–3305, 2017, doi: 10.1109/CVPR.2017.351.

[27] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3," *1st Int. Conf. Unmanned Veh. Syst.*, pp. 1–6, 2019, doi: 10.1109/UVS.2019.8658300.