# Identifying Objects in Real-Time at the Lowest Framerate

Md. Mamun Hossain*, Md. Ashiqur Rahman, Humayra Ahmed

*Department of Computer Science and Engineering,*
*Bangladesh University of Business and Technology, Dhaka, Bangladesh*
**Corresponding Author*

***Abstract*: The practice of finding instances of semantic objects of a certain class, including people, cars, and traffic signs, in digital photos and videos is known as object identification or detection. Due to the development of high-resolution cameras and their widespread usage in everyday life, the detection is one of the most difficult and rapidly expanding study fields in computer science, particularly in computer vision. For automatic object recognition, several researchers have experimented with a variety of techniques, including image processing and computer vision. In this research, we employed a deep learning based framework YOLOv3 using Python, Tensorflow, and OpenCV to identify objects in real time. We do a number of tests using the COCO dataset to verify the effectiveness of the suggested strategy. The results of the experiments show that our suggested solution is resource and cost effective since it uses the fewest frames per second.**

***Index Terms*: Object recognition, Realtime, YOLOv3, Tensorflow, COCO dataset**

## I. INTRODUCTION

Over the past several years, object detection has had a significant impact on how the world has adapted to artificial intelligence. Real-time object detection is essential in autonomous systems that are Computer Vision (CV) capable. Its precision and speed are equally crucial for ensuring reliable functioning. Although object recognition for static images has been extensively investigated, real-time object detection presents a number of distinct difficulties, including motion blur produced on by moving objects, focusing issues, and real-time speed limitations for autonomous agents. Real-time object identification, however, also creates fresh opportunities that may be taken advantage of. The key findings is that when it comes to image scaling, accuracy and speed should not always be traded off. Our findings demonstrate that occasionally real-time object recognition accuracy is improved by downscaling the image to a lower resolution. Future autonomous agents like self-driving vehicles, drones, and robots need real-time object detection as a crucial building component for visual cognition. Therefore, it is crucial for the detectors to be quick and precise in order to construct systems with trustworthy performance. Real-time object detection is becoming more and more necessary, and automated feature analysis has generated a lot of interest in object recognition methods. The ability to identify moving objects in live streaming is essential for a range of computer vision topics that are now of interest. Deep learning-based object identification techniques including Region-based Convolutional Neural Networks (R-CNN), Spatial Pyramid Pooling Networks (SPPNet), Region-based Fully Convolutional Networks (R-FCN), and You Only Look Once (YOLO) outperform more established techniques. One of the quickest object detection techniques, YOLO has been enhanced since it was first presented, including YOLO-V1, YOLO-V2, and YOLO-V3. It has good real-time performance and high accuracy. In order to further deepen the network layer and achieve a breakthrough in accuracy, YOLO-V3 makes advantage of the residual structure. Running on the robust GPU platform has allowed YOLO and its upgrades to achieve excellent accuracy and quick speed. Tinier-YOLO is utilized to minimize the model size while obtaining increased detection accuracy and real-time performance, resulting in a more effective object identification model for confined situations that was derived from Tiny-YOLO-V3. In order to make the model smaller, Tinier-YOLO reduces the amount of model parameters. This seeks to extract certain object type info from image datasets. To address the issues with real-time object detection, the object detection model YOLO-V3 using Python, Tensorflow, and OpenCV for image processing and discussion of findings was presented. Darknet-53, a backbone also created by the YOLO founders Joseph Redmon and Ali Farhadi [1], is used by YOLOv3. Darknet-53 is more potent than Darknet-19 and more effective than rival backbones since it uses 53 convolutional layers as opposed to the preceding 19 layers (ResNet-101 or ResNet-152). Figure 1 shows the layerd architecture of YOLOv3.

When compared to other classification methods, the YOLO algorithm is quicker. Although the YOLO algorithm has localization flaws, it anticipates fewer false positives in the background.

These algorithms are not adequately validated with data sets that were randomly collected; instead, they were trained using academic data sets like ImageNet, COCO, and VOC. Images taken in a real setting typically include the following problems:

1. The photographs may be captured blurry due to the camera’s instability.
2. The photos may also not be sufficiently clear if the item is blocked.
3. The photographs might be of inferior quality due to inadequate lighting, overexposure, or low resolution.

The rest of the paper is organized as follows: SectionII provides an overview of existing works related to our methodologies. SectionIII discusses the proposed machine learning based YOLO framework. Next, SectionIV illustrates and analyzes the experimental result from our machine learning models. Finally, SectionV gives a brief conclusion and future research directions in this field of research.

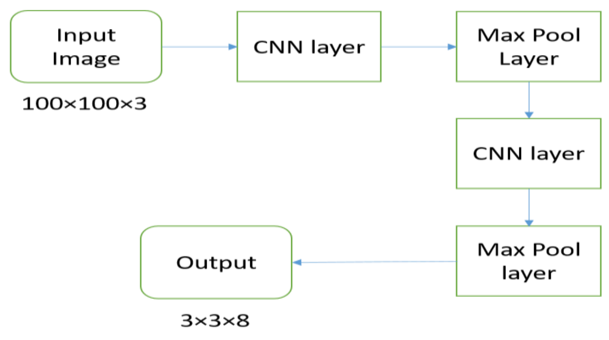


Fig. 1: YOLOv3 layer architectures

## II. RELATED WORK

YOLO is employed in many applications because of its higher object recognition performance and usability, however there are still issues with real-time video object detection that have been the subject of several research.

A real-time embedded pedestrian detection system was also suggested by Jin et al. [2]. The suggested method combined MobileNet with YOLO, and it was run on the Jetson TX2 machine. An enhanced YOLO network was suggested by Chen et al. [3]. To lessen the complexity of computing in embedded devices, the suggested technique quantified the network parameters and customized the YOLOv3-tiny network layer structure.

Additionally, research has been done on how to implement YOLO's object detection function in hardware with less resources. On non-GPU systems, Md. Bahar Ullah presented CPU Based YOLO [4], a real-time object detection paradigm. Sparse-YOLO [5], which is based on hardware/software co-design of an FPGA accelerator for YOLOv2, was proposed by Wang et al.

The You Only Look Once v2 (YOLOv2) technique is one of the unified pipeline framework-based approaches that researchers have presented in recent years [6]. Anchor boxes are used by YOLOv2 to forecast bounding boxes and enhance convergence and avoid overfitting, while batch normalization boosts recall. Other developments that improve detection accuracy include a high-resolution classifier, direct position prediction, dimension cluster, and multi-scale training. A shallow real-time detection method based on the YOLOv2 method was recently proposed by Pedoeem and Huang [7]. Their method reduces the size of the input image by half in order to speed up detection and eliminates batch normalization of shallow layers in order to reduce the number of model parameters.

Juan Du's Understanding of Object Detection Using CNN Family and YOLO. They generally discussed object detection families such CNN and R-CNN in this study, compared their efficacy, and presented the YOLO technique to improve it [8]. Matthew B. Blaschko's Learning to Localize Objects using Structured Output Regression. The topic of this essay is object localization. To get around the limitations of the sliding window approach, they adopted the bounding box method for object localisation in this [9].

The Fast YOLO approach, developed by Shafiee et al., uses an evolutionary deep intelligence framework to provide an efficient network architecture when applied to embedded devices [10]. The motion-adaption inference framework may utilize the optimized network architecture to accelerate the detection process and, as a result, lower the embedded device's energy consumption. In order to estimate multi-class 3D boxes in Cartesian space for detecting RGB pictures, Simon et al. devised the complex-YOLO approach [11]. The authors claim that this method significantly increases the speed of 3D object identification.

The Single Shot MultiBox Detector (SSD) approach [12] was created by Liu et al. to identify items of various sizes by creating multi-scale feature maps. This approach carefully balances detection speed and accuracy, however in the shallow layer, the feature map's expressive power is insufficient. Fu et al. have suggested the Deconvolutional Single Shot Detector (DSSD) approach [13] to increase the expression ability of shallow feature maps. This method utilizes a skip connection, a deconvolution layer, and the ResNet extraction network (which produces superior features) [14].

A novel SSD approach based on the feature pyramid has been presented by Qin et al. to increase the detection accuracy of the SSD method for tiny objects [15]. Their approach widens the convolution network to learn low-level location information while using a deconvolution network at the top of the feature pyramid to extract semantic information. To increase the accuracy of finding tiny items, their technology builds a multi-scale detection framework. In order to increase the detection accuracy for small objects, Redmon and Farhadi have suggested adopting the YOLOv3 approach for binary cross-entropy loss for class predictions [16], which uses scale prediction to forecast boxes at various sizes.

## III. METHODOLOGIES

### A. *YOLO Architecture*

Object identification is regarded as a regression problem by the YOLOv3 approach. With a single feed-forward convolution neural network, it directly predicts class probabilities and bounding box offsets from entire pictures. In order to create a real end-to-end detection system, it totally does away with feature resampling and region proposal creation and incorporates all phases into a single network. The input image is divided into $S \times S$ tiny grid cells via the YOLOv3 algorithm. The grid cell is in charge of detecting the object if the center of the object falls inside the grid cell. Each grid cell calculates the objectness scores related to the B bounding boxes and forecasts the location information of the boxes. These results may be achieved for each objectness score:

$$C_i^j = P_{i,j}(\text{ Object }) * IOU_{\text{pred}}^{\text{truth}}$$

whereby $C_i^j$ is the objectness score of the $j$ th bounding box in the $i$ th grid cell. $P_{i,j}$ ( Object ) is merely a function of the object. The $IOU_{\text{pred}}^{\text{truth}}$ reflects the anticipated box and the ground truth box's intersection over union (IOU). Binary cross-entropy of anticipated objectness scores and true objectness scores is one of the loss functions used by the YOLOv3 approach. It may be said in the following way:

$$E_1 = \sum_{i=0}^{S^2} \sum_{j=0}^{B} W_{ij}^{abj}[C_i^j \log(C_i^j) - (1 - \hat{C}_i^j)\log(1 - C_i^j)]$$

whereby $S^2$ is the number of grid cells of the image, and $B$ is the number of bounding boxes. The $C_i^j$ and $\hat{C}_i^j$ are, respectively, the projected abjectness score and the truth abjectness score. The position of each bounding box is based on four predictions: $t_x, t_y, t_w, t_h$, on the assumption that $(c_x, c_y)$ is the distance the grid cell is displaced from the image's top left corner. The top left corner of the image is displaced from the center of the final anticipated bounding boxes by $(b_x, b_y)$. Those are computed as follows:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$

whereby $\sigma()$ is a sigmoid function. The width and height of the predicted bounding box are calculated thus:

$$b_w = p_{ww} e^{t_w}$$
$$b_h = p_h e^{t_h}$$

whereby $p_{to}, p_h$ are the bounding box's preceding width and height. By using dimensional clustering, they are obtained. The ground truth box consists of four parameters $(g_x, g_y, g_w$ and $g_h)$, which correspond to the predicted parameters $b_x, b_y, t_w$ and $t_h$, respectively. Based on (3) and (4), the truth values of $\hat{t}_x, \hat{t}_y, \hat{t}_w$ and $\hat{t}_h$ can be obtained as follows:

$$\sigma(\hat{t}_x) = g_x - c_x$$
$$\sigma(\hat{t}_y) = g_y - c_y$$
$$\hat{t}_w = \log(g_w/p_w)$$
$$\hat{t}_h = \log(g_h/p_h)$$

The square error of coordinate prediction is one of the loss functions used by the YOLOv3 algorithm. It may be said in the following way:

$$E_2 = \sum_{i=0}^{S^2} \sum_{j=0}^{B} W_{ij}^{abj}\left[(\sigma(t_x)_i^j - \sigma(\hat{t}_x)_i^j)^2 + (\sigma(t_y)_i^j - \sigma(\hat{t}_y)_i^j\right.$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} W_{ij}^{obj}\left[((t_w)_i^j - (\hat{t}_w)_i^j)^2 + ((t_h)_i^j - (\hat{t}_h)_i^j)^2\right]$$

Initially, the YOLOv3 algorithm divides a picture into a grid. Each grid cell foretells the presence of a specific number of boundary boxes (also known as anchor boxes) around items that perform well in the aforementioned predetermined classifications. Only one item is detected by each border box, which has a corresponding confidence score indicating how correct it expects that prediction to be. The ground truth boxes' dimensions from the original dataset are clustered to identify the most typical sizes and shapes before being used to create the border boxes. YOLO is taught to do classification and bounding box regression simultaneously, unlike systems like R-CNN and Fast R-CNN. The diagrammatic representation figure 2, illustrates a solution model to a given problem.

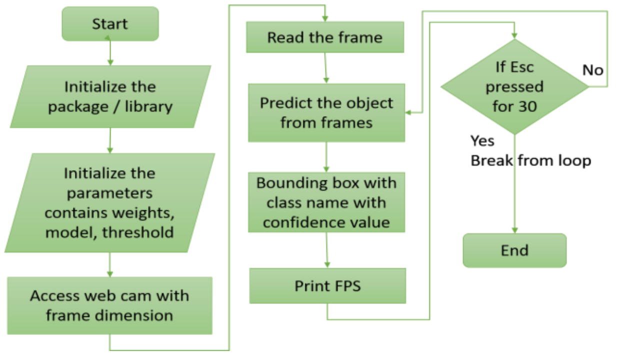

Start
Initialize the package / library
Initialize the parameters contains weights, model, threshold
Access web cam with frame dimension
Read the frame
Predict the object from frames
Bounding box with class name with confidence value
Print FPS
If Esc pressed for 30
No
Yes
Break from loop
End

Fig. 2: Flow diagram of the YOLO framework

## B. System Design

We get data for our system from the camera. Following the frame separation and picture pre processing, the input video data is then collected. Following the pre processing, a standard scale is measured and the data is entered into the YOLO framework. The COCO dataset is then compared to the marked object data, and an object score is calculated. The item is then recognized with the appropriate frame size after being categorised using class confidence. The result displays the detected item together with the appropriate confidence level and frame. The diagrammatic representation of the system is delineated in figure 3.

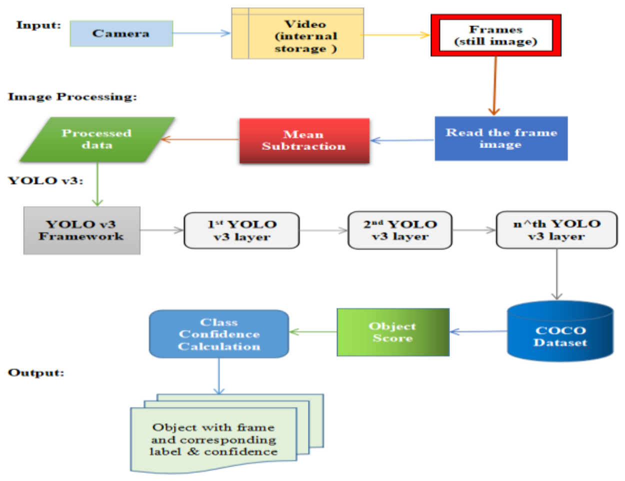
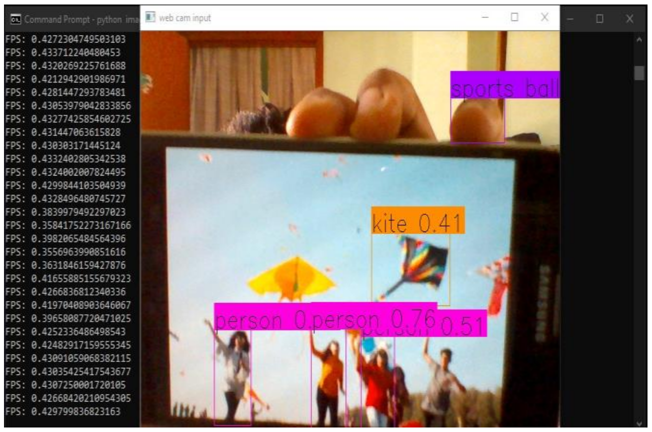


Fig. 3: System Diagram

## IV. EXPERIMENTAL RESULTS

The performance and analysis of the experiment were its most crucial components. This section details how our tests went, and the analytical data are also outlined in detail here. Through output figure 4 analysis and graphical comparison, the analytical result is displayed.

We can see in the image above that the system is identifying every conceivable object that is present in the input image. Additionally, the FPS (frames per second) is displayed on the image's side. Given that it is relatively low, the FPS is excellent. And our original strategy for real-world object detection with a low frame rate is successful. With the label frame and confidence level in the image above, an object has been correctly identified. 0.451 frames per second (FPS) is the average.



```
FPS: 0.4272304749503103
FPS: 0.433712240480453
FPS: 0.4320269225761688
FPS: 0.4212942901986971
FPS: 0.4281447293783481
FPS: 0.43053979042833856
FPS: 0.43277425854602725
FPS: 0.431447063615828
FPS: 0.430303171445124
FPS: 0.4332402805342538
FPS: 0.4324002007824495
FPS: 0.4299844103504939
FPS: 0.4328496480745727
FPS: 0.3839979492297023
FPS: 0.35841752273167166
FPS: 0.3982065484564396
FPS: 0.3556963990851616
FPS: 0.3631846159427876
FPS: 0.41655885155679323
FPS: 0.4266836812340336
FPS: 0.41970408903646067
FPS: 0.39658087720471025
FPS: 0.4252336486498543
FPS: 0.42482917159555345
FPS: 0.43091059068382115
FPS: 0.43035425417543677
FPS: 0.4307250001720105
FPS: 0.42668420210954305
FPS: 0.429799836823163
```

Fig. 4: Results obtained the camera view

Figure 5 is graphically displaying performance comparison for Yolo-v3, Inception, and Faster RCNN. We can observe that because it is quite high and far to the left, YOLOv3 is good. YOLOv3 operates far more quickly than competing detection techniques of equivalent capability.

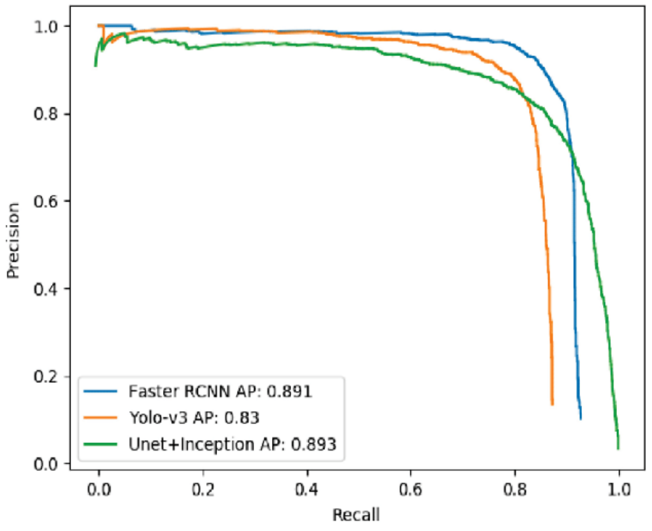


Fig. 5: comparison of Yolo-v3 and others

## IV. CONCLUSIONS AND FUTURE WORK

We propose a detailed investigation of the potential for YOLOv3 to enhance both speed and accuracy with a low frame rate in video object recognition given the significance of real-time object detection. The trained item produced good detection and tracking results, and this model may be used in other contexts to find, follow, and react to the targeted objects in the video surveillance. Increasing the scope of the investigation to look for weapons and ammunition to raise the alert in the event of a terrorist attack. Additionally, more effort is required to evaluate and speed up the performance of the current classifiers.

## REFERENCES

[1] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[2] J. Lee and K.-i. Hwang, "Yolo with adaptive frame control for real-time object detection applications," Multimedia Tools and Applications, pp. 1–22, 2021.

[3] S. Chen and W. Lin, "Embedded system real-time vehicle detection based on improved yolo network," 10 2019, pp. 1400–1403.

[4] M. B. Ullah, "Cpu based yolo: A real time object detection algorithm," 2020 IEEE Region 10 Symposium (TENSYMP), pp. 552–555, 2020.

[5] Z. Wang, K. Xu, S. Wu, L. Liu, L. Liu, and D. Wang, "Sparse-yolo: Hardware/software co-design of an fpga accelerator for yolov2," IEEE Access, vol. 8, pp. 116 569–116 585, 2020.

[6] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[7] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers," in 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018, pp. 2503–2510.

[8] J. Du, "Understanding of object detection based on cnn family and yolo," in Journal of Physics: Conference Series, vol. 1004, no. 1. IOP Publishing, 2018, p. 012029.

[9] M. B. Blaschko and C. H. Lampert, "Learning to localize objects with structured output regression," in European conference on computer vision. Springer, 2008, pp. 2–15.

[10] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, "Fast yolo: A fast you only look once system for real-time embedded object detection in video," arXiv preprint arXiv:1709.05943, 2017.

[11] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, and H. Michael Gross, "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in European conference on computer vision. Springer, 2016, pp. 21–37.

[13] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," arXiv preprint arXiv:1701.06659, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[15] P. Qin, C. Li, J. Chen, and R. Chai, "Research on improved algorithm of object detection based on feature pyramid," Multimedia Tools and Applications, vol. 78, no. 1, pp. 913–927, 2019.

[16] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement. 2018," arXiv preprint arXiv:1804.02767, vol. 20, 1804.