An Optimization to YOLOv3-tiny For Real-Time Detection of Small, Fast Moving Objects

Pavan Shiralagi[1], Rahul Bhandary[2], Rajeshwari B[3], Bajarangbali R[4]
[1,2,3,4]*Department of ECE, PES University, Bangalore, India*

***Abstract*— A project to obtain tennis statistics based on tennis ball tracking led us on a search to find real time object detection on small, fast moving objects. Realizing there were no methods available that satisfied our requirements, we optimized the detection method that came closest, YOLOv3 tiny [You Only Look Once Version 3 Tiny], to come up with YOLOOv3 tiny [You Only Look Once, Optimized]. YOLOv3 tiny is the third iteration of a Computer Vision application that is used to detect objects in real time. However, it is limited by the size and speed of the object relative to the camera's position along with the detection of False Positives due to incorrect localization. In this paper, we explore optimization techniques to extend the use of YOLOv3 tiny to accurately detect small, fast-moving objects. The techniques discussed in the paper were tested on detecting a tennis ball moving up to 160kmph with a minimum angular size of 0.414 degrees at a rate of 30 frames per second. The accuracy was found to be 95.268% on a video containing 4600 frames sampled at 30 frames per second (assuming the object is always in the frame) on a GeForce 1050 Graphics Processing Unit. This optimization includes a method of elimination of false positives to increase accuracy.**

***Keywords*— Real Time; Small Object; Fast Moving; False Positives; Optimization**

I. INTRODUCTION

Computer Vision has been one of the fastest developing fields in the recent past, with new methods for faster and more accurate object detection leading the growth. However, there is still much room for the improvement of detections on small and fast-moving objects relative to a camera. Many methods such as R-CNN [1], fast R-CNN [2], faster R-CNN [3], SSD [4] and YOLO [5] have been proposed for object detection. However, these methods struggle with the detection of small objects [6], primarily because of the smaller resolutions of the final layers of the neural network, which cause the features of the small object extracted at lower layers to become negligible at the final layers, leading to the problem of false positives where objects that may bare a slight resemblance to the object of interest are detected as the object of interest. Some of the methods proposed to detect smaller objects more accurately are: Multi-scale faster R-CNN [7] which uses feature combination to overcome the problem described above; data augmentation for small object detection [8] which oversamples the small object in the training images to improve accuracy; by adding high level features to low level features using deconvolutional fusion blocks and using a region context network [9]. An optimization has been proposed for RCNN to enable it to detect small objects more accurately [10]. Some of the other approaches that have been proposed are: Using convolutional auto encoders and symmetric skip connections to preserve features [11]; Using special region proposal networks [12] and a single pipeline that generates proposals and classifies them [13]. However, these networks are not capable of real time detection with an accuracy comparable to that of YOLOv3-tiny [14]. YOLOv3-tiny detects objects at multiple scales and uses residual blocks to improve detections of small objects. The architecture of YOLOv3-tiny is shown in Fig 1. YOLOv3-tiny is known to be the fastest object detection algorithm which pays for the increase in speed with a decrease in accuracy. It is a convolutional neural network consisting of nine convolutional layers and three fully connected layers.

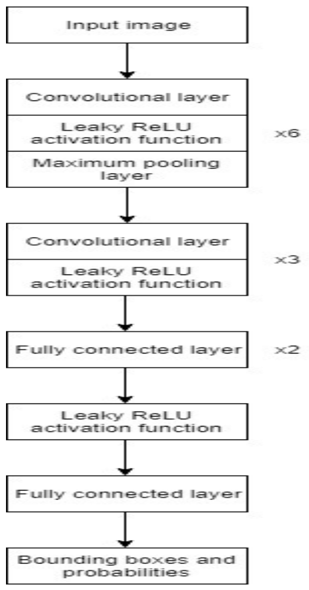
Input image
Convolutional layer
Leaky ReLU activation function
Maximum pooling layer
×6
Convolutional layer
Leaky ReLU activation function
×3
Fully connected layer
×2
Leaky ReLU activation function
Fully connected layer
Bounding boxes and probabilities

Fig 1: The YOLOv3-tiny architecture.

A convolution layer consists of a weights matrix (filter) which is convoluted with the image matrix to identify parameters. The values of the filter are modified through deep learning to allow detection of custom objects. Leaky ReLU [shown in Fig 2] is an activation function or a rectifier. It increases linearly and proportionally for positive values and decreases at a lower rate for negative values.

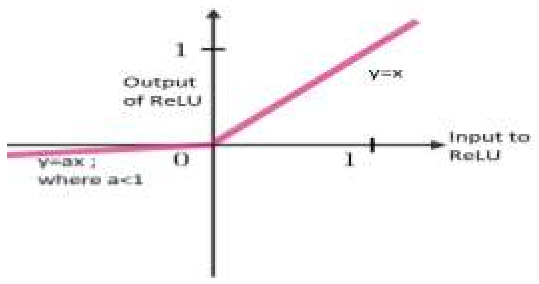

Fig 2: The Leaky ReLU activation function.

The Max Pool layer is a simple layer without any weights, consisting of set values which determines the largest value (prevalent features) in different parts of a matrix. This helps determine prevalent features and discard the rest to help the network focus. The last layers are fully connected. This means that each neuron of the output of a layer is connected to each input neuron in the next layer.

Since YOLOv3-tiny can detect objects in images in real time, it can be used for a wide variety of applications. However, its accuracy is still low when it comes to detection of small objects when there is less contrast between the object of interest and its background. In this paper, we propose a method to detect and track small objects in real time video feed using various optimization techniques. These techniques are implemented over three modes: Initial splitting; False Positive Removal and Object Following. The flow of the proposed algorithm is shown in Fig 3. The method discussed here does not require extreme computation power. Our detection method works in real time (30 FPS) on a Nvidia GeForce 1050 GPU.

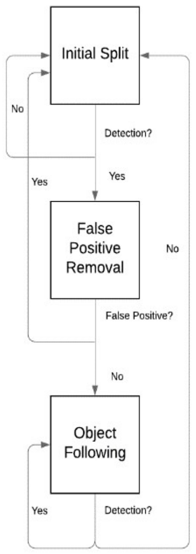

Fig 3: Flow chart depicting our various optimization modes for detection of small, fast moving objects in an image.

II.THE OPTIMIZATION

A. Terminologies

The terms used in the paper are elaborated in Table I.

TABLE I

TERMS USED IN THE PAPER

Term	Meaning
fpsv	Frames per second of the video
fpsp	Number of frames processed in one second
initial_split	Number of parts each image is divided in order to get the first detection
S	Area occupied by the object in pixel^2 units
scaling_factor	Number of metres/pixels
max_speed	Maximum speed of the object of interest in m/s
FS	Frame Skip Number
l	Length of the object of interest in pixels
b	Breadth of the object of interest in pixels
p	Padding around the detection for false positive removal

B. Initial Split

To obtain the first detection, we split each frame into an optimal number of sub-frames [Fig 4], to maintain increased accuracy while abiding to real time constraints. Our object detection network then runs on each subframe independently. If a detection is not obtained, the same procedure is repeated on the next frame provided to our network. If a detection is obtained, False Positive Removal mode is activated. None of the sub-frames or frames of the video are being saved after detection, this saves memory and drastically speeds up the system.

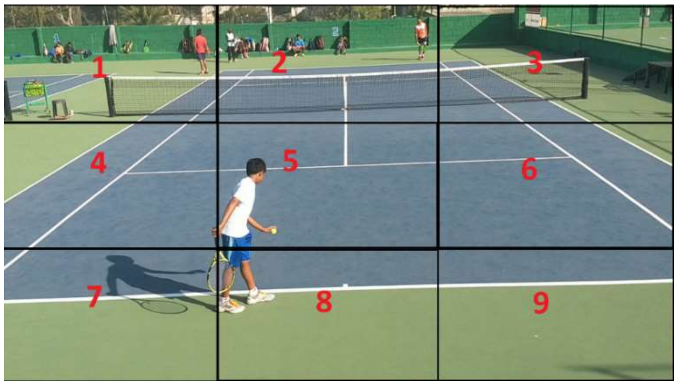

Fig 4: Splitting of a frame into multiple subparts to increase accuracy of initial detection, Initial split in this case is nine.

Cropping into each frame helps remove unnecessary context and allows the network to focus on what’s important. This significantly increases object detection accuracy before we know where the object is. However, splitting the frames into sub-frames and running detection on each frame significantly increases computation time. To ensure a balance is maintained between accuracy and processing time, calculations for the Initial Split value is carried out to find minimum initial split for maximum accuracy and maximum initial split to ensure real-time application. Picking values for Initial Split will determine the trade-off between accuracy and time. The ideal value for the initial split to ensure real-time operation without frame skip can be found using Eq. (1).

$$time\_initial\_split = max(fpsp - fpsv, 1) \qquad (1)$$

When accuracy is the primary factor, the ideal value for Initial split can be found using Eq. (2).

$$\text{initial\_split} = \frac{240 * \text{area\_of\_frame} * (\%\ of\ accuracy\ desired)}{area\ occupied\ by\ the\ object\ in\ the\ frame} \qquad (2)$$

When a higher Initial Split value is required to maintain accuracy, we use a method of Frame Skipping to ensure real-time application.The ideal value for frameskip [assuming the object is always in frame] can be calculated using Eq. (3).

$$FS = \left(\left(\frac{accuracy\_initial\_split + fpsv - 1}{fpsp}\right) - 1\right) * fpsv \qquad (3)$$

For example, assume fpsv=30 fps, fpsp=16 fps and accuracy_initial_split=6 in Eq. (3). The value of FS in this case will be 33 frames. This implies that 33 frames must be skipped after detection on every 30 frames in order to maintain real time operation.

C. False Positive Removal

As soon as there is a detection in the Initial Split mode, our network crops into the detection (45x magnification [in the case of a tennis ball] with respect to the original frame) and detects on the object again. This mode is activated only post the initial detection and removes a very large percentage (60%) of false positives since the features of the object in the cropped portion around the detection is more likely to resemble the features learnt by the YOLOv3-tiny algorithm during training than the features of false positives. If the detection is found to be a false positive, the system goes back to Initial Split mode. If the detection is a true positive, the system goes into Object Tracking mode. The increase in crop again allows us to remove all unnecessary context. However, cropping the entire frame into many sub-frames at this level of cropping is not possible, hence we only crop into the area where probability of detection is high based on the detection in Initial Split. False positives are found by running detections in an area around the detection obtained. This area is obtained by zooming into the detection with a magnification as given by Eq. (4) with respect to the original frame. A padding of a few pixels around the detection is necessary to reduce distortion. In Eq. (4), p is the padding around the object in pixels.

$$Magnification = \frac{Area\ of\ the\ frame\ in\ pixel^2}{S + 2p(l + b) + 4p^2} \qquad (4)$$

D. Object Tracking

The Object Tracking method is used to continuously track the small, fast-moving object post FPR. Based on the maximum and minimum speeds of the object relative to the camera, only the required area of the frame is analysed by shifting the window of detection. This pixel value can be calculated for any object based on the size and speed of the object relative to the camera. This allows us to remove a lot of context that is not required in the image and allows our network to focus on the important part of the image. This method basically allows objects of smaller dimensions to be viewed as larger objects. The size of the object is directly proportional to the size of the detection window, hence no matter what the actual dimension of the object is, all objects will have equal size relative to the frame size while in Object Tracking mode. The formula to calculate the number of pixels around the previous detection on four sides [left, top, right and bottom] [detection window] for object following is given in Eq. (5). The size of the detection window determines the amount of context taken into consideration for the next detection. An illustration of the window of detection for a tennis ball is given in Fig 5.

$$detection\_window = \frac{\text{max\_speed}}{\text{fpsv} * \text{scaling\_factor}} \qquad (5)$$

In Eq. (5), max_speed/fpsv is the maximum distance in meters that can be travelled by the object of interest in one frame. The scaling factor converts this distance in m to distance in pixels, and this represents the maximum possible displacement of the object in one frame.

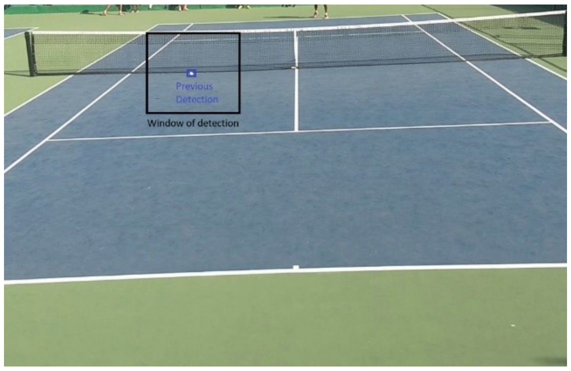

Fig 5: An illustration of the window of detection for small object following.

E. Training

To train YOLOOv3-tiny, the images in the training set are a mixture of cropped parts of the frame both with and without the object of interest, as shown in Figs. 6 and 7.

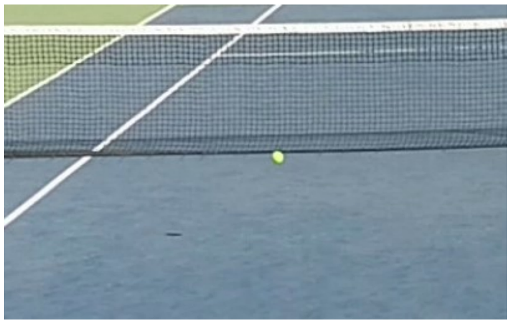

Fig 6: An image from the training set with a ball.

Fig 7: An image from the training set without a ball.

The area of each training image is given by Eq. (6).

$$Area\ of\ each\ training\ image = \frac{Area\ of\ the\ entire\ image\ in\ pixel^2}{initial\_split} \quad (6)$$

Images without a ball are added in accordance with the training guidelines given by the authors of the YOLOv3 paper [15]. The actual frame from which these images are taken is shown in Figure 8.

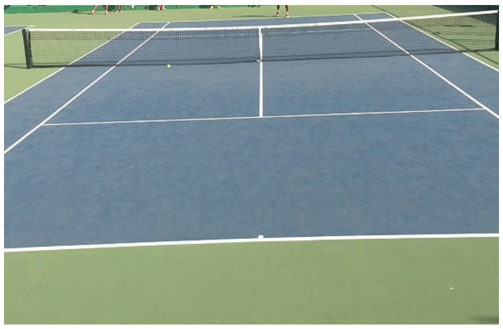

Fig 8: The image from which Figs 6 and 7 were obtained.

F. Detections

The overall method for obtaining detections through our algorithm is shown in Fig 3.

1) In order to detect the position of the small object of interest in the image, we first divide the image into several parts, as shown in Fig 4.
2) If there is a detection in any of the parts of the image, we perform a test for false positives by running the YOLOv3-tiny algorithm on the part of the image around the detection. If there is a detection in this part of the image, the detection is assumed to be a true positive and the algorithm proceeds into the next step. If not, the original detection is a false positive and the algorithm goes back into step (1). If there are no detections in any part of the image, the algorithm goes back to step (1).
3) The next step of the algorithm is object following. In this step, the coordinates of the detection are obtained from step (2) and the window of detection ismoved accordingly to find the position of the small object in the next frame. If the object is found in the window of detection, small object following continues with the new coordinates as reference. If the object is not found in the window of detection, the algorithm goes back to step (1).

III. RESULTS AND ANALYSIS

Fig9 shows the working of the algorithm when the object of interest is a tennis ball. Tables II, III and IV display the various accuracies based on training and testing in different lighting conditions.

The time taken for detection on each frame was 20 ms [50 fps] on a GTX 1050 GPU. Assuming an initial split value of 6, our method uses 700 ms [120 ms for the first frame and 20 ms for each subsequent frame] in the worst case for processing one second of a 30 fps video [assuming the object is always in frame].

TABLE II

ACCURACIES OBTAINED ON TRAINING IN SUNNY CONDITIONS AND TESTING IN DIFFERENT LIGHTING CONDITIONS.

	Training in sunny conditions	
	Testing in sunny conditions	**Testing in overcast conditions**
YOLOv3-tiny	5%	2%
YOLOOv3-tiny	95%	15%

TABLE III

ACCURACIES OBTAINED ON TRAINING IN SUNNY CONDITIONS AND TESTING IN DIFFERENT LIGHTING CONDITIONS.

	Training in overcast conditions	
	Testing in sunny conditions	**Testing in overcast conditions**
YOLOv3-tiny	5%	60%
YOLOOv3-tiny	12%	96%

TABLE IV

ACCURACIES OBTAINED ON TRAINING IN SUNNY AND OVERCAST CONDITIONS AND TESTING IN DIFFERENT LIGHTING CONDITIONS.

	Training in sunny conditions	
	Testing in sunny conditions	**Testing in overcast conditions**
YOLOv3-tiny	5%	2%
YOLOOv3-tiny	75%	68%

Our method can be used in real time applications. It is seen that our optimization provides a significant increase in accuracy over YOLOv3 – tiny in various lighting conditions.

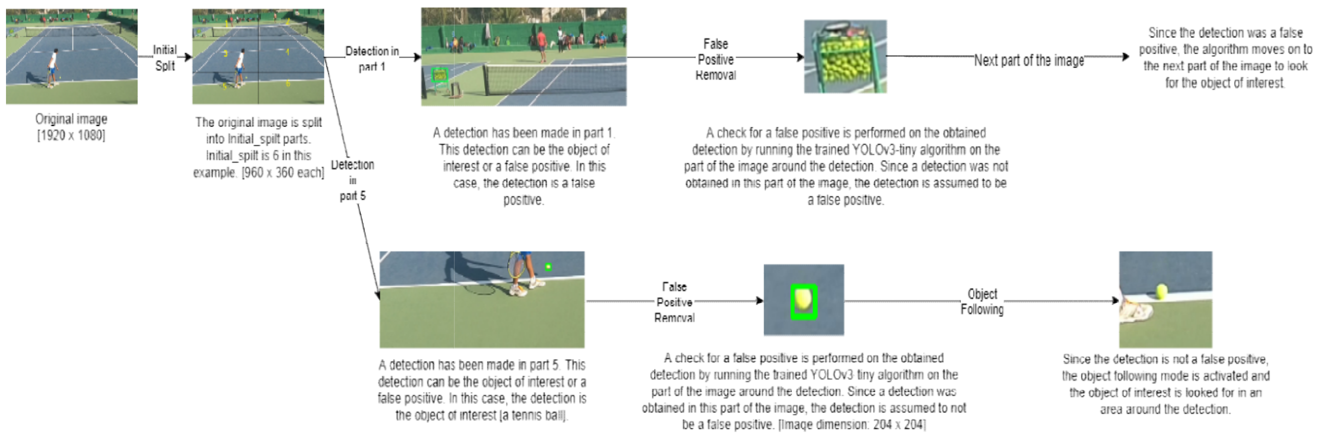
Original image
[1920 x 1080]
Initial Split
The original image is split into Initial_spilt parts. Initial_spilt is 6 in this example. [960 x 360 each]
Detection in part 1
Detection in part 5
A detection has been made in part 1. This detection can be the object of interest or a false positive. In this case, the detection is a false positive.
False Positive Removal
A check for a false positive is performed on the obtained detection by running the trained YOLOv3-tiny algorithm on the part of the image around the detection. Since a detection was not obtained in this part of the image, the detection is assumed to be a false positive.
Next part of the image
Since the detection was a false positive, the algorithm moves on to the next part of the image to look for the object of interest.
A detection has been made in part 5. This detection can be the object of interest or a false positive. In this case, the detection is the object of interest [a tennis ball].
False Positive Removal
A check for a false positive is performed on the obtained detection by running the trained YOLOv3 tiny algorithm on the part of the image around the detection. Since a detection was obtained in this part of the image, the detection is assumed to not be a false positive. [Image dimension: 204 x 204]
Object Following
Since the detection is not a false positive, the object following mode is activated and the object of interest is looked for in an area around the detection.

Fig 9: An example of the working of the algorithm when the object of interest is a tennis ball.

The detection accuracy can be greatly improved if the lighting conditions of each frame can be pre-determined.

The confusion matrix for YOLOOv3 - tiny run on a set of 4600 test images with a tennis ball as the object of interest is given in Table V.

This data proves that our optimization has the scope to be used in various industries from healthcare to defence to detect various small, fast moving objects in real time with high accuracy. (95.268%)

IV.PARAMETERS REQUIRED

YOLOOv3 – tiny requires two parameters to be known for usage.

1) The objects maximum speed, which we use to calculate the window shifting value.
2) The objects approximate size, which is used to determine the magnification for detections.

TABLE V

THE CONFUSION MATRIX OBTAINED ON TESTING THE PROPOSED ALGORITHM ON A 4600 FRAME VIDEO WHEN THE OBJECT OF INTEREST WAS A TENNIS BALL.

		Actual		
		Positive	**Negative**	**Total**
Predicted	**Positive**	919	26	945
	Negative	240	3415	3655

V. CONCLUSION

In this paper, we have introduced YOLOOv3 - tiny, an optimized real-time detector for small, fast-moving objects. YOLOOv3 - tiny is the only object detection method to detect small, fast moving objects in real-time to the best of our knowledge. YOLOOv3 - tiny can be used in diverse fields including sports technology, defence and any other domain which requires tracking of extremely small objects in real time. Our method of detecting and tracking small objects achieves an accuracy of 95.268% on a video containing 4600 frames sampled at 30 frames per second.

ACKNOWLEDGMENTS

We are grateful to our team at ESML Sports Pvt Ltd for assisting us in carrying out this research. We are also thankful to our mentor, Prof Rajeshwari [Associate Professor at PES University] for providing us with valuable insights throughout the research effort.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR
[2] R. Girshick. (2015). Fast R-CNN. In ICCV.
[3] S. Ren, K. He, R. Girshick, and J. Sun. (2015). Faster RCNN: Towards real-time object detection with region proposal networks. In NIPS.
[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. (2015). SSD: single shot multibox detector. In CoRR, abs/1512.02325.
[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. (2015). You only look once: Unified, real-time object detection. In arXiv, preprint arXiv:1506.02640.
[6] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. (2016). Speed/accuracy trade-offs for modern convolutional object detectors. In arXiv: 1611.10012.
[7] Guo X. Hu, Zhong Yang, Lei Hu, Li Huang, and Jia M. Han. (2018). Small Object Detection with Multiscale Features. In International Journal of Digital Multimedia Broadcasting, vol. 2018, Article ID 4546896, 10 pages. https://doi.org/10.1155/2018/4546896.
[8] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, Kyunghyun Cho.Augmentation for small object detection. In arXiv: 1902.07296v1 [cs.CV].
[9] Mingliang Xu, Lisha Cui, Pei Lv, Xiaoheng Jiang, Jianwei Niu, Bing Zhou, Meng Wang. MDSSD: Multi-scale Deconvolutional Single Shot Detector for Small Objects. In arXiv:1805.07009v2 [cs.CV].
[10] C. Chen, M.Y. Liu, O. Tuzel, and J. Xiao. (2017). R-CNN for small object detection. In 13th ACCV Proceedings.
[11] X.J. Mao, C. Shen, and Y.-B. Yang. (2016). Image restoration using convolutional auto-encoders with symmetric skip connections. In arXiv:1606.08921.

[12] C. Eggert, D. Zecha, S. Brehm, and R. Lienhart. (2017). Improving small object proposals for company logo detection. In ICMR.
[13] H. Krishna and C. V. Jawahar. (2017). Improving Small Object Detection. 4th IAPR Asian Conference on Pattern Recognition (ACPR), Nanjing, 2017. pp. 340-345. doi: 10.1109/ACPR.2017.149
[14] J. Redmon and A. Farhadi. (2018). Yolov3: An incremental improvement. In arXiv preprint arXiv:1804.02767.
[15] https://github.com/AlexeyAB/darknet#how-toimprove-object-detection