Object Detection in Real-Time Systems: Going Beyond Precision

Anupam Sobti[1] Chetan Arora[2] M. Balakrishnan[1]
[1]IIT Delhi [2]IIIT Delhi

Abstract

Applications like autonomous driving, industrial robotics, surveillance, and wearable assistive technology rely on object detectors as an integral part of the system. Thus, an increase in performance of object detectors directly affects the quality of such systems. In the recent years, convolutional neural networks (CNNs) and its variants emerged as the state of art in object detection, where performance is usually measured either in terms of mean average precision (mAP) or number of frames processed per second (fps). Many applications which use object detectors are resource constrained in practice. Even though it is clear from the published results, that a frame-level analysis of the system in terms of mAP or fps proves the superiority of one algorithm over the other, we observe that such metrics do not necessarily apply to real time applications with resource constraints. A slower algorithm even though highly accurate may need to drop frames to maintain the necessary frame rate and lose on the accuracy. We propose a closer look at the metrics used for performance in real-time applications, and suggest some new evaluation criterion. Our comparison of state of the art detectors on these metrics has also thrown some surprises in terms of conventional wisdom, which we present in this paper. Our framework is available at https://www.github.com/anupamsobti/object-detection-real-time-systems.

1. Introduction

Object Detection is the task of localizing and classifying objects in a given image. Applications such as assisted and autonomous driving [9], navigation aids for visually impaired [21], and robotics applications [13] use object detector module as an integral step for motion planning, landmark detection etc. The efficacy of these detectors on a set of images is typically measured using conventional metrics like precision and recall. For evaluating performance on video input, True positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) determined using the Jaccard index criteria are simply accumulated over all frames of the video. Jaccard index, more commonly known as IoU is the intersection over union of the bounding box predicted using the object detector under consideration with that given in the ground truth. A detection is considered as positive if the Jaccard Index is greater than a threshold. Precision-Recall curves are then plotted for different values of the threshold of the detector and Mean Average Precision (mAP) is often used as a measure of how well the detector performs.

(a) A slower running detector which has a higher accuracy has to skip frames in order to maintain real-time operation. When too many frames have to be skipped to keep up with the real-time performance, a lot of pedestrians go undetected resulting in a poor performance on the application level.

(b) A fast running detector with lower accuracy is able to run detections on all frames but unable to detect all pedestrians in the frame. Nevertheless, it may still outperform a slower detector.

Figure 1: There are a number of factors which affect the performance of detectors under different constraints. We discuss some of these constraints and situations. We also suggest an evaluation criterion which can capture this information and take the evaluation of object detectors closer to the application performance.

However, in an end to end application, with a live stream being captured at a constant rate, a simple accumulation over frames makes little sense. Figure 1 shows how an object detector which is better in terms of conventional accuracy might perform relatively poorly on the application level. If an algorithm is able to detect a certain object, say a person in one frame of the video, one can simply track the person in the following frames using simple object trackers.

So, the detection performance in the consecutive frames becomes redundant. It also makes sense to reward early detections since the detections thereafter can be obtained relatively easily using trackers. By filtering the bounding boxes using a spatio-temporal correlation (in successive frames), false positives could be reduced [10]. However, conventional precision-recall metrics do not allow such heuristics to be evaluated for an algorithm, when acting as a module in an end to end application pipeline.

In various applications like autonomous driving, industrial or domestic robotics, navigation systems, it is often important that the objects in the view are detected before a specified minimum distance and the information is presented to the user/system within a time limit such that appropriate decisions can be made. *E.g*., in an autonomous driving scenario, a pedestrian should be detected at a distance far enough, so that brakes could be applied in case the pedestrian is in the driving path and the car can come to a halt or find an alternate path avoiding the pedestrian. Similarly, a visually impaired person walking towards a stray dog, in an assistive system like MAVI [21], should be alerted in advance before actually disturbing the dog.

The notion of a decision distance has also been introduced by Fregin *et al*. in [12], which is the minimum distance before which the system must detect the state of a traffic light with sufficient time to halt the car. Using this insight, we propose the number of objects detected before a specified minimum distance as a metric for applicability to real-time systems. We present results for the task of pedestrian detection in this paper. The analysis is done on complete video sequences. The analysis method can however, be extended to other object classes as well.

The major contributions of the paper are as follows:

- We suggest a new evaluation criterion to predict utility of an object detection algorithm in an end to end application. The complete system, with an object prediction module, can therefore be analyzed as a black box along with integration of other heuristics.

- The real time performance requirement with a computation budget can impede the accuracy of an object detection system because of frame dropping. We identify the hardware resources/power budget required to achieve certain accuracy in such applications. Conversely, we find the best performing class of detectors/systems given the constraints of hardware resources/power budget along with real time performance requirement.

2. Related Work

In this section, we describe the work which relates to the analysis of object detectors in the bigger picture of system-level integration. Therefore, we discuss three major areas in which the related work has had a direct impact on computer vision applications, namely Object Detection, Tracking and the metrics which have been used to evaluate these.

Object Detection: Object detection has been widely explored in challenges like COCO Detection Challenge [22], ILSVRC [2], and Pascal VOC [11]. In recent years, the use of deep networks for the task has led to major strides in the frame-level accuracy. However, at $\sim 95\%$ recall, state of the art object detectors still make about $10\times$ more errors than human baseline [26]. With deep networks like Inception-v4 [25] having hundreds of convolutional layers, one of the major disadvantages of the newer techniques with respect to traditional algorithms have been the significantly higher amount of memory and compute power needed. Researchers have noticed this and suggested various techniques to reduce the hardware requirement using model compression methods like quantization, hashing *etc*. or by using entirely different architectures [17, 15].

Howard *et al*. [15] have proposed a new streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks. They have introduced new simple global hyper-parameters that efficiently trade off between latency and accuracy and allow the model builder to choose the right sized model for their application based on the constraints of the problem. Tiny YOLO [5] is another effort for creating smaller models with a tradeoff in accuracy. Distillation [14] has been proposed as a way to better train a smaller network using a pre-trained deeper and more accurate network.

Tracking: Given the performance of current state of the art in object detection, it is common for system designers to also use tracking to improve the performance of the overall application. This can be done in two ways:

- Appearance/feature based tracking

- Tracking by detection

The appearance/feature based trackers can be used to reduce the number of frames to be processed by an object detector. *E.g*., one could run object detector on every tenth frame and use trackers to find object location for the intermediate frames. Since, run time for the trackers are typically much less than object detectors, this can lead to many fold increase in the runtime speed. However, no new detections can be made during intermediate frames which can adversely effect the accuracy. Methods like Lukas-Kanade [6], Track-Learn-Detect (TLD) [20] tracker and other methods like Medianflow [19] are examples of such trackers. A recent survey of tracking techniques using newer object detectors can be found in [24].

Evaluation Metrics: Zhang *et al*. [26] analyze the gaps for improvement of the frame-level accuracy as the evaluation metric. They propose to evaluate quality of detections using a miss rate Vs false positives per image (FPPI) criteria.

The lack of more appropriate metrics has led to non-uniform evaluation of different techniques, as observed by Buch *et al*. [8]. For the problem of traffic light monitoring, a similar observation has been made by Jensen *et al*. in [18]. Fregin *et al*. [12] have suggested more appropriate metrics for the problem which takes the evaluation measure closer to application performance. A need for better metrics has also been expressed by Bengler *et al*. [7]. Recent work by Huang *et al*. [16] compares the detection techniques on speed, accuracy and memory footprint.

Our work goes one step further and tries to answer the following questions:

- Given a resource constrained platform (say constrained by energy), which detector/algorithm is best for achieving the maximum application performance?
- How do we estimate the resources required when a certain application performance is desired?

This analysis intends to facilitate the system designers to choose the best algorithms/detectors within their constraints as well as provide an intuition to object detector designers regarding the direction in which efforts are required to improve system level application performance. To the best of our knowledge, this is the first work addressing this problem.

3. Proposed Approach

As described using Figure 1, a better detector need not detect objects in all the frames accurately. Instead a detector which is able to detect most people from a large distance is perhaps better suited from an overall system perspective. Therefore, we propose the number of people detected before a specified minimum distance as a measure of how well the detector is expected to perform in an application. Based on this assumption, we evaluate how different detectors perform in two scenarios:

- Infinite resource setting (IRS)
- Resource-constrained setting (RCS)

Before going into the details of these settings, we discuss the details of the dataset used.

3.1. Dataset

We use sequences MOT-02, MOT-05, MOT-09, MOT-10 and MOT-11 from MOT16 training dataset and ETH Bahnhof from MOT15 training dataset. All these datasets are part of Multi-object Tracking Challenge as described in [24]. Another specific observation about these datasets is that the recording equipment has very limited vertical motion. Therefore, the world distance of a pixel at a specific row number has little variation for the entire duration of the video. This enables us to use row number as the measure of distance of a point from the camera. The approach is generalizable for a limited cases only and alternate techniques like annotated depths/radial distance approximation may be required in other scenarios. Since, pixel row numbers go from top to bottom, therefore, the value is inversely related to the distance. See Figure 2 for a visual explanation. To maintain uniformity of the evaluation criteria, we resize all images to VGA (640×480) resolution and downsample the videos to 15 FPS. Each frame in the video is annotated with bounding boxes as well as an ID number for all pedestrians. Note that a new ID is assigned if a person is occluded in some intermediate frame and re-appears in a later frame. This is the convention followed for the MOT Challenge [24]. More details about the dataset are provided in Table 1.

Dataset	FPS	Length (no. of frames)	Resolution
ETH-Bahnhof	15	1000	640×480
MOT16-02	30	600	1920×1080
MOT16-05	15	837	640×480
MOT16-09	30	525	1920×1080
MOT16-10	30	654	1920×1080
MOT16-11	30	900	1920×1080

Table 1: Details of sequences used from Multi-object Tracking Challenge [24].

For a better insight into dynamics of these sequences, we compute a histogram of the amount of time/frames for which a person is visible in the videos. Figures 3 and 4 show the histograms. Here, a video in which people (or the camera or both) are moving fast will have a very low mean time of presence and a video in which people are mostly stationary will have a high mean time of presence. It is important to understand that a low mean time of presence imposes a bigger challenge for the object detector since skipping a few frames during processing of the video feed may result in missing a lot of objects which came during the skipped time interval. We also quantify this in Section 5.1. The mean time of presence (as indicated in the figures 4 and 3), is therefore inversely related to the entropy in the video. The entropy here refers to the level of activity in the video.

3.2. Infinite resource setting

We define the speed at which the detector is able to process the incoming stream (with frame rate measured in

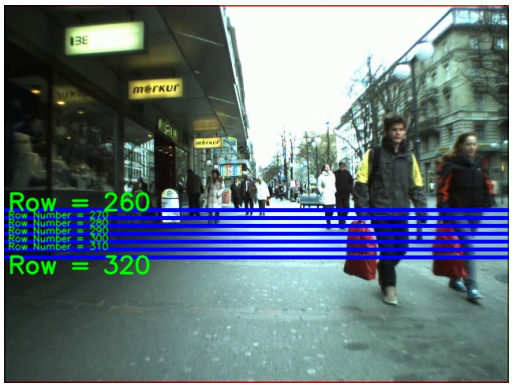


Figure 2: We define row numbers as the pixel distances which are indicative of the distance from the camera. We pick the indicated distances to highlight the area of interest where we see maximum pedestrian activity.

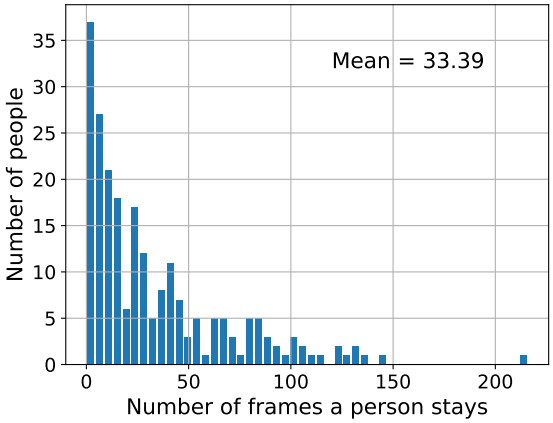


Figure 3: Histogram of the number of frames for which people stay in the ETH Bahnhof dataset video. The high density around low times suggests that the people in the video move rather quickly.

frames per second) as the *Frame Processing Rate* (FPR) of the particular detector. In Infinite Resource Setting (IRS), we are trying to establish the limit of various detectors. If there are no constraints on the hardware resources, one could use multiprocessing/multithreading techniques to run a detector at a desired FPR. If one would like to reduce the latency, then FPGA based accelerators/ASICs (Application Specific Integrated Circuit) could also be designed. Therefore, we analyze the various detectors here independent of their FPR.

In our experiments, True Positives (TP) are defined as the number of people who are uniquely identified in the complete video. A detection proposed by the object detector is matched against all the annotated bounding boxes

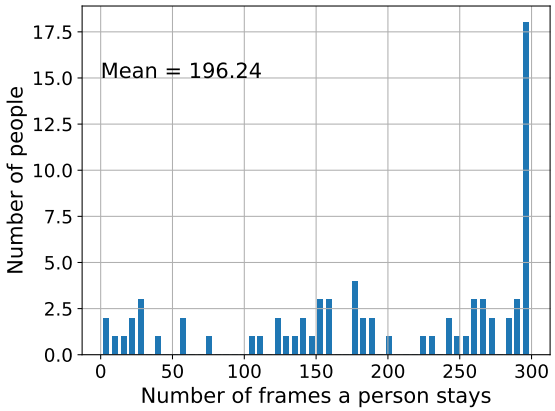


Figure 4: Histogram of the number of frames for which people stay in the video for MOT16-02 dataset. The high density around larger time suggests that some people are present in the video frame for a long time.

(ground truth) for that video frame. If the maximum overlap (Jaccard criteria) for the match is greater than 0.3, the ID corresponding to the annotated bounding box is marked detected and the pixel distance of the bottom of the bounding box is noted as the distance of detection. Otherwise, the proposed detection is marked as a false positive. False negatives are defined as the number of people for whom an ID was present but were not reported in any of the detections. False positives are still defined in the traditional sense, where all detections which do not correspond to a bounding box in the annotated data are accumulated over all frames.

In IRS, we measure the true positives/false positives for different speeds of detector implementations (FPR) at different pixel distances. Results from this analysis are presented in Section 5.2.

3.3. Resource-constrained Setting

In Resource-constrained Setting (RCS), a detector is only able to process a limited number of frames. A detector which runs at 5 FPR on a single CPU system can only process 1 in 3 frames for a video recorded at 15 FPS. We use the same matching technique as specified in last section (Section 3.2) for the IRS. In RCS evaluation, detectors with high accuracies may end up missing a lot of frames owing to an FPR lesser than the video sampling rate, resulting in a drop in the number of people identified.

Alternatively, one could buffer a few frames and then process them in an offline manner. However, with such a system, the latency increases with more and more frames in the buffer and real-time operation is not possible. RCS represents the practical scenario a system designer faces when a particular application needs to be realized, given the con-

straints of cost, energy and performance. Our results for this setting in Section 5.3 provide interesting insights for system designers of computer vision applications as well as a method for object detection researchers to work on detectors which have maximum effect on the application performance.

4. Experimental Setup

Efforts like Google Object Detection API[1], Darknet[23] have enabled easy to use object detection frameworks. Cross-platform support from Tensorflow [4] has made this analysis much easier. OpenCV [3] has been of great assistance to the vision community in providing cross-platform support for implementation of numerous vision algorithms. We use Tensorflow and OpenCV on four configurations: CPU + GPU, High-end CPU, Low-end CPU and an ARM processor for the resource-constrained setting described in Section 3.3. Details of the configurations of these systems is provided in Table 2. We use the pre-trained models for neural network based detectors from the Tensorflow Model Zoo [1]. The models used are shown in Table 3. We also include the pre-trained HOG implementation from OpenCV in our comparison. Mean average precision (mAP) is calculated using the frame-based metrics described in Section 1.

For the Infinite Resource Setting described in Section 3.2, we generate the detections offline for all frames of the videos. Since the video is sampled at 15 FPS, the numbers for true positives, false positives *etc*., are generated by skipping enough frames to reach the required FPR. *E.g*. if the processing speed is at 15 FPR, all frames are processed. Similarly, every third frame is processed for an FPR of 5. For RCS also, the FPR of an object detector implementation is estimated by processing the videos and calculating the average time per frame. Thereafter, the accuracy numbers are estimated in the same fashion as IRS, *i.e*., by sub-sampling according to the FPR to be achieved.

5. Results

We present results in three domains. In Section 5.1, we show the correlation of entropy (level of activity) of a video to the FPR required to achieve a particular accuracy for the same. In Sections 5.2 and 5.3, we discuss the results for IRS and RCS settings described in Section 3. The videos are downsampled to 15 FPS and resized to 640×480 for maintaining uniformity. Note that, we skip frames in order to downsample the video instead of just slowing it down because we want to maintain the original dynamics of the video, which essentially captures how fast/slow the pedestrians are moving in real time as seen in Section 3.1. Therefore, one second of the video corresponds to one second of wall clock time. The histogram of number of frames for which people stay in the video for all the datasets together is shown in Figure 5. It can also be observed that the tail of the distribution starts from around 100. Therefore, very few people stay in the video for more than 100 frames. This implies that if the detector has to skip more than 100 frames to maintain real-time operation, it would miss most of the people in the video.

Configuration	Memory	Clock Speed	Power Rating (microprocessor)
Xeon E5-1620 GTX 1080	64GB 8GB	3.5GHz 1.7GHz	140W 180W
i7-4790 CPU	16GB	3.6GHz	84W
Atom Z8350	2GB	1.44GHz	2W
RaspberryPi 3B	1GB	1.2GHz	< 1W

Table 2: Details of Processing Units and the platforms used for our experiments. We have used four configurations: CPU + GPU, High-end CPU, Low-end CPU and an ARM processor for the resource-constrained settings.

Model Name	COCO mAP
ssd_mobilenet_v1_coco	21
ssd_inception_v2_coco	24
rfcn_resnet101_coco	30
faster_rcnn_resnet101_coco	32
faster_rcnn_inception_resnet_v2_atrous_coco	37

Table 3: The Tensorflow models from the model zoo for various object detectors used in our experiments.

5.1. Entropy Comparison

If the designer expects a fast moving scene, like counting the number of people in a crowded scene, the entropy of the video is likely to be higher as in Figure 3. On the other hand, if the application is a security surveillance setting where most people will be present for long durations of time, *e.g*. if they stand inside the area being captured in the video, then the entropy will be lower as in 4. Figure 6 shows that variation in FPR of the detector reduces the number of people being detected by the system in a high entropy video however, it remains relatively unaffected in a low entropy video. Therefore, a video/scene context is also an important factor in choosing the detection rates required for the application.

5.2. Infinite Resource Setting (IRS)

In this section, we show the number of people who have been detected upto a certain distance as per the criterion

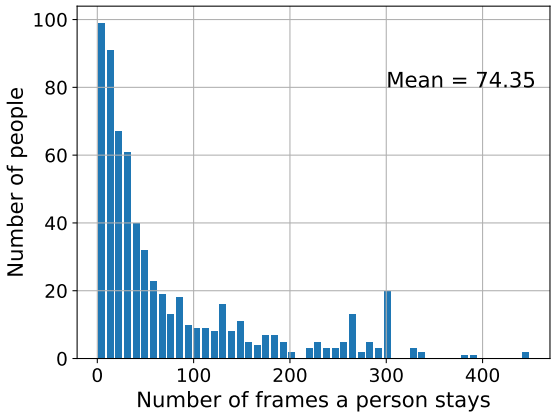


Figure 5: Histogram of the number of frames for which people stay in the video accumulated for all datasets. The high density around low times suggests that the majority of pedestrian are in the field of view for a relatively short time.

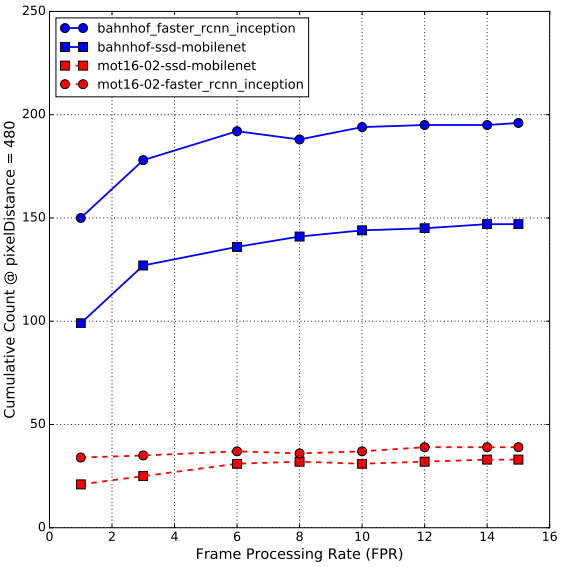


Figure 6: The accuracy on Bahnhof dataset falls on using a lower frame rate (Blue), while it remains relatively constant for MOT16-02 video (Red) since the entropy is higher in Bahnhof video. Therefore, the application being targeted also determines the applicability of the detector.

discussed in 3. So, at a pixel distance of 260, all pedestrians detected before crossing the 260^{th} row in the video are shown. The results are accumulated for different datasets.

Firstly, we show results on how many people are detected in the complete frame, *i.e.*, at a pixel distance of 480. In IRS, all the detectors would run at the same FPR. Therefore, we compare the number of people detected using different

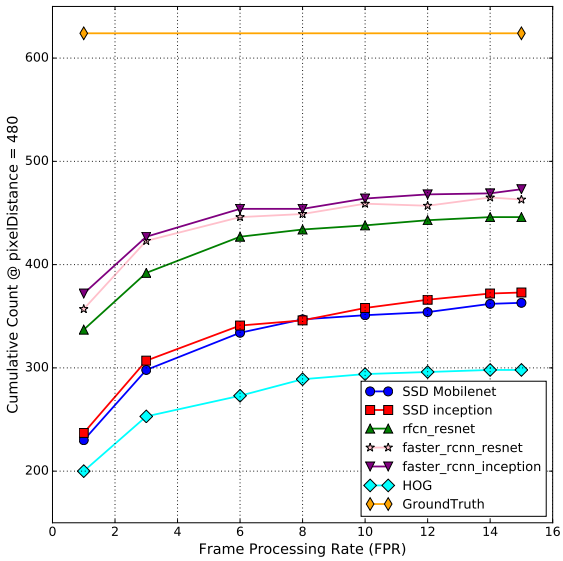
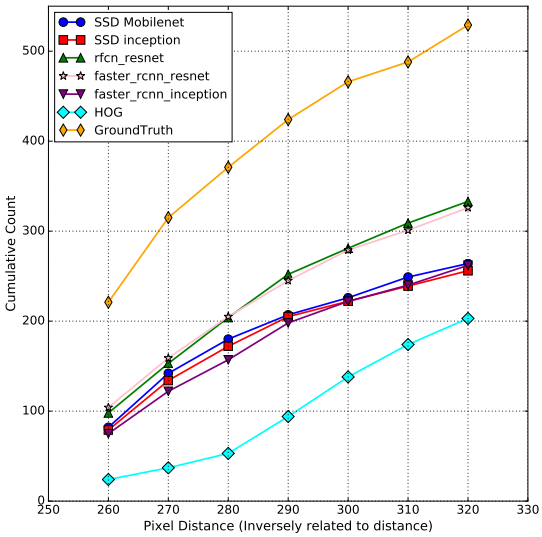


Figure 7: Variation in the number of people detected at different FPRs of the detectors in a video sampled at 15 FPS.

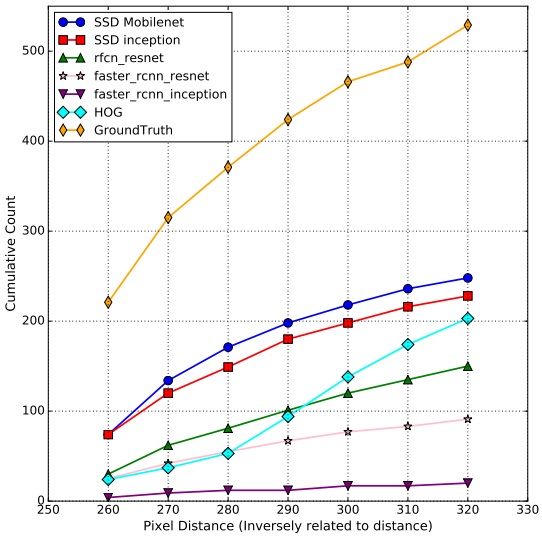
detectors running at the same FPR. The Figure 7 shows that the number of people detected follows the same trend as the accuracy typically reported in the object detection literature. Therefore, mean average precision (mAP) acts as a good criterion for the application performance if all the detectors were able to run at the same FPR. The plot shows the variation of number of people detected when the FPR varies from 2 to 15 while processing a video recorded at 15 FPS. Note that further increase in the FPR of the detector will not result in increased accuracy since all the frames of the video are being processed when the detector is running at an FPR of 15. The ground truth is not varied with FPR since the number of people in the video remains the same.

Now, we look at how detections at different distances are affected with more/less precise detectors. A detector which can detect objects from a higher distance is preferred since it provides enough time to the system to make a decision based on the detection. Figure 8 shows that the detectors which are better (as per mAP) are able to detect more number of people at a greater distance. However, this holds only when all the detectors are running at the same FPR, *i.e.* 15 in this case. We evaluate the case of different FPRs of detectors in Section 5.3.

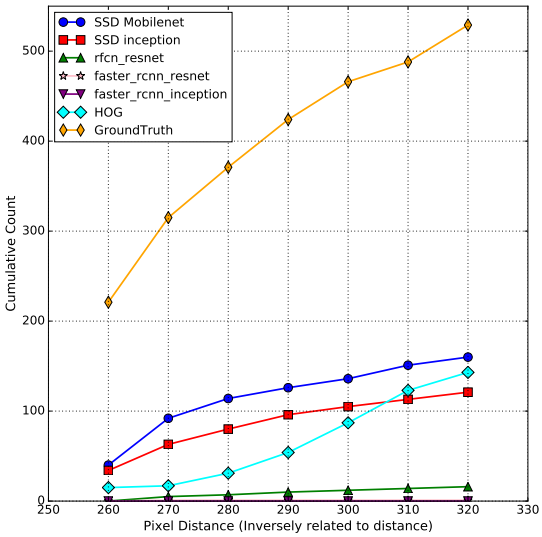
Another observation in this analysis is regarding the False Positives. We analyzed the distances at which different detectors produced false positives. A detection is declared a false positive if the bounding box does not have an overlap with any annotated bounding box which is greater than 0.3. As expected, the false positives at high distances are disproportionately high.



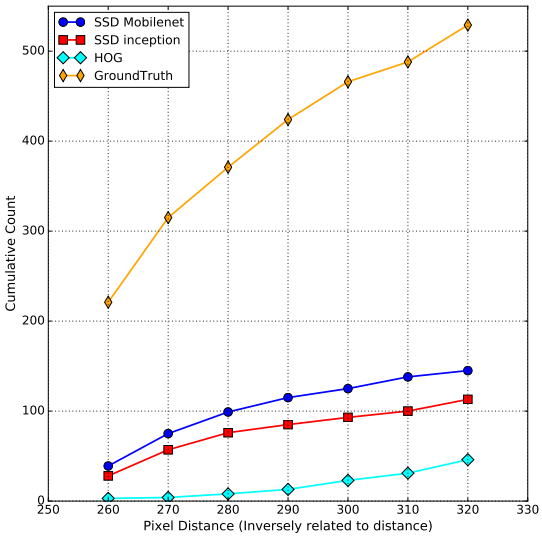
(a) Number of people detected on a CPU+GPU system



(b) Number of people detected on an i7 system



(c) Number of people detected on an Intel Atom System



(d) Number of people detected on a Raspberry Pi Model 3B

Figure 9: The results vary a lot on different platforms. It is clear that the faster detectors perform much better than the more accurate but slower detectors. As we go towards low-resource devices, smaller networks perform even better. The depletion of performance of high-end detectors is evident in all systems without a high end GPU (faster_rcnn). The cross-overs in Figures 9b,9c indicate that depending on criticality of detecting objects from a distance, one detector may be better than the other. Faster detectors do better at smaller distances, however, more accurate detectors (still running at a reasonable speed with respect to entropy of video) work better at a larger distance.

5.3. Resource-constrained Setting (RCS)

In RCS, the Frame Processing Rate (FPR) of a detector on the given system becomes very crucial. The FPRs of different detectors are provided in Table 4. The "-" indicates the field was not calculated because the model couldn't be loaded into the memory. Figure 9 shows the distance-wise comparison of number of people detected on different platforms. This is where the results start reversing from conventional wisdom. On the most high-end system with a CPU as well as a GPU (details in Table 2) shown in Figure 9a, one would expect that the detector with highest mAP yields maximum accuracy. However, *rfcn_resnet* and *faster_rcnn_resnet* emerge as the winners. This is because the detectors have a reasonable FPR apart from being accurate. On a CPU only system shown in Figure 9b,

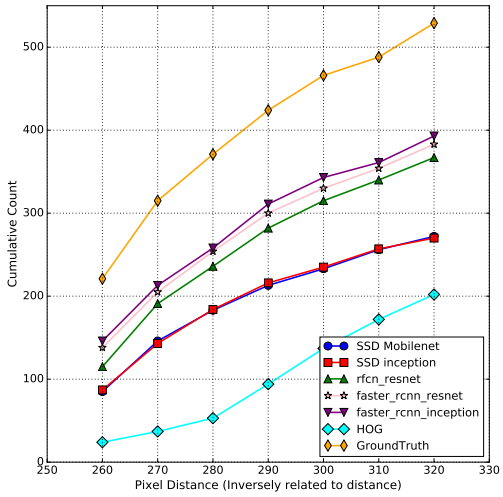


Figure 8: The figure shows how detectors which are better as per their mAP value are able to detect more objects at all distances. The numbers are calculated assuming an FPR of 15 for all detectors.

deeper neural networks lose their edge because of significantly lower FPRs.

Most portable applications which run on a battery would prefer a low power envelope. As shown in Table 2, a power budget of $< 2W$ is available for Intel Atom/Raspberry Pi devices. Clearly, FPR makes a bigger difference on such devices. SSD Mobilenet is able to perform much better than other detectors primarily because of having a reasonable accuracy at a very high processing rate, thus making it an ideal choice for portable systems. On an Intel Atom CPU, as seen in Figure 9c, SSD Inception performs better at higher distances owing to its higher accuracy, however HOG takes over after a certain limit because it is able to run at a faster pace. This indicates that an application which would need more time to make decisions like path planning/alert generation should work with a detector with higher accuracy instead of directly choosing speed over accuracy for low-power systems.

6. Conclusion

Comparison of object detectors on the basis of mAP has been helpful in achieving a higher detection accuracy on a frame-by-frame basis. However, in real-time applications where object detection is a input to other modules in the pipeline and the resources are constrained, higher mAP may not result in the best choice of detector. We have shown here that as the resources are scaled down, speed of the detector becomes an important factor in determining the success of the object detector. However, accuracy remains important in such scenarios as well, even if at the cost of higher run-time since it enables the system to take better decisions by detecting objects at a higher distance. Moreover, different application objectives may demand different detection rates. High entropy scenarios would favor speed over accuracy, however low entropy scenarios would work better with more accurate detectors at reasonable speed. Our evaluation criteria also allows testing of heuristics like spatial and temporal pooling, tracking by detection or other similar techniques to reduce false positives and increase reliability of the complete system.

Platform	Model Name	FPR
CPU + GPU	OpenCV HOG	225
	ssd_mobilenet	12.46
	ssd_inception	11.41
	rfcn_resnet	3.71
	faster_rcnn_resnet	2.86
	faster_rcnn_inception	0.86
CPU	OpenCV HOG	15.95
	ssd_mobilenet	8.03
	ssd_inception	5.38
	rfcn_resnet	0.35
	faster_rcnn_resnet	0.14
	faster_rcnn_inception	0.04
Atom	OpenCV HOG	1.99
	ssd_mobilenet	1.54
	ssd_inception	0.80
	rfcn_resnet	0.041
	faster_rcnn_resnet	0.016
	faster_rcnn_inception	0.005
Raspberry Pi	OpenCV HOG	0.24
	ssd_mobilenet	1.21
	ssd_inception	0.61
	rfcn_resnet	-
	faster_rcnn_resnet	-
	faster_rcnn_inception	-

Table 4: Frame Processing Rates (FPRs) of different detectors on a given system. Refer to Table 2 for details of the platforms.

7. Acknowledgement

We would like to thank Rajesh Kedia for discussions which were helpful in materializing some of these ideas. Anupam Sobti has been supported by Visvesvaraya PhD Fellowship by the Government of India. Chetan Arora has been supported by Visvesvaraya Young Faculty Research Fellowship and Infosys Center for Artificial Intelligence.

References

[1] Google Object Detection API. https://research.googleblog.com/2017/06/supercharge-your-computer-vision-models.html.

[2] IMAGENET Large Scale Visual Recognition Challenge. http://image-net.org/challenges/LSVRC.

[3] OpenCV. www.opencv.org.

[4] Tensorflow. www.tensorflow.org.

[5] Tiny YOLO. https://pjreddie.com/darknet/yolo.

[6] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[7] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner. Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent Transportation Systems Magazine*, 6(4):6–22, 2014.

[8] N. Buch, S. A. Velastin, and J. Orwell. A review of computer vision techniques for the analysis of urban traffic. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):920–939, 2011.

[9] C. Caraffi, T. Vojíř, J. Trefnỳ, J. Šochman, and J. Matas. A system for real-time detection and tracking of vehicles from a single car-mounted camera. In *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 975–982. IEEE, 2012.

[10] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2179–2195, 2009.

[11] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2012 results, 2012. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[12] A. Fregin and K. Dietmayer. A closer look on traffic light detection evaluation metrics. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 971–975. IEEE, 2016.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[14] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012*, 2016.

[17] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv preprint arXiv:1602.07360*, 2016.

[18] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi. Vision for looking at traffic lights: Issues, survey, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1800–1815, 2016.

[19] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *International conference on Pattern recognition (ICPR)*, pages 2756–2759. IEEE, 2010.

[20] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

[21] R. Kedia, K. Yoosuf, P. Dedeepya, M. Fazal, C. Arora, and M. Balakrishnan. Mavi: An embedded device to assist mobility of visually impaired. In *International Conference on VLSI Design and International Conference on Embedded Systems (VLSID)*, pages 213–218. IEEE, 2017.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[23] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[24] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *arXiv preprint arXiv:1701.01909*, 2017.

[25] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.

[26] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. How far are we from solving pedestrian detection? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1259–1267, 2016.