This CVPR Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

The Power of Tiling for Small Object Detection

F. Özge Ünel
Burak O. Özkalaycı
Cevahir Çığla
Aselsan Inc., Turkey
founel@aselsan.com.tr
bozkalayci@aselsan.com.tr
ccigla@aselsan.com.tr

Abstract

Deep neural network based techniques are state-of-the-art for object detection and classification with the help of the development in computational power and memory efficiency. Although these networks are adapted for mobile platforms with sacrifice in accuracy; the resolution increase in visual sources makes the problem even harder by raising the expectations to leverage all the details in images. Real-time small object detection in low power mobile devices has been one of the fundamental problems of surveillance applications. In this study, we address the detection of pedestrians and vehicles onboard a micro aerial vehicle (MAV) with high-resolution imagery. For this purpose, we exploit PeleeNet, to our best knowledge the most efficient network model on mobile GPUs, as the backbone of an SSD network as well as 38x38 feature map in the earlier layer. After illustrating the low accuracy of state-of-the-art object detectors under the MAV scenario, we introduce a tiling based approach that is applied in both training and inference phases. The proposed technique limits the detail loss in object detection while feeding the network with a fixed size input. The improvements provided by the proposed approach are shown by in-depth experiments performed along Nvidia Jetson TX1 and TX2 using the VisDrone2018 dataset.

1. Introduction

In recent years, object detection has been extensively studied for different applications including face detection, video object co-segmentation, video surveillance, self-driving cars and also for higher level reasoning in the context of human-computer interaction [1]. Convolutional neural networks(CNNs) are the workhorse behind the state-of-the-art object detection techniques. In this field, groundbreaking and rapid adoption of deep learning architectures have produced highly accurate object detection methods such as R-CNN [2], Fast R-CNN [3], Faster R-CNN [4], RetinaNet [5], that are later extended to faster and still accurate versions such as SSD[6], YOLO[7], and variants. Generally trained and evaluated on well-known datasets such as ImageNet[8], Pascal VOC12[9], COCO[10], comprehensive comparisons are provided by recent studies.

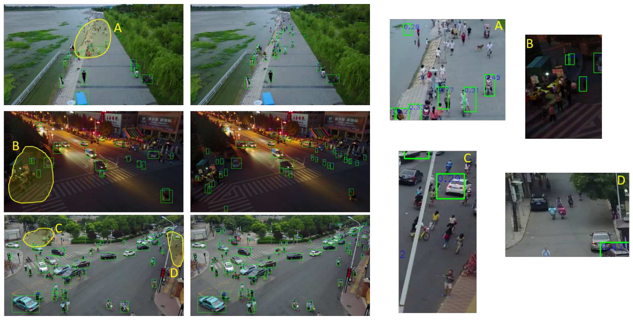

Figure 1. The proposed approach improves small object detection accuracy that is a common problem for recent object detection frameworks

It is important to note that these common data sets mostly involve low-resolution images *(256x256)* including considerably large objects with large pixel coverage. Therefore, the trained models provide very successful detection performance for those types of input data. On the other hand, they yield significantly lower accuracy on small object detection tasks in high-resolution images generated by the high-end cameras. The recent advances in camera and robotics technologies have pioneered surveillance applications in many ways including drones, 4K cameras, and enabled long-range object detection that is met under (D)etection, (O)bservation, (R)ecognition and (I)dentification (DORI) criteria [11]. DORI criteria define the minimum pixel height of the objects for different tasks. According to [11], *10%* the image height is required to detect and observe the objects (*108* pixels in HD videos), while the percentage increases to *20% f*or recognition. Even though DORI criteria is met under certain circumstances, relatively small pixel coverage and down-sampling affect the capabilities of CNN based object detection approaches. In addition, these techniques cannot cope with high-resolution images due to memory requirements and computational constraints.

The challenges met during real-time small object detection problem mostly apply for micro aerial vehicle (MAV) applications [12],[13], where size, weight and power (SWaP) are the limiting factors for use of high performance processors. The MAVs observe the ground at a certain altitude where objects (pedestrian, car, bicycle etc.) in a scene cover considerably smaller pixel areas [14]. Besides, real-time processing is required for these vehicles to enable instant flight control for detection and tracking that are common surveillance applications [14] .

In this study, we propose an efficient solution for small object detection on high-resolution images while maintaining fast execution with low complexity and memory footprint. For this purpose, we focus on pedestrian and vehicle detection on board of MAVs that involves the problems discussed so far. Solution for these problems relies on three stages: In the first step, training data set is augmented with the use of subset tiles that are cropped within the original high-resolution images while the object bounding boxes are mapped accordingly. Such crops map small objects to a larger relative area and prevent misses at the early layers of CNN structure during the train phase. The second step focuses on the deployment of CNN, where target images are divided into overlapping tiles and object detection is executed on each tile independently. Object proposals in each tile are merged for final detections in the original resolution of the input image. In order to benefit the power of tiling while fulfilling real-time requirements, we exploit an efficient framework, Pelee [15], that is originally applied on images of size *304x304*. We modify the feature resolution in Pelee to tackle small objects as the third alternative. All the experiments are conducted on NVIDIA Jetson TX1 and TX2 modules which are very popular mobile GPUs.

The organization of the paper is a follows. In section 2, we briefly talk about the previous work on object detection mainly focusing CNNs. In the third section, problem is clarified by introducing the main algorithm design, description of the data set used in the paper. After presenting experimental results in Section 4, we conclude the paper with certain remarks on the use of tiling at different stages.

2. Related Work

The recent learning-based object detection techniques can be categorized into two: The region proposal based methods that involve generation of region proposals followed by classification and the regression/classification based methods adopting a unified framework for detection and classification. Each mainstream approach is summarized in the following sub-sections providing the drawbacks specifically on small object detection and give certain details about the recent advances in this topic.

2.1. Region Proposal Based Framework

These methods, including R-CNN, SPP-net [16], Fast R-CNN, Faster R-CNN, R-FCN [17], FPN [18], and Mask R-CNN [19], formulate the object detection task as a two-stage problem. The first stage generates region proposals within an image while the second stage classifies each proposal into different object categories according to CNN based deep features. These methods use regions to localize the objects rather than looking at the full frame.

Significant amount of research is devoted to improve the extraction of candidate bounding boxes that replaces-the well-known sliding window approach. R-CNN generates about *2K* region proposals for each image based on selective search [20] that uses a simple bottom-up grouping and provides more accurate candidate boxes in arbitrary sizes. This approach can be computationally expensive due to straightforward two-step approach: the first step crops externally computed box proposals, then the second step executes a CNN classifier on these cropped boxes. Fast R-CNN alleviated this problem by pushing the entire image once through a feature extractor and then cropping at an intermediate layer. In order to accelerate the process, Faster R-CNN generates box proposals via neural networks instead of an external proposal utilized in both R-CNN and Fast R-CNN. Although these methods have provided more accurate results, they are far from real-time deployment on mobile devices due to high computation and memory requirements as given in Table 1.

2.2. Regression/Classification Based Framework

Due to the ever increasing number of applications requiring accurate object detection on mobile devices, extensive research is conducted on efficient model design. This tendency pioneered the regression/classification based approaches such as MultiBox [21], AttentionNet [22], G-CNN [23], YOLO [7], SSD[6], YOLO variants [24, 25] and Pelee[15]. These techniques exploit a single convolutional network to predict both the bounding boxes and class probabilities of the objects in a single run that significantly reduces the computational load.

In this class of algorithms, YOLO splits an image into an *MxM* grid, within each, only one object is predicted. Each grid predicts a fixed number, say *N*, of bounding boxes, which keep class probability and offset values. Hence, a total of *MxMxN* boxes are predicted. The boxes having a class probability above a specified threshold are used to locate the objects in whole image. YOLO struggles in generalization of objects with unusual aspect ratios and dealing with small objects that impose strong spatial constraints on bounding box predictions. In order to handle these problems, SSD is introduced that is a combination of YOLO and Faster R-CNN. SSD takes an input image and learns the bounding box and class probabilities at the same time as in YOLO. Furthermore, it uses anchor boxes at various aspect ratios such as *1/2*, *1/3*, *1*, *2*, *3* similar to Faster R-CNN, hence, reducing the previous problems of both.

Feature extraction methods such as MobileNetv2 [26] and ShuffleNet [27] used in SSD are heavily dependent on the depth-wise separable convolution, which lacks efficient implementation. On the contrary, a recent method, Pelee, proposes an efficient architecture that is built on conventional convolution with DenseNet [28] to alleviate the efficiency problem. In addition to stem block and composite function optimizations, Pelee uses 5 scale feature maps *(19x19, 10x10, 5x5, 3x3, 1x1)* to reduce computational cost instead of one *38x38* feature map used in SSD. As a result, Pelee has faster and more lightweight feature extractors, while yielding still accurate results [15].

A comprehensive comparison for the regional and regression-based techniques is given in Table 1, which is an extended version of a table presented in [15], with the entries copied from the related papers. It is clear that regional methods are far from real-time deployment on mobile platforms. Pelee and YOLOv3 are the most efficient models with comparable accuracy.

2.3. Improvements for Small Objects

SSD models are competitive with Faster R-CNN and R-FCN on large objects, while they typically have (very) poor performance on small objects [29]. For this reason, studies have been revealed to ensure speed balance of accuracy in small objects. RMNet[30], a recent study, which does not compromise the accuracy while satisfying the real-time computational constraints. The authors emphasize the depth of the CNN based network as the key component of a robust feature extractor and utilize a residual deep network with a hundred layers. In order to overcome the computational burden of a very deep network, the residual blocks are lightened by depth-wise convolutions, bottlenecks and the limited number of feature channels. An SSD model using RMNet as the backbone, which is available in the OpenVino toolkit[31], ingests images of *1024x1024* and leverages relative high-resolution in small object detection.

[32] uses a two-level tiling based technique in order to detect small objects. In the first level YOLO-v2 object detection model is utilized as an attention model to focus on the regions of interest with a coarse tiling of the high-resolution images up to 8K. In the second level, attention outputs are used to select image crops of a finer tiling, and the same object detection model is applied once more on these image crops. Both the first and second level detection tasks for the corresponding image crops are performed on a GPU server. In contrast to their work, we design a fixed scheme tiling and avoid any extra computational needs like an attention model to operate fully on the edge side.

Recently, [33] proposes a dynamic tiling approach where the sizes of tiles are arranged according to the network input size. Even though, it is not constrained by real-time execution (by use of tiny-YOLO and DRONET), [33] introduces novel attention and memory mechanisms for efficient and dynamic tile utilization. In contrast to their work, we propose overlapping static tiles to preserve object detection along the tile boundaries, extend the use of tiles during train stages and focus on real-time execution.

3. Problem Description

The constraints on the real-time deployment of object detection frameworks on mobile devices (especially MAVs and battery equipped vehicles) strictly limits the number of available networks. The aforementioned region-based approaches are not convenient for prompt operation, while regression-based techniques are designed for onboard processing. In Table 1, accuracy on COCO data-set and computation timings (frame per seconds (FPS)) on Nvidia TX2 are illustrated for various regression-based techniques with their mobile adapted versions as well. According to the table, recently introduced Pelee is not only faster than all mobile detection frameworks (SSD+MobileNet and YOLOv3-tiny [35]) but also enables comparable accuracy with respect to the models having the 10x larger size and *4x* slower computation capability (such as YOLOv2). To our best knowledge, Pelee is not only the currently best alternative among various state-of-the-art techniques but also provides enough room for additional computation.

Even though Pelee enables real-time performance on mobile GPUs, it still suffers from the detection of small objects in high-resolution images due to characteristics of SSD as shown in Figure 1. In the figure, typical outcomes of Pelee detection (*304x304*) are shown for a couple of images in the VisDrone2018 Video[36] data set. The poor object detection performance is obvious especially for smaller objects along with the areas indicated by yellow *A-B-C-D*.

VisDrone2018 VID[36] data set with more than a million bounding box annotations in training set, *140k* annotations in validation set for 11 different class labels (pedestrian, person, bicycle, car, van, truck, tricycle, awning-tricycle, bus, motor, others) is a perfect choice for our scenario. We group the classes in VisDrone2018 into two main groups as pedestrian and vehicle for a more well defined simpler task. The pixel height and width histograms of human and vehicle classes in the training set are shown in Figure 2 with purple color (after scaling the images into *1920x1080*). It is clear that half of the annotations correspond to objects having pixel heights and widths smaller than *50* pixels. This value is half of the threshold that is discussed in the introduction section according to the standards [11]. In the VisDrone2018 dataset, small objects form the majority and only half of the objects meet the monitoring threshold and only *20%* of the objects meet the detection threshold of *10%* of the image height. This is a common case for MAV based surveillance. Under these circumstances, the expected detection rate for a perfect detector can be at most *50%* after shifting the margins for a favor (setting the threshold as half of the DORI standards).

4. Proposed Approach

In order to handle the small object detection problem, we propose a tiling approach that is applied both in the training

Table 1. Object detection result on COCO test-dev2015

			Speed on		Avg. Precision (%), Iou:	
Model	Input Dimension	Model Size	TX2(FPS)	Titan(FPS)	0.5:0.05:0.95	0.5
Fast R-CNN [3][1]	224x224	-	NA	-	19.7	35.9
Faster R-CNN[4][1]	224x224	-	NA	7	21.9	42.7
SSD300[6]	300x300	-	-	46	23.2	41.2
SSD512[6]	512x512	-	-	19	26.8	46.5
SSD300[6][2]	300x300	34.30 M	-	46	25.1	43.1
SSD512[6][2]	512x512	-	-	19	28.8	48.5
YOLOv2[15]	416x416	67.43 M	32.2	-	21.6	44.0
YOLOv3[15]	320x320	62.3 M	21.5	-	-	51.5
YOLOv3-Tiny	416x416	12.3 M	105	-	-	33.1
SSD+MobileNet	300x300	6.8 M	80	-	18.8	-
SSDlite+MobileNetv2	320x320	4.3 M	61	-	22	-
Pelee[15]	304x304	5.98 M	120	-	22.4	38.3

[1] VGG-16[34] based [2] These models are trained with the new data augmentation

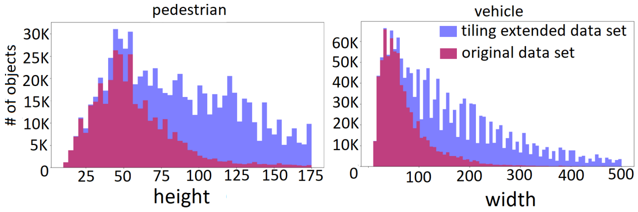

Figure 2. The height and width historams of pedestiran and vehicle objects for original and tile extended data set

and inference stages as illustrated in Figure 3. In the figure, a typical *3x2* tiling is shown, where arbitrary tiling can be applied depending on the image resolution and target object aspect ratios. The input images are divided into overlapping tiles so that the relative pixel area of small objects increases with respect to the images fed into the network. This is applicable to any kind of network, on the other hand, we have chosen Pelee [15] to obtain significant improvement in accuracy with the help of fast execution.

4.1. Pelee Architecture

The entire PeleeNet network consists of the Stem Block motivated by Inception-v4 [37], seven stages of feature extractor and ResBlocks. First, four stages of feature extractor consist of Dense Block which connects each layer to every other layer in a feed-forward fashion. In the training process, the input frame is downsampled to *304x304* resolution. Then, the downsampled image is fed to the stem block which improves the feature expression performance without too much computational cost. In feature extraction stages, the network learns visual representation in 5 scales of feature maps *(19x19, 10x10, 5x5, 3x3, and 1x1)* with different aspect ratio, which is given to the residual block (ResBlock) [38] before producing predictions for object class and bounding box location.

Our object detection system is based on the source code of SSD [1] and is trained with Pytorch [39]. The VGG-16[34] network used for feature extraction is replaced with Pelee. The batch size is set to *32*. Our momentum value is *0.9*, weight decay is *5e-4* and gamma value is *0.5*. The learning rate is set to *0.001* initially, then decreased by a factor of 10 after 10K, 20K, 30K, 40K and 70K iterations, respectively. The training is terminated at *120Kth* iteration.

4.2. Tiles in network training

In order to alleviate small object problems, we decrease the effect of image down-sampling that is a common tool applied during training. The images are divided into smaller images by the help of overlapping tiles, where the sizes of tiles are selected according to the size of the images utilized in the related train framework. As shown in Figure 3, lower resolution images *(MxN)* are cropped from the original images by overlapping tiles. It is important to note that, input image resolution is set to *1920x1080* just before tiling in order to fix the size of tiles. Each tile corresponds to a new image where ground truth object locations are arranged accordingly without change of object size. In that way, the relative object sizes are increased in the cropped image compared to the full frame. The cropped *(MxN)* images and the full frame are utilized as the input data for the training of the network. It is important to note that the full frame is also fed into the network training in order to detect large objects in the scene.

The overlaps between the tiles are used to preserve the objects along the tile boundaries and prevent any miss due to image partitioning. In this study, we have chosen *25%* intersection between consecutive tiles; resolution *N* of the

[1] https://github.com/amdegroot/ssd.pytorch

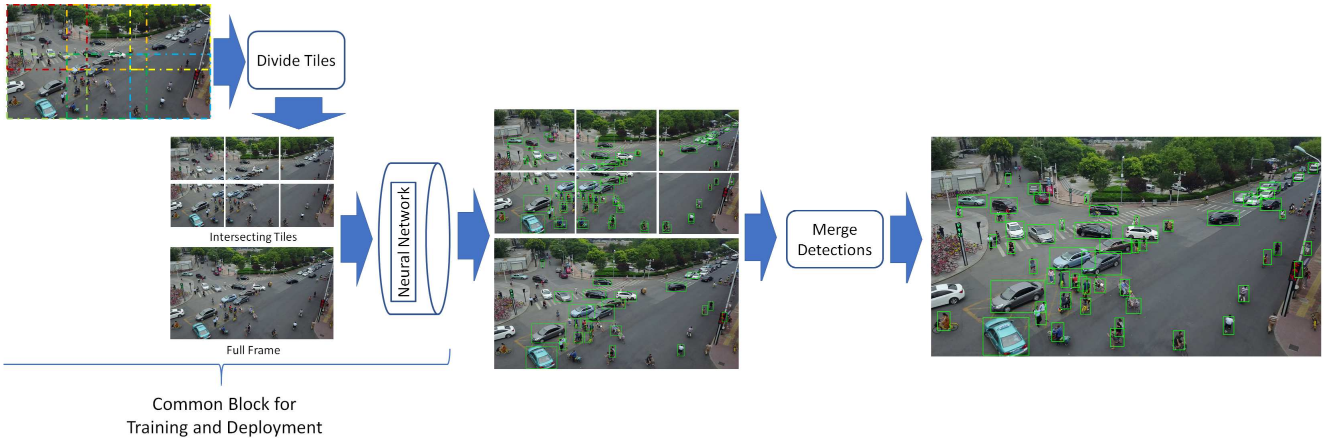

Figure 3. The proposed tiling approach is both applied in training the network and deployment stages. The object detection results on the tiles and the full frame are merged for the final outcome.

Table 2. The effect of tiling grid on cropped image size, number of images and annotated bounding boxes

Tiling Grid	Resolution(wxh)	Image	Bbox
1x1	1920x1080	24198	1106990
3x2	768x432	147412	2527166
5x3	480x432	256341	2553814

sub-sampled images (for both width and height) is given according to the number of tiles (T) and image size (S) as:

$$N = 4S/(3T + 1). \tag{1}$$

The effect of tiling on the cropped image resolution, the number of bounding boxes and images for VisDrone2018 data set are given in Table 2. Increasing the grid size to *5x3* results in *2.5* fold increase in the number of annotated bounding boxes due to overlapping ratio. Comparing the image resolution of Pelee, which is *304x304*, and the tile resolutions, the rate of down sampling is greatly decreased to keep small objects in the detectable range of network. As the tiling increases, larger objects may not fit within the tiles and the intersecting areas, and the risk of loosing larger object annotations also increases. Therefore, at some point the increase in tiles starts to decrease the number of annotations. The effect of tile extension on the distribution of object sizes is given in Figure 2, where the blue color represents the extended histogram. The object annotations within the tiles are scaled up with respect to the scale that maps cropped images to full resolution (*1920x1080*) to extract the histogram. Hence, as observed the increase of large objects in Figure 2, smaller objects are treated as larger objects by the proposed tiling approach.

4.3. Tiles during inference

The same structure given in Figure 3 is also exploited during the detection of objects. First of all, the input frames are resized to *1920x1080* and tile images are generated by cropping the input frames according to the number of tiles that is determined by the computational capacity. The tile grid during inference can be different from the grid in training phase due to the computational issues. Each tile is treated independently as well as the original frame and the resulting detection boxes with class probabilities are gathered as the initial results. At this point, there will be duplicate object detections in the initial results due to overlaps between the tiles and the full frame. The initial results are merged according to the intersection of bounding boxes and class scores. If the intersection of duplicate detections is above *25%*, then the one with higher score is accepted as a better choice and the other one is removed from the detection list. In the merge step, small objects get higher scores within the tiles compared to the full frames in general, while reverse is the case for bigger objects that have sizes comparable with the tiling area. Therefore, both small and large objects are handled carefully.

The number of tiles linearly increases the complexity of the overall detection framework. Therefore, this approach is applicable for lightweight and efficient networks in order to meet real-time inference without increasing memory usage. On the other hand, small object detection accuracy can be increased obviously because relative object sizes can really be very small in the original high-resolution frames. Even low number of tiles can improve small object detection significantly which is presented in the experiments section.

4.4. Pelee Framework Modifications

The tiling approach introduces additional computation that linearly depends on the number of grids. In order to tackle the small object problem, the number of feature extraction layers can also be increased to detect small objects without increasing the computational time too much. In the original SSD structure, *38x38, 19x19, 10x10, 5x5, 3x3, and 1x1* feature vectors are utilized. SSD with Mobilenet [40] does not use the *38x38* feature vector to provide the balance between speed and accuracy. Instead of *38x38*, they add another *2x2* feature map for prediction. In this way, they use a smaller network to gain from speed while scarifying accuracy in small object detection. The original Pelee architecture also discards the *38* feature map for speed versus accuracy trade-off but does not utilize a *2* feature map like MobileNet. In our framework, we train the Pelee network with *38* feature map to increase accuracy in small objects.

5. Experiments

We have examined the effects of different feature extraction approaches and various tiling grid sizes in the train and inference phases on the speed and accuracy for small object recognition. It is important to note that computation times on mobile GPUs are the limiting factors for the tiling grid size. We make use of the criteria given in Embedded Real-time Inference Challenge [41], indicating at least 5 fps on TX2. Vino, Pelee and Pelee38 (that uses *38x38* feature maps) run at *16.5*, *101* and *77.8* FPS on TX2 accordingly. Thus, we can do at most *2x1* tiling for Vino, while tiling can be up to *5x3* for Pelee. Vino tiling does not include the full frame because of the computational issues. The performance metric is calculated on the VisDrone2018 [36] validation videos involving *2843* images with *114k* annotations.

In Table 3, the comparison between two efficient networks Vino [31] and Pelee[15] that utilize different feature extraction frameworks is given with their single (first two rows) and best performance tiling versions under 5 fps limitation on TX2 (rows 5 and 6) based on the accuracy. Pelee_T5x3_I5x3 indicates that the Pelee model is trained through *5x3* tiling and *5x3* tiling is utilized in inference time. There are two main differences between these networks: Vino gets images with size *1024x1024* while Pelee gets images of size *304x304* as input, and Pelee utilizes DenseNet based feature extraction methods while Vino makes use of RMNet, which is a larger and more expensive network. As shown in Table 3, Vino has a higher accuracy than Pelee when original networks are applied without any tiling; because Vino is trained on larger images and is a deeper model.

Tiling significantly improves the performance of Pelee such that the MAP (IoU: 0.5) is increased from 11% to 36%. The accuracy of vehicles and pedestrians in *5x3* tiling are increased by *2.5x* and *5x* correspondingly. It is clear that, tiling is more effective on small objects, pedestrians in this case, while it also boosts up medium sized objects.

As indicated previously, Vino is almost 10 times more time consuming than Pelee, hence tiling grid size is smaller. Tiling Pelee outperforms Vino by *25%* while the computation is kept over 5 fps on TX2. The results in Table 3 indicate that the Vino *2x1* tiling model in inference can not increase accuracy much more than single version of Vino because of the structure of SSD. SSD makes many predictions for better coverage of location, scale and aspect ratios such as *1/2, 1/3, 1/1, 2/1, 3/1*. Vino *2x1* tiling grid which is selected based on the available processing limits, match up with SSD aspect ratio.

The effect of feature extraction layer is also analyzed in Table 3, where a comparison between different number of feature extraction networks (Pelee and Pelee38) is given with their single and tiling versions based on the accuracy. Pelee38 uses one additional feature vector (extra 38) compared to original Pelee. It has been observed that the object detection MAP (IoU: 0.5) increases from 11% to 23% while frame rates are decreased by 25% on average for both TX1 and TX2. Pelee38 improves pedestrian detection more significantly than the vehicle detection, *200%* and *64%* correspondingly. In no tiling case, by the use of larger feature set *(38x38)*, the network shows greater success, as expected, in small objects.

The improvement with tiling for Pelee38 is not as significant as in the case of Pelee19 due to smaller object adaptation with the additional feature introduced in Pelee38. Moreover, Pelee38 increases computation by *16%* and *25%* for TX1 and TX2. In that manner, original Pelee feature map is more appropriate for tiling operations while Pelee38 is more efficient for non-tiling scenarios.

The effect of object height on the detection accuracy is also analyzed in Figure 4, where the first row compares the hit rate of the networks (Vino, Pelee, Pelee38 and tiling versions) for the IoU of *0.5*. Pelee and Pelee38 with tiling outperform Vino for smaller objects in both pedestrian and vehicle classes where after the height of *100* pixels in pedestrian and width of *150* pixels in vehicle classes, Vino outperforms Pelees. In general, object detection accuracy increases with the size of the objects while there is a sudden accuracy drop in all networks for vehicles around width of *270* pixels. This is mainly due to the low number of objects at that width such that false prediction affects the accuracy with higher impact. Figure 5 illustrates the height and width distribution of pedestrian and vehicles(blue) and the number of detected objects at the specific height and widths (green) by Pelee_T5x3_I5x3. The vehicles wider than *120* pixels are almost detected by *100%* while detection does not exceed *85%* for pedestrians at any pixel heights. Figure 5 is informative for the comprehension of the capability of networks with respect to the object sizes.

For the sake of completeness, visual results are also given on three samples of VisDrone2018 image set in Figure 6. The results also support that tiling has a profound effect on the detection of smaller objects as well as the objects located in low contrast regions. The lower left region in the night image (center column) and lower right region of the image in the third column have low contrast regions

Table 3. The accuracy achieved by Vino and Pelee with/without tiling. The tile grid for Vino is smaller than Pelee according to the computational complexity.

Model	Avg. Precision (%), Iou:									FPS on:	
	Vehicle			Pedestrian			MAP				
	0.5	0.75	0.5:1.0	0.5	0.75	0.5:1.0	0.5	0.75	0.5:0.95	TX1	TX2
Vino	**32.27**	17.29	17.21	**24.74**	3.64	8.75	**28.50**	10.46	12.98	11.7	16.5
Pelee	17.03	5.68	7.39	5.93	0.18	1.43	11.48	2.93	4.41	58	101
Pelee38	28.45	14.30	14.95	17.97	1.28	5.28	23.21	7.79	10.11	48.2	77.8
Vino_I2x1	33.30	15.40	17.17	24.51	3.60	8.75	28.91	9.50	12.96	5.7	8.3
Pelee_T5x3_I5x3	41.39	19.55	21.07	**30.26**	2.94	9.61	35.82	11.24	15.34	3.5	6.3
Pelee38_T5x3_I5x3	**44.35**	22.64	23.53	28.99	3.25	9.69	**36.67**	12.95	16.61	3	4.8

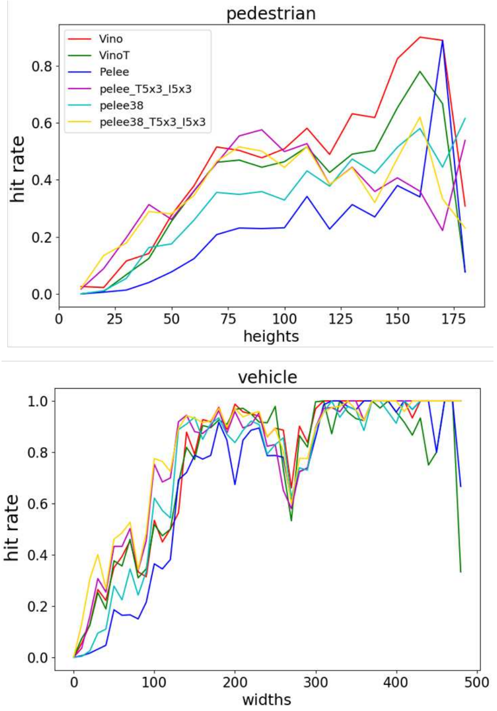

Figure 4. Detection performance for different networks with respect to object sizes

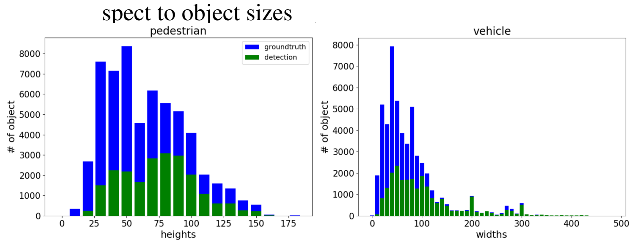

Figure 5. Detection performance for Pelee_T5x3_I5x3 with respect to object sizes

that makes object detection even harder. On the other hand, tiling on Pelee can detect images significantly better than the other techniques.

Table 4. The effect of different tiling grids in network training and inference when IoU is 0.5

T/I	3x2	4x4	5x3	6x3
Single	26.55	25.77	28.48	27.81
3x2	30.63	34.62	34.11	31.69
5x3	28.57	30.86	35.82	35.88

Table 4 shows the effect of different tiles in training and inference stage on MAP accuracy when the IoU is 0.5 on the Pelee network. Rows and columns indicate the tiling grids in train (T) and inference (I) phases. Tiles in the training phase have a significant impact on the accuracy, such that each value in the last two rows is larger than the values in the first row. There is almost *40%* increase in performance between the worst and the best cases. There is also an obvious relation between the tiles in inference and train such that inference tile grids should be more than train tile grids and the difference should not be too large. The best performing tiles in inference for the case where training is achieved by tiles of *3x2* and *5x3* are *4x4* and *6x3* correspondingly, which are within a neighborhood of train tiles.

In order to observe all the possibilities in tiling for different networks (Vino, Pelee-P, Pelee38-PP) with the effects on Top-1 accuracy and computational time (on TX2), the experiments are summarized in Figure 7. The red line indicates the *5 FPS* threshold that is a criteria in the aforementioned challenge. We discard tiles larger than *3x2* for Vino since the timings increase significantly while accuracy does not improve; moreover we include full frame (as indicated by *+1*) Vino detection for this plot. As shown in the Figure, Pelee_T5x3_I5x3 has the best accuracy within the available processing limits. Vino tile *3x2* has the best accuracy with a significant overhead in computation, while Pelee38 with single tile is the fastest approach that exceeds *20%* top-1 accuracy threshold. Pelee38 trained through *5x3* tile and inference with *3x2* tile provides a good trade-off between accuracy and computational time.

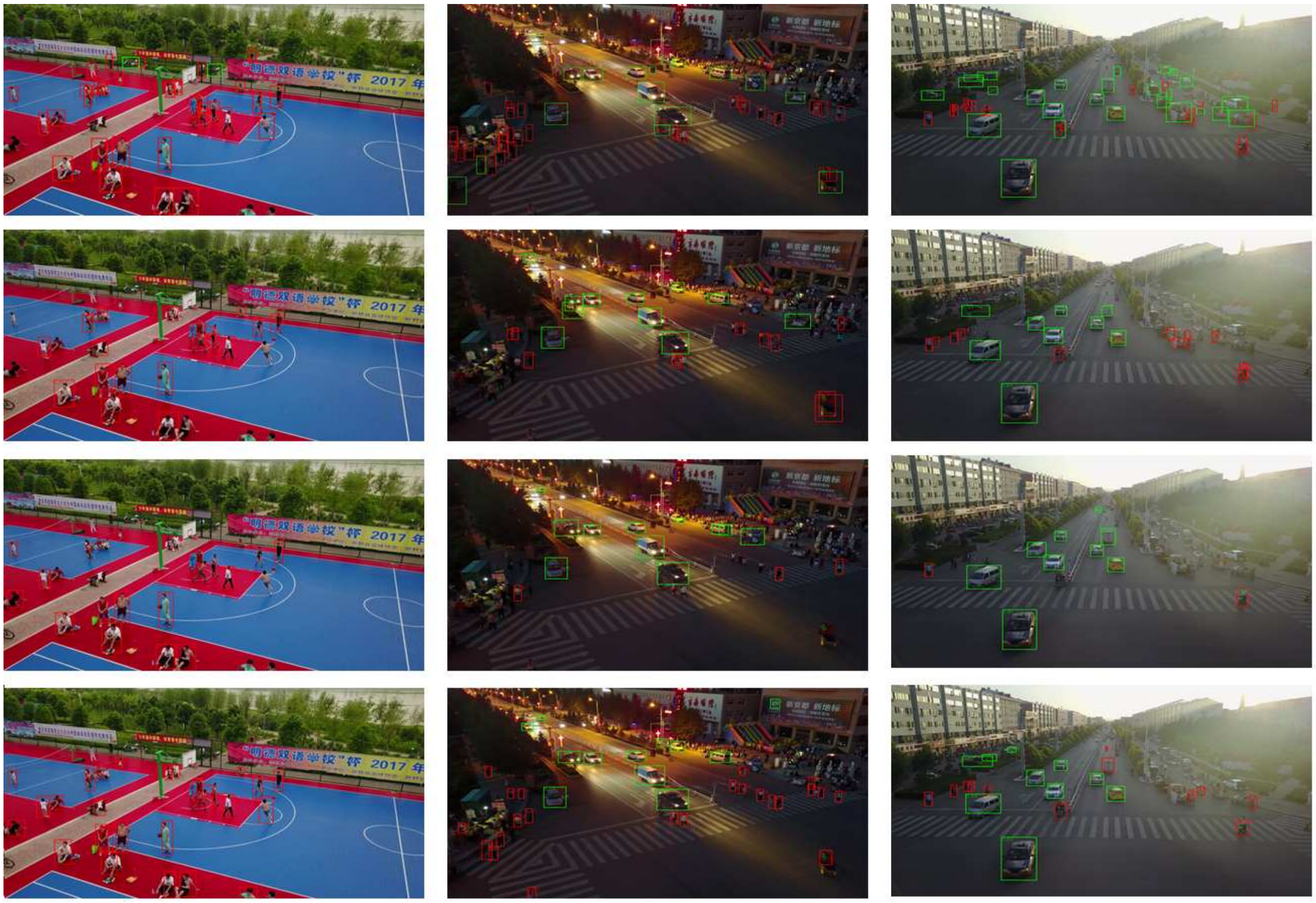

Figure 6. Top to bottom: ground truth, Pelee, VINO, Pelee 5x3 tiling with pedestrian (red) and vehicle labels(green)

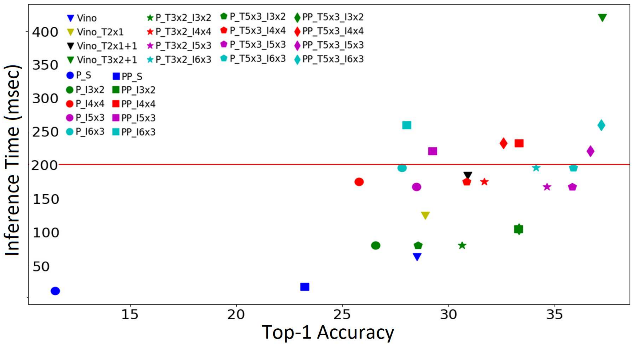

Figure 7. Speed versus Accuracy graph on different network

6. Conclusion

Processing fixed size input makes the deep neural network based detectors prone to miss small objects. Especially for the SSD approach, prior boxes used as a reference for the localization problem defines a lower bound for the size of minimum detectable objects. Extending the feature maps of Pelee with 38x38 feature map as in the original SSD, boost the accuracy more than *3* times for the pedestrian class. Complementary to using finer feature maps in SSD network is to provide high-resolution images into network without losing their details as much as possible. The processing time of PeleeNet gives opportunity to use it multiple times to focus on the details of the image of interest. The proposed approach exploits this fact and partitions the image into tiles according to real-time constraints. The accuracy boost obtained by the tiling approach reaches to *4* fold with respect to the conventional method. The experiments show that mimicking the inference tiling at the training phase is also beneficial. Providing the training data in a similar distribution of image resolution makes the network to learn image space representations better. Using tiles at training step as an additional data augmentation method also increases the small object detection performance *20%* significantly.

Training a network with higher resolution images through larger feature maps, result in high computation and memory requirements. The proposed tiling approach increases the computational time linearly while keeping memory requirements fixed due to sequential tile processing. Computation and memory budgets can also be traded-off by feeding the tiles in batch format. Hence the proposed approach can be even used as a parameter in network design, considering the budgets of a targeted platform.

References

[1] Georgia Gkioxari, Ross B. Girshick, Piotr Dollár, and Kaiming He. Detecting and recognizing human-object interactions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8359–8367, 2018.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[3] Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 91–99. Curran Associates, Inc., 2015.

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37, 2016.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[11] European Committee for Electro-technical Standardization. Alarm systems- cctv surveillance systems for use in security applications. *European Standart*, August, 2012.

[12] S. Raj, R. Adriaan, R. Artem, L. Vincent, G. Denis, F. Pascal, and M. Alcherio. Vision-Based Unmanned Aerial Vehicle Detection and Tracking for Sense and Avoid Systems. *IEEE International Conference On Intelligent Robots And Systems*, pages 1556–1561, 2016.

[13] C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris, and C. S. Bouganis. DroNet: Efficient Convolutional Neural Network Detector for Real-Time UAV Applications. In *Design, Automation Test in Europe Conference Exhibition*, pages 967–972, March 2018.

[14] Nils Tijtgat, Wiebe Van Ranst, Bruno Volckaert, Toon Goedemé, and Filip De Turck. Embedded real-time object detection for a uav warning system. *IEEE International Conference on Computer Vision Workshops*, pages 2110–2118, 2017.

[15] Robert J. Wang, Xiang Li, and Charles X. Ling. Pelee: A real-time object detection system on mobile devices. In *Advances in Neural Information Processing Systems*, pages 1963–1972. 2018.

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

[17] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016.

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[20] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[21] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2155–2162, 2014.

[22] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S Paek, and In So Kweon. AttentionNet: Aggregating weak directions for accurate object detection. In *IEEE International Conference on Computer Vision*, pages 2659–2667, 2015.

[23] Mahyar Najibi, Mohammad Rastegari, and Larry S. Davis. G-CNN: an iterative grid based object detector. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2369–2377, 2016.

[24] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[25] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[26] M Sandler, A Howard, M Zhu, A Zhmoginov, and LC Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. arxiv 2018. *arXiv preprint arXiv:1801.04381*.

[27] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[29] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7310–7311, 2017.

[30] Evgeny Izutov. Fast and accurate person re-identification with RMNet. *arXiv preprint arXiv:1812.02465*, 2018.

[31] https://docs.openvinotoolkit.org/R5/, 2019.

[32] Vít Ržička and Franz Franchetti. Fast and accurate object detection in high resolution 4K and 8K video using GPUs. In *IEEE High Performance extreme Computing Conference*, pages 1–7, 2018.

[33] George Plastiras, Christos Kyrkou, and Theocharis Theocharides. Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, ICDSC '18, pages 3:1–3:6, New York, NY, USA, 2018. ACM.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[35] YOLOv3-tiny. https://pjreddie.com/darknet/yolo.

[36] Pengfei Zhu, Longyin Wen, Xiao Bian, Ling Haibin, and Qinghua Hu. Vision meets drones: A challenge. *arXiv preprint arXiv:1804.07437*, 2018.

[37] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR 2016 Workshop*, 2016.

[38] Kyoungmin Lee, Jaeseok Choi, Jisoo Jeong, and Nojun Kwak. Residual features and unified prediction network for single stage detection. *arXiv preprint arXiv:1707.05031*, 2017.

[39] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

[40] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[41] Embedded Real Time Inference Challenge 2019. https://sites.google.com/site/uavision2019/challenge.