Real-Time Vehicle Detection from Short-range Aerial Image with Compressed MobileNet

Yuhang He[1], Ziyu Pan[1], Lingxi Li[2], Yunxiao Shan[1], Dongpu Cao[3] and Long Chen[1]

***Abstract*— Vehicle detection from short-range aerial image faces challenges including vehicle blocking, irrelevant object interference, motion blurring, color variation *etc.*, leading to the difficulty to achieve high detection accuracy and real-time detection speed. In this paper, benefiting from the recent development in MobileNet family network engineering, we propose a compressed MobileNet which is not only internally resistant to the above listed challenges but also gains the best detection accuracy/speed tradeoff when comparing with the original MobileNet. In a nutshell, we reduce the bottleneck architecture number during the feature map downsampling stage but add more bottlenecks during the feature map plateau stage, neither extra FLOPs nor parameters are thus involved but reduced inference time and better accuracy are expected. We conduct experiment on our collected 5-*k* short-range aerial images, containing six vehicle categories: truck, car, bus, bicycle, motorcycle, crowded bicycles and crowded motorcycles. Our proposed compressed MobileNet achieves 110 FPS (GPU), 31 FPS (CPU) and 15 FPS (mobile phone), 1.2 times faster and 2% more accurate (mAP) than the original MobileNet.**

I. INTRODUCTION

Object detection is a classic and fundamental problem in robotics. It serves as prerequisite prior information for other advanced technologies, such as simultaneous locationlization and mapping (SLAM) [1][2], point cloud semantic segmentation [3][4] and object tracking [5][6][7][8]. Object detection targets at localizing object and identifying its category simultaneously. It has gained significant progress in recent years, owing to the successful design of powerful detection neural network [9][10][11] and the availability of various large-scale object detection datasets [12][13]. On the one hand, accuracy oriented object detection algorithms [9][14] often consist of two stages, regional proposal network (RPN, aka backbone) and the subsequent proposal regression and classification network (aka head). The exploitation of heavy backbone, like ResNet [15] and NasNet [16], and the mining of useful image context information [17][18] often result in two-stage detection algorithms staying in the front rows of various object detection challenges. The corresponding speed oriented object detection algorithm, however, aims to achieving real-time application in mobile devices with an affordable accuracy loss. To this end, various light weight neural networks have been designed [19][20][21] and the one-stage detection methods also have been created [10][11].

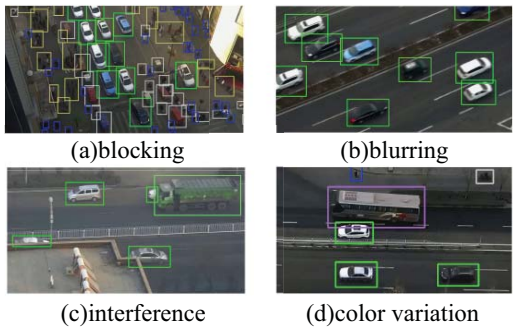

Fig. 1. Sample challenges that commonly exist in vehicle detection from short-range aerial images. Vehicles with different categories are surrounded with bounding boxes with different colors for better visualization.

As a special kind of object detection, vehicle detection from short-range aerial images has strong application in either autonomous driving or video surveillance. It provides essential vehicle information for both internet of vehicles (IoV) and realtime transportation condition for city scale surveillance. Under this circumstance, achieving real-time detection with minimal hardware cost and low latency is preferable than emphasizing detection accuracy alone, even with some inevitable but acceptable accuracy loss. Vehicle detection from short-range aerial images actually faces various challenges. For example, Fig. 1 presents fours kinds of challenges, *blocking* usually happens around road intersection when the red traffic light is on, where a large group of vehicles are chaotically clogged. *Blurring* indicates motion blurring and it often happens on freeway where vehicles are driving with high speed. *Interference* means irrelevant but quasi-vehicle object interferes with vehicles, (*i.e.* the building and fence lying close to vehicles in Fig. 1 (c)), precipitously increasing the difficulty to distinguish them accurately. *Color variation* means the color difference among different vehicle categories. In Fig. 1 (d), the truck and the nearby cars share completely different color layout, resulting in the necessity for the detection algorithm to be able to handle color variations. All these challenges have increased the difficulty for an algorithm to keep detection accuracy while maximizing detection speed.

To achieve the best detection accuracy/speed tradeoff, we design a light-weight compressed MobileNet that best fits for vehicle detection from short-range aerial images. Vehicles in short-range aerial images share two main characteristics: ve-

[1]Y. He (yuhanghe01@gmail.com), Z. Pan, Y. Shan and L. Chen are with School of Data and Computer Science, Sun Yat-sen University. China. Corresponding author: Long Chen. chenl46@mail.sysu.edu.cn.
[2] L. Li is with depart. of electrical and computer engineering, Purdue School of Engineering and Technology, Indiana University-Purdue University Indianapolis, USA. LL7@iu.edu
[3] D. Cao is with depart. of Mechanical and Mechatronics Engineering, University of Waterloo, Canada. dongpu@uwaterloo.ca

This is the author's manuscript of the article published in final edited form as:

He, Y., Pan, Z., Li, L., Shan, Y., Cao, D., & Chen, L. (2019). Real-Time Vehicle Detection from Short-range Aerial Image with Compressed MobileNet. 2019 International Conference on Robotics and Automation (ICRA), 8339–8345. https://doi.org/10.1109/ICRA.2019.8793673

hicles are homogeneous and less size-variant but they suffer from challenges as is shown in Fig. 1. Thus it allows the detection model to cover less scales but requires the model to acquire powerful feature expressiveness. We benefit from the current development of MobileNet family network engineering and propose to re-engineer the original MobileNet v2 network by reducing the bottleneck architectures during the feature map downsampling stage but adding more bottlenecks when the feature map's spatial resolution stagnates. Our proposed compressed MobileNet enjoys much fewer FLOPs and negligible parameters increase, ensuring the fast detection performance when comparing with the original MobileNet v2. Moreover, we squash the compressed MobileNet into SSD [10] detection framework. Unlike original MobileNet v2 that extract features at different scales (exactly six scales), our compressed MobileNet extracts features at fewer scales but multiple times for an individual scale. This leads to full exploitation of the feature map results in better detection accuracy when comparing with original MobileNet v2, especially for the hard vehicle categories, such as motorcycle, bicycle and motorcycle crowded.

In sum, we make two main contributions: 1) we exploit the latest development in object detection to devise a light weight neural network that achieves the best detection accuracy/speed tradeoff for vehicle detection. 2) our proposed neural network is scalable and hardware deployment friendly, it can be easily extended to other relevant detection tasks and further optimized for more efficient performance.

II. Related Work

Object detection has gained significant progress in recent years, benefiting from the rapid development of deep convolutional neural networks (CNN). We briefly review related work from three perspectives: accuracy-oriented, speed-oriented and light weight neural network engineering.

Accuracy-oriented objected detection often consists of two stages: region proposal generation and proposal regression and classification. The serial R-CNN [22], Fast-RCNN [23] and Faster-RCNN [9] successively construct the algorithmic pipeline for two stage based detection. For region of interest (RoI) generation, R-CNN[22] depends on selective search [24], Fast-RCNN [23] and Faster-RCNN [9] have devised region proposal network (RPN) and Faster-RCNN [9] steps further to combine RPN and other components together so as to make the whole network end-to-end trainable. The first stage trains a backbone network for proposal generation while the second stage trains a head network to further adjust proposals. Thus, various following work has been proposed to either upgrade backbone network [17] or modify head network [14] to further improve the overall performance. Accuracy oriented detection algorithms are usually of low detection speed because decoupling of backbone and head network works much slower than coupling them together.

Speed-oriented detection has long attracted attention and it targets at achieving fast detection speed without much penalizing the accuracy. Alongside the two-stage detection pipelines, various attempts have already been made to reduce the computation complexity. For instance, He *et al.* [25] have proposed spatial pyramid pooling network to share computations among candidate proposals. Dai *et al.* [26] exploit fully convolutional network to speed up the inference time. Different from two-stage detection pipelines, one-stage detection algorithms have also been devised [10][11], which are comparably faster than two-stage algorithms because they finish proposal general and proposal adjustment at one time. The two most representative one-stage detectors are YOLO [11] and SSD [10]. YOLO treats object detection as a regression problem and it predicts the potential proposal bounding box location and its associated class probability within one computation. SSD [10] further improves the performance by producing proposals at different scales from different convolution layers. These one-stage detectors have been harnessed for real-time detection applications.

Light-weight network engineering tries to design light weight network architecture that maximally reduces required computation but with non-obvious or acceptable accuracy loss, by still obeying the general convolutional neural network design philosophy. In a nutshell, this kind of engineering work focuses on cutting off unnecessary multiply and add operation number within a sophisticated network architecture. Existing work [20][19][27] overwhelmingly depend on shallow and thin neural network and further simplify required computation by reformulate the convolution operation. For example, ShuffleNet [20] proposes group convolution and channel shuffle to simplify the network. MobileNet [19][27] uses depth-wise convolution and point wise convolution to release computation burden. These successful design has already achieved real-time detection even on mobile devices like smart phone. Our work builds on these design too and we will discuss them in detail in the next section. Note that other relevant model compression technologies such model pruning [28] and model binarization [29] can also be applied to shorten the detection time, yet we do not talk about them in this paper as they often lead to large accuracy loss.

vehicle detection is a classic problem is either autonomous driving or robotics domain [30][31][32]. Existing work on vehicle detection either exploit LiDAR point cloud information [30][31] or focus on vehicle detection from images captured by on-road camera [32]. No existing work exploits deep neural network for vehicle detection from short-range aerial images.

III. Preliminary and Our Approach

A. Metrics for Detector Efficiency

How to qualitatively determine the efficiency of an available detector? Currently existing work [20] [19][27][33] adopt three widely-used metrics: float-point multiplication or adds operation numbers (FLOPs), model parameter number (Params) and memory access cost (MAC). They describe the detector's efficiency from different perspectives.

To compute FLOPs for convolutional neural networks (CNNs), we assume the convolution operation is computed by a sliding window sliding the feature maps with fixed stride and the following nonlinearity is computed for

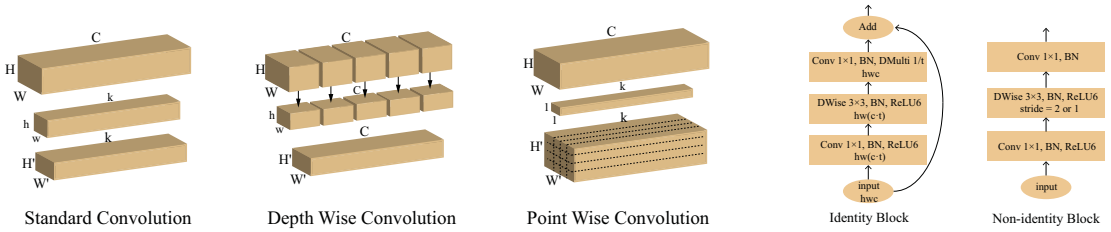

Fig. 2. Visualization of different kinds convolution operations (left side) and linear bottleneck and inverted residual connection (right side).

free. For an input feature map with height h width w and channel c_i, the kernel size is of $k \times k$ and the output channel is c_o, the corresponding FLOPs [1] is,

$$FLOPs = 2hw(k^2 c_i)c_o \tag{1}$$

FLOPs is feature spatial map resolution dependent, Larger feature map leads to higher FLOPs. Params merely depends on network architecture,

$$Params = (k^2 c_i)c_o \tag{2}$$

Memory access cost (MAC) indicates the memory/cache switch number for the detector's inference per time. It heavily depends on the capability of hardware. For simplification, we assume the output feature map is of the same spatial resolution of input feature map, then the MAC value is,

$$MAC = hw(c_i + c_o) + k^2 c_i c_o \tag{3}$$

In general, the lower of the FLOPs, Params or MAC value, the potentially more efficient of the detector. However, none of them alone guarantees the efficient performance in real scenarios. For example, MobileNet v2 [27] shares comparable FLOPs value with NASNet-A [16] but MobileNet v2 is much faster than NASNet-A. Various peripheral factors should be taken into consideration in real scenario applications, such as neural network architecture, hardware bandwidth *etc.*.

Memory access coat (MAC) has a lower bound,

$$MAC \geqslant \sqrt{2} \cdot \sqrt{hw \cdot FLOPs} + \frac{FLOPs}{2hw} \tag{4}$$

to get the lower bound, c_i must be equal to c_o, $c_i = c_o$ (please see [20]). Thus, state of the art light weight neural networks usually adopt this channel-equal guide to design networks [20] [19][27].

B. Guidelines for Light Weight Neural Network Design

The ultimate goal of light weight neural network design is to maximumly reduce model capacity (specifically, reduce FLOPs) while maximizing model expressiveness. For model capacity reduction, since the most time-consuming computation lies in convolution and standard convolution operation produces each output channel by convolving all input channels, various light weight convolutions have been devised to mitigate the computation burden, including depth wise convolution, point wise convolution, *etc.*.

Depth Wise Convolution The channel correspondence between input feature map and output feature is many-to-many in standard convolution. Depth wise convolution, on the other hand, depends on one-to-one correspondence and directly produces each output channel from its corresponding single input channel. For the same convolution in Eqn. (1), the FLOPs for depth wise convolution is $hw(2k^2 c_i)$, reducing standard FLOPs by a factor of c_i. Depth wise convolution requires the input feature map channel and output feature map channel to be the same.

Point Wise Convolution adopts 1×1 kernel and it enjoys much flexibility to expand or shrink the output feature map channels, regardless of the input feature channel number. Moreover, 1×1 kernel is hardware friendly (both mobile and embed hardware) and has been widely used in various light weight neural network design [19][27][20]. Point wise convolution often works in combination with depth wise convolution to construct "bottleneck" module (we will present it in the following subsection).

The two widely adopted convolution genres as well as the standard convolution are shown in Fig. 2 left side. In addition to these two genres, ShuffleNet [20] has introduced group convolution and channel shuffle to reduce the computation. We do not introduce them here as they are unfriendly to hardware deployment. Unilateral model capacity reduction, however, accompanies model expressiveness decrease. In defence of model expressiveness, MobileNet v2 [27] has introduced linear bottleneck and inverted residual connection.

Linear Bottleneck "Bottleneck" like network architecture derives from ResNet [15], it enables information flow and communication from different network depth, thus becomes a widely-used guide for current neural network engineering. Conventional convolution is usually followed by a batch normalization [34] and ReLU activation (in light weight neural network, ReLU6 which clips values larger than 6 is often used instead of ReLU). It works well for large neural networks. But when it comes to light weight neural network, ReLU easily leads to training collapse and information loss (we refer [27] for more theoretical proof). M. Sandler *et al.* propose to eliminate the ReLU activation before the last convolution within all bottleneck connection. Experiment result has already shown large improvement in COCO [13] object detection and other relevant tasks.

Inverted Residual Connection aims at strengthening the

[1] Note that FLOPs here treats *multiply* and *add* operation separately.

model's expressiveness by expanding the channels within the side path but return back to the original channel when it merges with the main path by elementwise add operation. The channel expansion is achieved by three specially designed sequential convolutions: a point wise convolution for channel expansion, then a depth wise convolution for further information encoding and in the end a point wise convolution for channel reduction. An intuitional illustration of linear bottleneck and inverted residual connection is show in Fig. 2 right side.

Aforementioned network engineering guidelines have been treated as basic components for modern light weight network design, including ShuffleNet [20], MobileNet V1/V2 [19][27], Xception [33], *etc.*. Our network also builds on these guidelines. In summary, we can get three main conclusions from these guidelines,

1) Linear bottleneck and inverted residual connection are the keys to increase model expressiveness. Depth wise and point wise convolution guarantee low model capacity while maintaining appropriate model expressiveness.
2) Model expressiveness stores in the feature map's spatial resolution as well as its channels, that means smaller spatial resolution can be remedied by large channels for equivalent model expressiveness.
3) Try to apply identity block because it costs minimal memory access cost (MAC).

C. Our Approach

Neural network based detection pipeline can be roughly divided into two main components: feature extraction and the followed box prediction. Feature extraction module depends on a light weight neural network that strikes the providential balance between detection accuracy and detection speed. It has occupied the largest part for the final detection performance, thus most work focus on light weight network engineering. For the box prediction, SSD pipeline [10] is the mostly used counterpart for two main reasons: the first one is that SSD belongs to one-stage detector, it is much faster than two-stage detector like Faster-RCNN [9], conforming to our real-time detection requirement. The second one is that, unlike other one-stage detectors like YOLO [11], SSD extracts features from different resolution scales, resulting in more robust feature representation and better detection performance for objects with various sizes. We thus embrace SSD as our box predictor. For SSD introduction, we refer to [10].

We build our feature extraction on MobileNet v2 [27], which has shown promising performance in vision tasks such as classification on ImageNet dataset [35], detection on COCO dataset [13] and semantic segmentation on PASCAL VOC 2012 dataset [36]. Moreover, unlike ShuffleNet [20] that is hardware implementation unfriendly because it requires frequent memory/cache with operation, MobileNet v2 is hardware-friendly and can be easily deployed on either mobile or embedded devices. Instead of directly adopting MobileNet v2, we propose a compressed version that is more suitable for vehicle detection from short-range aerial images because it holds the following two main characteristics,

4) Vehicles in short-range aerial images are homogeneous and share less size variation, requiring less spatial resolution scales for SSD feature extraction.
5) The challenges shown in Fig. 1 require large model expressiveness to handle interference, blurring and blocking problems.

TABLE I
COMPRESSED MOBILENET ARCHITECTURE AND ORIGINAL MOBILENETV2 ARCHITECTURE.

Input	Operator	t	c	n_m	n_o	s
$224^2 \times 3$	conv2d	-	32	1	**1**	2
$112^2 \times 32$	bottleneck	1	16	1	**1**	1
$112^2 \times 16$	bottleneck	6	24	2	**1**	2
$56^2 \times 24$	bottleneck	6	32	3	**1**	2
$28^2 \times 32$	bottleneck	6	64	4	**2**	2
$14^2 \times 64$	bottleneck	6	96	3	**1**	1
$14^2 \times 96$	bottleneck	6	160	3	**1**	2
$7^2 \times 160$	bottleneck	6	320	1	**3**	1
$7^2 \times 320$	conv2d 1×1	-	1280	1	**2**	1
$7^2 \times 1280$	avg pool 7×7	-	-	-	**-**	1
$1 \times 1 \times 1280$	conv2d 1×1	-	k	1	**-**	1

As a consequence, we propose a compressed MobileNet based on the aforementioned five rules. First, we achieve the feature map spatial $\times 32$ downsampling (224×224 to 7×7) via 8 bottlenecks, unlike the 16 bottlenecks used by MobileNe V2. This is achieved by reducing the bottlenecks for each spatial resolution. The fast downsampling strategy reinforces the neural network to encode feature expressiveness into feature map channels, relatively ignoring the spatial resolution. Second, the saved memory can be further used to further enhance the model expressiveness on 7×7 feature map (plateau stage) by stacking multiple identity blocks (echoes rule 2) and 3)). Third, unlike MobileNet v2 that exploits feature map at different spatial resolution scales (14×14, 7×7, 4×4, 2×2 and 1×1), compressed MobileNet merely extracts features from 14×14 and 7×7 scales. The reason is twofold: since vehicles in short-range aerial images are homogeneous and small in size, feature map with smaller spatial resolution is not that needed because smaller spatial resolution often corresponds to larger object detection in SSD detection framework due to the anchor point generation (echoes rule 4)); multiple feature extraction from 7 spatial resolution fully leverages the powerful feature expressiveness we have constructed with multiple identity blocks (responsible for challenges in 5)).

The detailed compressed MobileNet as well as the original MobileNet v2 architecture is shown in Table 1, in which t indicates depth multiplier, c indicates output channel number, n_m and n_o indicate the bottleneck number of the corresponding spatial resolution for the original MobileNet v2 and our compressed MobileNet respectively. s is the stride. We can observe that our compressed MobileNet has less bottlenecks for each spatial resolution before it reaches 7×7 resolution. On the contrary, multiple bottlenecks are

added to each of 7×7 spatial resolution feature maps (3 bottlenecks for $7^2 \times 160$ and 2 for $7^2 \times 320$) to enhance feature expressiveness.

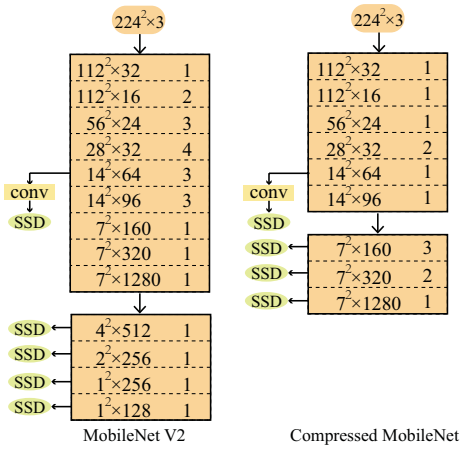

Fig. 3. *SSDLite* based object detection on MobileNet v2 and compressed MobileNet network. We can have a clear understanding towards the feature extraction network module workflow as well as the node location *SSDLite* builds on to extract features. The number in the left side indicates the intermediate feature map while in the right side indicates the corresponding bottleneck number.

We show the way *SSDLite* extracts features from our compressed MobileNet and the original MobileNet v2 in Fig. 3. *SSDLite* is a special SSD variant by changing its convolution to depth wise convolution, reducing the Params by a factor of 7 and FLOPs by a factor of 4. We can clearly see that unlike the original MobileNet v2 that extracts features from six kinds of feature maps with various scales and consecutively exploits the last $7^2 \times 1280$ feature map by adding extra convolution operator, our compressed MobileNet fully leverages the 7×7 feature maps to extract feature multiple times. Experimental result in the next section shows superiority of our proposed method.

IV. Experiment

Network Initialization We train all models with TensorFlow [37] and GPU 8 GTX 1080Ti. We pre-train our proposed compressed MobileNet on ILSVRC 2012 dataset [35] and construct four compressed MobileNet variants by differing the depth multiplier t with 1.4, 1.0, 0.75, 0.5 in descending order. The initial learning rate is 0.045, decay rate is 0.98 per epoch. The weight decay is set to 0.00004 and momentum is set to 0.9. RMSProp optimizer is adopted here. We report top-1 accuracy in Table 2, from which we can observe that our proposed compressed MobileNet ($\times 1$) have 20% more model parameters but 21% less FLOPs than its counterpart MobileNet v2. But it outperforms MobileNet v2 by nearly 1% top-1 accuracy even with less FLOPs and minimal more Params. It attests that our proposed compressed MobileNet shines in model expressiveness under limited computational budget and it fits for fast inference due to the reduced FLOPs. Note that, we do not report MAC value because it heavily depends on hardware and it is positively correlated with FLOPs and Params.

TABLE II
Performance in ILSVRC 2012 dataset.

Network	Top-1 Accu	FLOPs	Params
MobileNet v2 ($\times 1.4$)	74.7%	1170 M	6.9 M
MobileNet v2 ($\times 1.0$)	72.0%	600 M	3.4 M
Compressed MobileNet ($\times 1.4$)	75.8%	950 M	7.18 M
Compressed MobileNet ($\times 1.0$)	73.1%	487 M	4.04 M
Compressed MobileNet ($\times 0.75$)	69.8%	322 M	2.53 M
Compressed MobileNet ($\times 0.5$)	61.3%	140 M	1.35 M

The pre-trained compressed MobileNet models are further used to fine-tune *SSDLite* based vehicle detection model. We train all detection models with TensorFlow object detection API [38]. For evaluation metric, we adopt the widely-used COCO detection evaluation metric [13], that is recording the average precision (AP) for each category and the total mean average precision across all categories (mAP). For detection accuracy test, we conduct experiments on both our proposed compressed MobileNet and MobileNet v2 with various depth multiplier t and input image sizes (*i.e.* 300×300, 224×224, 196×196). Note that by observing the performance by changing the depth multiplier t and input image size, we can gain in-depth understanding on models' expressiveness under various computational budget. For detection time test, we run experiments on three representative hardwares: GPU with GTX 1080Ti, CPU with 2.8 GHz Intel Core i5 processor and mobile device with iPhone X. The mobile detector is generated by TFLite [38]. We report the frame per second (FPS) as the run-time efficiency metric. For detector training, we leverage the recommended hyper parameter setting strategy recommended by TensorFlow object detection API [38].

We have collected 5-k images from camera bounded on the roof of various urban buildings, capturing vehicles under different road conditions, motion states. Each image associates with 15 vehicles on average. We randomly divide it into 3.5-k, 0.5-k and 1-k for train/val/test respectively.

The first thing we want to figure out is the feasibility and superiority of our proposed compressed MobileNet in vehicle detection when comparing with other popular relevant methods, such as MobileNet v2 [27] and ShuffleNet [20]. To this end, we exploit the officially provided models pre-trained on ILSVRC 2012 to train vehicle detection models for MobileNet v2 and ShuffleNet v2 respectively. The depth multiplier $t = 1$ and the image input size that is taken into consideration is 224×224. The detailed result is reported in Table 3, from which we can learn that, comparing with two other methods, our proposed compressed MobileNet enjoys the least FLOPs but outperforms MobileNet v2 by 0.02 mAP improvement and achieves comparable performance with ShuffleNet v2 [20]. The largest advantage of compressed MobileNet is that it runs much faster in hardware deployment and the poorer the hardware is, the relatively faster it appears to be when comparing with either MobileNet v2 or ShuffleNet v2. ShuffleNet v2 performs inferior in low-level hardware although it achieves relatively high detection accuracy. This attests ShuffleNet is unfriendly to hardware.

TABLE III

COMPARISON OF OUR PROPOSED METHOD AND OTHER METHODS.

Network $t = 1$	FLOPs	Params	Evaluation Result (mAP and AP, larger is better)								FPS		
			mAP	car	bus	truck	motorcycle	bicycle	motor crowd	bicy crowd	GPU	CPU	Mobile
ShuffleNetv2 (224×)	650.8 M	3.40 M	0.85	0.98	0.92	0.80	0.71	0.85	0.94	0.76	100	22	8
MobileNetv2 (224×)	601.8 M	3.47 M	0.83	0.98	0.91	0.77	0.67	0.83	0.95	0.73	98	23	10
Comp. Mobi (224×)	487.3 M	4.02 M	**0.85**	0.98	**0.93**	0.80	**0.72**	0.83	0.95	0.75	**110**	**31**	**15**

TABLE IV

COMPARISON OF VARIOUS COMPRESSED MOBILENET VARIANTS.

Network (size, t)	FLOPs	Params	Evaluation Result (mAP and AP, larger is better)								FPS		
			mAP	car	bus	truck	motorcycle	bicycle	motorcrowd	bicycrowd	GPU	CPU	Mobile
MobileNet v2 (224×, 1.4)	1.16 B	6.06 M	0.84	0.98	0.92	0.78	0.69	0.83	0.95	0.76	90	21	7
MobileNet v2 (224×, 1.0)	601.84 M	3.47 M	0.83	0.98	0.91	0.77	0.67	0.83	0.95	0.73	98	23	10
MobileNet v2 (224×, 0.75)	410.21 M	2.21 M	0.80	0.97	0.89	0.76	0.62	0.80	0.91	0.69	112	27	16
MobileNet v2 (224×, 0.50)	183.11 M	1.21 M	0.76	0.92	0.85	0.72	0.58	0.76	0.90	0.65	130	35	25
Comp. Mobi (300×, 1)	939.3 M	4.02 M	0.85	0.98	0.92	0.81	0.73	0.82	0.95	0.74	104	26	10
Comp. Mobi (224×, 1)	487.3 M	4.02 M	0.85	0.98	0.93	0.80	0.72	0.83	0.95	0.75	110	31	15
Comp. Mobi (196×, 1)	440.4 M	4.02 M	0.82	0.97	0.90	0.77	0.70	0.81	0.92	0.72	120	35	21
Comp. Mobi (224×, 1.4)	951.1 M	7.18 M	**0.86**	0.98	0.94	0.82	0.73	0.85	0.95	0.77	106	27	13
Comp. Mobi (224×, 0.75)	321.63 M	2.53 M	0.83	0.97	0.90	0.79	0.70	0.81	0.93	0.74	124	37	24
Comp. Mobi (224×, 0.50)	**140.5 M**	**1.35 M**	0.80	0.92	0.89	0.76	0.66	0.79	0.90	0.71	**147**	**49**	**32**

(a) detection with motion blurring (b) inter-category crowdedness (c) dense environment (d) non-vehicle interference

Fig. 4. Qualitative visualization of sample detection result images. The attention areas are drawn with yellow dotted lines. Zoom in for better visualization.

On the contrary, our proposed compressed MobileNet obtains the best detection accuracy/speed tradeoff in terms of vehicle detection from short-range aerial images. Note that we do not report MobileNet v1 and ShuffleNet v1 performance as they have already been shown inferior to their v2 counterparts [27][20], we find this applies to vehicle detection too.

To further test the robustness of compressed MobileNet, we conduct experiment on different compressed MobileNet variants by changing the depth multiplier t and the input image spatial resolution. To this end, we create the first set of variants by changing the depth multiplier but fixing the spatial resolution as 224×224. After that, we reversely create the second variants by changing the spatial resolution but fixing the depth multiplier as $t = 1$. For direct comparison, we also report the MobileNet v2 performance of 224×224 input size under various depth multiplier. The detailed result is shown in Table 4. Three conclusions can be summarized from this table: with the decreasing of depth multiplier t, MobileNet v2 suffers more accuracy loss (mAP) than the compressed MobileNet counterpart. Under the same depth multiplier, compressed MobileNet achieves higher FPS than MobileNet v2. We have further noticed compressed MobileNet performs better on hard object detection than MobileNet v2 under small depth multiplier. For example, with the depth multiplier $t = 0.5$, the average precision of motorcycle and bicycle crowd obtained by compressed MobileNet is at least 5% higher than the value obtained by the corresponding MobileNet v2.

We further provide qualitative detection visualization in Fig. 4. The detection result is generated by compressed MobileNet with input size 224×224 and multiplier $t = 1$. We can see from this figure that our proposed compressed MobileNet can successfully tackle the challenges presented in Fig. 1, including motion blurring, non-vehicle object interference, inter-category or intra-category crowdedness. For example, the black car largely occluded by light pole has been successfully detected (the right most figure in Fig. 4).

In summary, our proposed compressed MobileNet strikes great balance between detection accuracy and speed. It fully exploits the internal characteristics of short-range aerial images and builds on the state of the art light weight network engineering guidelines, outperforming the popular MobileNet v2 by a large margin in terms of accuracy and speed. Moreover, compressed MobileNet is self scalable and mobile or embedded device deployment friendly.

V. ACKNOWLEDGEMENT

This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1305002 and the National Natural Science Foundation of China under Grant 61773414.

REFERENCES

[1] P. Bergmann, R. Wang, and D. Cremers, "Online photometric calibration of auto exposure video for realtime visual odometry and slam," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, pp. 627–634, April 2018.

[2] H. Lim, J. Lim, and H. J. Kim, "Real-time 6-dof monocular visual slam in a large-scale environment," in *ICRA*, 2014.

[3] L. Chen, Y. He, J. Chen, Q. Li, and Q. Zou, "Transforming a 3d lidar point cloud into a 2d dense depth map through a parameter self-adaptive framework," *Trans. on Intelligent Transportation Systems (TITS)*, 2017.

[4] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications," in *ICRA*, 2017.

[5] G. Izatt, G. Mirano, and E. Adelson, "Tracking objects with point clouds from vision and touch," in *ICRA*, 2017.

[6] C. Wang, H. K. Galoogahi, C.-H. Lin, , and S. Lucey, "Deep-lk for efficient adaptive object tracking," in *ICRA*, 2018.

[7] L. Chen, L. Fan, G. Xie, and A. Nuetcher, "Moving-objects detection from consecutive stereo pairs using slanted plane smoothing," *Trans. on Intelligent Transportation Systems (TITS)*, 2017.

[8] L. Chen, X. Hu, T. Xu, H. Kuang, and Q. Li, "Turn signal detection during nighttime by cnn detector and perceptual hashing tracking," *Trans. on Intelligent Transportation Systems (TITS)*, 2017.

[9] S. Ren, K. He, , R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2016.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[12] K. Ivan, D. Tom, A. Neil, F. Vittorio, A.-E.-H. Sami, K. Alina, R. Hassan, U. Jasper, P. Stefan, K. Shahab, M. Matteo, P.-T. Jordi, V. Andreas, B. Serge, G. Victor, Gupta, Abhinav, S. Chen, C. Gal, C. David, F. Zheyun, N. Dhyanesh, and M. Kevin, "Openimages: A public dataset for large-scale multi-label and multi-class image classification," *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[13] 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 *ECCV*, 2014.

[14] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head r-cnn: In defense of two-stage object detector," in *arXiv preprint arXiv:1711.07264*, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[16] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[17] B. Li, T. Wu, L. Zhang, and R. Chu, "Auto-context r-cnn," in *CVPR*, 2018.

[18] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *CVPR*, 2018.

[19] 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.

[20] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," *arXiv preprint. arXiv: 1807.11164*, 2018.

[21] T. Zhang, G.-J. Qi, B. Xiao, and J. Wang, "Interleaved group convolutions for deep neural networks," in *ICCV*, 2017.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[23] R. Girshick, "Fast r-cnn," in *ICCV*, 2015.

[24] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014.

[26] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: object detection via region-based fully convolutional networks," in *NIPS*, 2016.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *arXiv preprint. arXiv: 1801.04381*, 2018.

[28] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l0 regularization," in *ICLR*, 2018.

[29] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," in *arXiv preprint arXiv:1602.02830*, 2016.

[30] S. Verma, Y. H. Eng, H. X. Kong, H. Andersen, M. Meghjani, W. K. Leong, X. Shen, C. Zhang, M. H. A. Jr, and D. Rus, "Vehicle detection, tracking and behavior analysis in urban driving environments using road context," in *ICRA*, 2018.

[31] X. Du, M. H. A. Jr., S. Karaman, and D. Rus, "A general pipeline for 3d detection of vehicles," in *ICRA*, 2018.

[32] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas, "A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera," in *ITS Conference*, Sep. 2012, pp. 975–982.

[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift. training deep neural networks," in *ICML*, 2015.

[35] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2019.

[36] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, 2014.

[37] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[38] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *CVPR*, 2017.