Fast and Lightweight Person Detector for Unmanned Aerial Vehicles

Lucas Steinmann[1] Lars Sommer[2,1] Arne Schumann[1] Jürgen Beyerer[1,2]

[1]Fraunhofer IOSB
Fraunhoferstraße 1
Karlsruhe, Germany

[2]Vision and Fusion Lab
Karlsruhe Institute of Technology
Karlsruhe, Germany

{firstname.lastname}@iosb.fraunhofer.de

***Abstract*—The surge of affordable, small unmanned aerial vehicles (UAVs) opens up many new possibilities for applications, which often rely on locating relevant objects within a scene. Besides accurate object detection, inference time is a key factor for real-world applications, as fast models can be deployed directly on mobile platforms, such as UAVs. However, current state-of-the-art detectors are typically not suited for this type of deployment. In this work, we propose a fast and lightweight on-board detector for the task of person detection in UAV imagery. We perform several adaptations to speed up the SSD detector, which comprises our base detection framework. Adjustments of the network include lightweight backbone architecture, filter pruning, and tuning of implementation details. A thorough evaluation of all factors is carried out on two mobile platforms. Our proposed detector is about 24 times faster on Jetson AGX platform compared to the baseline, while the detection accuracy is only slightly impacted. Furthermore, we demonstrate the robustness of our proposed detector on unseen scenarios in a cross domain setting.**

***Index Terms*—person detection, lightweight CNN, UAV imagery**

I. Introduction

In recent years, the surge of affordable, small unmanned aerial vehicles (UAVs) opened up many new possibilities for applications, such as disaster control, rescue missions, and security. In many of these cases, a degree of on-board image analysis is required to provide low-latency information for autonomous functions or to better utilize low-bandwidth transmission. Often the most important on-board task is object detection, as several more complex analysis tasks or decision functions are based on the locations of relevant objects in the scene. Examples for this are the transmission of image regions with observed objects in low-bandwidth situations or autonomous functions, such as following, counting, or zooming in on objects.

In this work, we propose a fast person detection approach suitable for deployment and online processing on-board a UAV. Our approach focuses on persons, as this is the most relevant object category in most applications, but can be extended towards additional object classes with little adaptation. The detector is based on convolutional neural networks (CNNs), since landmark CNN models, such as Faster R-CNN [1], SSD [2], and YOLOv2 [3] achieve top accuracies on current benchmark datasets. However, while these algorithms achieve real-time inference speeds on server-grade GPUs, they are unsuitable for real-time processing in mobile platforms, such as UAVs, where computational resources are severely limited. In our work, we focus on adapting the SSD detector to these additional requirements of UAV on-board computation. The convolutional backbone network of our detector is based on the Pelee architecture [4], a network specifically designed for fast inference of classification tasks. We further reduce the number of parameters and computational costs by applying the filter pruning strategy proposed in [5]. To analyze the improvements in latency and throughput of several important design decisions, as well as key implementation details, we measure the inference speed of our model on the Jetson TX2 and AGX platforms. Our final detector is about 26 and 24 times faster on Jetson TX2 and Jetson AGX, respectively, while the detection accuracy is only slightly worse compared to the baseline approach. We further demonstrate the robustness of our proposed detector on unseen scenarios from cross domain datasets. Our contributions can be summarized as:

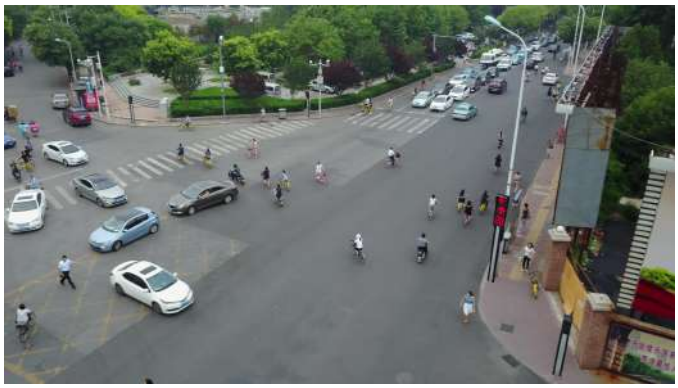

Fig. 1. Example image of the VisDrone dataset [15] used for validating our proposed detector. The image shows a complex scene with persons at varying positions, view angles, and scales.

- We propose a new, fast person detector for on-board processing on mobile platforms.

- We give a detailed analysis of model design choices with regard to inference time across several UAV recordings of scenes with high visual diversity.
- The impact of deployment environment and implementation details on the inference time is systematically evaluated on two different mobile platforms.

II. Related Work

Object detection has been a major focus of the computer vision community for many years. Recently, several deep learning detection frameworks have been developed, such as Faster R-CNN [1], SSD [2], and YOLOv2 [3]. Most of these approaches can be categorized as either one-stage (*e.g.* YOLO, SSD) or two-stage approaches (*e.g.* Faster R-CNN). While past research mainly focuses on improving the detection accuracy (*e.g.* by exploiting feature pyramids), lightweight and computational efficient networks for the use on mobile platforms experience growing interest. In the following, we restrict our literature review on work focusing on improving inference time.

Deep learning detection frameworks typically employ large CNNs as feature extractor, followed by layers for regression of bounding boxes and classification of object classes. Hence, most approaches for improving inference speed aim at reducing the number of parameters and computational operations in the feature extractor to speed up the detector. In recent years, a multitude of novel computation-efficient CNN networks, such as Mobilenet (V2) [11], [12], SqueezeNet [9], ShuffleNet (V2) [13], [14], and PeleeNet [4], have been proposed for classification tasks. Several subsequent works demonstrate the potential of these lightweight architectures as feature extractors for object detection.

Howard *et al.* [11] employ MobileNet as backbone architecture for SSD and achieve comparable detection accuracy on MS COCO while the parameter and floating point operations (FLOP) count is significantly reduced. MobileNet uses depthwise separable convolutions to greatly decrease the number of parameters and operations, while trading a fair amount of accuracy compared to VGG16. ShuffleNet [13] exploits pointwise group convolution and channel shuffle for cross-group information flow to realize a $13\times$ more efficient but equally accurate model compared to AlexNet. While separable convolutions require significantly fewer parameters, they are not supported as well as regular convolutions in deep learning frameworks and hardware. The ShuffleNet architecture is applied in [10] for fast vehicle detection in aerial imagery. SqueezeNet [9] is comprised of multiple *fire modules*, which include a bottleneck realized by pointwise convolutions as well. This enables drastically smaller model sizes while maintaining an AlexNet-level image classification accuracy. Wu *et al.* [20] achieve state-of-the-art accuracy on the KITTI dataset by employing SqueezeNet as feature extractor. PeleeNet [4], which we employ as feature extractor in this work as well, was shown to outperform MobileNet-SSD on PASCAL VOC and MS COCO.

TABLE I
NETWORK STRUCTURE OF PELEENET AND OUTPUT DIMENSION OF EACH STAGE FOR AN INPUT RESOLUTION OF 1280×704 PIXELS. THE OUTPUT OF THE LAST LAYERS OF THE HIGHLIGHTED STAGES ARE USED AS FEATURE MAPS, *i.e. stage2_tb/relu*, *stage3_tb/relu* AND *stage4_tb/relu*.

Network Stage	Output Dimension
Stage0 – Stem Block	$320\times176\times64$
Stage1 – $3\times$ Two-Way Dense Layer	$320\times176\times128$
Stage2 – $4\times$ Two-Way Dense Layer	$160\times88\ \times256$
Stage3 – $8\times$ Two-Way Dense Layer	$80\times44\ \times512$
Stage4 – $6\times$ Two-Way Dense Layer	$40\times22\ \times704$

III. Method

In this section, we describe the proposed detection algorithm applied for fast person detection. First, we introduce the utilized deep learning based detection framework and our main adaptations performed to optimize inference time. Furthermore, optimizations of the runtime environment of the proposed detection algorithm are discussed.

A. Detector

We adopt the Single Shot Detector (SSD) as base detection framework because of its good trade-off between detection accuracy and inference time. SSD is a fully convolutional network, which exploits a CNN base network for feature extraction. Multiple convolutional layers are employed as feature maps to predict detections at multiple scales. For this, default boxes centered at each feature map position are used as reference for bounding box regression.

To speed up the detector, we replace the standard base network with PeleeNet. Unlike recent lightweight architectures such as MobileNet and ShuffleNet, PeleeNet only consists of standard components, which allows for straightforward deployment. PeleeNet, which is inspired by DenseNet, mainly comprises two building blocks (see Table I). An initial building block termed *Stem block* enhances the feature expression ability in a computational efficient manner. Then, a sequence of so called *Two-Way Dense Layers* is used for feature extraction. Various kernel sizes are employed in order to achieve different scales of receptive fields, whereby 1×1 bottleneck convolutions are applied to reduce the number of subsequent 3×3 convolutions. Hence, the number of parameters and computational operations in PeleeNet are much reduced compared to most other CNN architectures.

As base networks are pre-trained for classification on benchmark datasets comprising a large number of classes, these networks are often oversized for detection tasks with only a few classes. To remove filters with redundant information, which are dispensable for our task, we apply the one-shot pruning strategy proposed by Li *et al.* [5]. The applied one-shot pruning initially computes the ℓ_1-norm for every filter f_i within a convolutional layer as $\sum_{j=0}^{n_i}|f_{ij}|$ where f_{ij} is the j^{th} weight of filter f_i. For every convolutional layer, filters are sorted according to the ℓ_1-norm, which proved to be a good heuristic to judge the usefulness of a particular filter. Then, a fixed percentage of filters with the lowest ℓ_1-norm is removed

from the network. Finally, the condensed network is re-trained any knowledge lost during the pruning process.

SSD generally exploits shallow layers with high spatial resolution to detect small objects, while deeper layers are used for nearly image-filling objects. While this procedure results in improved performance on common benchmark datasets, it is not practical for many real-world applications, such as person detection in UAV imagery. To account for the characteristics of UAV imagery acquired at altitudes in the range of 10m to 50m, we restrict the person prediction to shallow layers. Thus, only *stage2_tb/relu*, *stage3_tb/relu* and *stage4_tb/relu* are used as feature maps. Removing all deeper layers further reduces the computational costs.

To further account for the characteristics of UAV imagery, we adapt the default box settings. As the detection accuracy depends on the default box sizes used for bounding box regression, we adapt the default box sizes to the typical range of person sizes in the training data.

B. Runtime environment

For deployment of our detector, we compare the two mobile-suitable platforms NVIDIA Jetson TX2 and Jetson AGX. Several optimization strategies to optimize inference of deep learning applications on these platforms are available. Both Jetson TX2 and Jetson AGX can run with different power modes, *e.g.* by enabling the maximum CPU and GPU frequency. However, this option strongly depends on the respective application. The TensorRT fast inference library enables optimized inference for different deep learning frameworks, such as Caffe. By using TensorRT during deployment, instead of the original deep learning framework, the inference time can be notably reduced. In addition, TensorRT offers out-of-the-box INT8 quantization and FP16 precision implementations of common layers for deployment, which can further speed up the inference. Moreover, TensorRT includes extensive profiling tools, which allow profiling each layer with minimal overhead. The profiling results reveal that the SoftMax implementation was noticeably slower compared to the rest of the network. We thus replaced the layer with a simple implementation of the SoftMax function as a CUDA kernel and thereby improved the network's execution time.

IV. Experimental Analysis

A. Implementation Details

For training all networks, we employ the original Caffe SSD implementation [19]. Each network is trained for 30,000 iterations with an initial learning rate of 0.01 using *Stochastic Gradient Descent (SGD)*. The learning rate is decreased by a factor of 10 after 10k, 15k, 20k, and 25k iterations. The accumulated batch size is set to 32. For our pruned network, we remove 50% of the filters for each convolution of the *Two-Way Dense Blocks* as described in Section III-A. The pruned network is then retrained with the same settings. For all experiments, we employ three feature maps and the default box settings as described in Section III-A.

To train networks robust to a large variety of scenarios, we use subsets of four publicly available datasets as training data. The *VisDrone* [15] dataset contains 263 video sequences recorded by a UAV. The dataset possesses a large diversity of different scenes ranging from urban to suburban areas in 14 different cities across China. For our training, only video sequences with annotations of pedestrians or persons are considered. The *Mini-drone video dataset* [17] comprises 38 video sequences captured in full HD resolution with a Phantom 2 Vision+. The dataset entails different surveillance scenarios of a parking lot. Video sequences containing at least one person are added to our training data, whereby video sequences with missing annotations of persons are filtered out. *MOT2017* [18] is comprised of seven videos with annotated persons showing different scenarios, such as pedestrian street at night, people walking around a large square or busy shopping mall. *CityPersons* [16] is a subset of CityScapes, which contains images recorded by a camera mounted on a driving car. The images are acquired in different cities across Germany and Zurich, Switzerland. All images are down scaled to 1280×704 pixels. Thus, we force the trained networks to learn to predict even small persons. Images containing large persons whose height clearly exceed the height of persons in videos acquired by UAVs at an altitude of about 10m or more are removed. In case of VisDrone, images with a width above 1400 pixels are cropped into tiles of 1280×704 pixels.

B. Evaluation

We select two video sequences from the VisDrone validation set to quantitatively evaluate the performed experiments. The selected video sequences comprise camera perspectives and flying altitudes suited for the task of person detection by a UAV. Examples of both scenes are shown in Figure 2. To measure the detection accuracy, we use average precision (AP) as evaluation metric. For this, detections with an Intersection over union (IoU) overlap with a ground truth box above 0.5 are considered as true positives. We use the original image sizes of 1344×756 pixels and 2688×1512 pixels, respectively, as input for measuring the detection accuracy. Inference time is reported in frames per second (FPS) averaged over 1000 forward passes. As the inference time clearly differs for the different image resolutions, we used an input image size of 1280×704 pixels. The inference time is measured on two different mobile platforms: NVIDIA Jetson TX2 and NVIDIA Jetson AGX. Note that TensorRT version 5.0 and MAX-N power preset was used for all experiments if not stated otherwise. Due to preliminary experiments, we set the IoU-threshold of the non-maximum suppression step for all experiments to 0.5. Hence, the number of missed detections due to occlusions by nearby persons is reduced.

First, we analyze the impact of the utilized base network. As baseline, we employ VGG16 as base network, which is used by default. For this, *conv4_3*, *conv6*, and *fc7* are used as feature maps, which offer the same feature map resolution as the convolutional layers used in case of PeleeNet. Layers succeeding *fc7* are removed from the base network. The AP

Fig. 2. Example images of the VisDrone dataset used for evaluation. Qualitative detections (green) indicate the good detection accuracy.

Fig. 3. Qualitative examples on MultiDrone dataset [21] (left) and self-recorded images (right) demonstrate the good detection accuracy across domains.

and inference times are reported in Table II. Though the detection accurcay is good, the inference time is poor on both platforms.

Replacing VGG16 by PeleeNet results in a speed up by a factor of almost 10, while the detection accuracy only drops by 2.5% in AP. Reason for this is the lightweight architecture of PeleeNet, which requires considerably fewer computational operations. Further experiments showed that including *stage4_tb/relu* only has a minor impact on the detection accuracy. We therefore omit this feature map and remove all layers after *stage3_tb/relu*, which further reduces the number of parameters and computational operations. This leads to an improved inference speed by about 10% with no significant drop in detection accuracy. Finally, we remove redundant filters by applying the pruning scheme introduced in SectionIII-A. Eliminating 50% of filters within the *Two Way Dense Blocks* slightly increase the number of FPS while the detection accurcay remains unchanged. Further strategies to optimize the detection accuracy showed no significant impact. Upsampling features of deep layers by applying deconvolutional layers as described in [6] shows only minor impact on the AP, but the inference time gets clearly worse. Employing focal loss [7] does not considerably affect the detection accuracy either.

Next, we analyze the impact of purely implementation specific optimization techniques on the inference time (see

TABLE II
COMPARISON OF DIFFERENT BACKBONE NETWORK ARCHITECTURES ON THE SELECTED VISDRONE SEQUENCES.

Architecture	AP (%)	Inference Speed (FPS)	
		Jetson TX2	Jetson AGX
VGG16	73.2	1.1	3.4
Pelee	70.7	9.8	30.3
Pelee short	70.2	10.9	33.0
Pelee short$_{\alpha=0.5}$	70.2	13.2	35.4

TABLE III
IMPACT OF IMPLEMENTATION SPECIFIC CHANGES ON INFERENCE SPEED AND DETECTION ACCURACY; *REPLACED TENSORRT SOFTMAX; OC = OVERLOCKED (MAX-N); NO OC = MAX-P/15W

Caffe	TensorRT	SoftMax*	FP32	FP16	INT8	AP (%)	Inference Speed (FPS)			
							Jetson TX2		Jetson AGX	
							no oc	oc	no oc	oc
✓			✓			69.6	3.4	3.4	6.5	7.8
	✓		✓			70.2	13.2	16.1	35.4	52.9
	✓	✓	✓			70.2	15.9	19.8	36.4	58.3
	✓	✓		✓		70.2	**19.4**	**28.9**	40.4	81.6
	✓	✓			✓	63.2	15.2	19.8	**45.0**	**99.0**

Table III). Employing the Caffe framework for deployment results in poor inference time. We only achieve 3.4 FPS and 6.5 FPS on Jetson TX2 and Jetson AGX, respectively. The number of FPS slightly increases by setting the power mode to MAX-N. As the regular Caffe framework is not optimized for deployment, we use the TensorRT environment to execute our deep learning based person detector. Due to the including inference optimizer, the number of FPS is increased by a factor of up to 6.8 on Jetson AGX. Note that the AP slightly differs due to the different deployment environment and the corresponding layer implementations. Utilizing TensorRT's extensive profiling tools indicates that the applied SoftMax implementation is a bottleneck of the network. Compared to other layers of the network, SoftMax is noticeably slower. Hence, we replace the TensorRT implementation with a naive CUDA implementation of the SoftMax function leading to slightly improved inference time. The highest impact on the inference time exhibits the use of FP16 precision implementations of common layers for deployment. The number of FPS is increased by 22% and 11% on Jetson TX2 and Jetson AGX, respectively or 46% and 40% when overclocked, while the detection accuracy is not affected. Using INT8 quantization results in an even larger speed up. However, the detection accuracy drops by 7.0% in AP.

Qualitative detection results highlighted with green boxes are depicted in Figure 2. Even small persons, partially occluded persons and persons on bicycles are mostly detected. Nevertheless, there exist some missed detections, which exhibit room for improvement, *e.g.* post-process detections by a tracker.

Finally, we evaluate the cross domain applicability of our proposed detector. For this purpose, we employ two different datasets: the publicly available MultiDrone dataset [21] and an own dataset. Both dataset contain images with persons recorded by UAVs. Exemplarily detection results given in Figure 1 demonstrate the good transferability of our detector to unseen scenarios with different viewing perspectives and resolutions.

V. Conclusion

We proposed a fast and lightweight detector for the task of person detection in UAV imagery. For this purpose, we adjusted the SSD detector used as base detection framework with regard to inference time and detection accuracy. Replacing the computational expensive base network by PeleeNet – a lightweight CNN architecture – resulted in a speed up of almost factor 10 on two mobile platforms. The inference time was further improved by focusing only on relevant feature maps and filter pruning to reduce redundant filters. Besides the employed detector architecture, we also focused on aspects of the deployment environment to optimize the inference time. Setting the power mode to MAX-N, exploiting TensorRT instead of the Caffe framework, modifying the SoftMax layer, and using FP16 precision implementations of common layers lead to a speed up of more than 100% while retaining the detection accuracy. By employing four different datasets for training, our proposed detector achieves robust detections on unseen scenarios, as demonstrated qualitatively on cross domain datasets. In future work, we will focus on post processing the detection output by an online tracker to reduce false positive and false negative detections.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu and A. C. Berg. SSD: Single Shot Multibox Detector, In ECCV, pages 21-37, 2016.

[3] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In CVPR, 2017.

[4] R. J. Wang, X. Li, and C. X. Ling. Pelee: A Real-Time Object Detection System on Mobile Devices. In NIPS 31, pages 1967-1976, 2018.

[5] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In ICLR, 2017

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

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal Loss for Dense Object Detection. In PAMI, 2017.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft COCO: Common objects in context. In ECCV, pages 740-755, 2014.

[9] 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.

[10] S. M. Azimi. ShuffleDet: Real-Time Vehicle Detection Net- work in On-board Embedded UAV Imagery. In ECCV, 2018.

[11] A. G. Howard, M. Zhu, Bo 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.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In CVPR, pages 4510-4520, 2018.

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In CVPR, 2018.

[14] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In ECCV, 2018.

[15] P. Zhu, L. Wen, X. Bian, L. Haibin, and Q. Hu. Vision meets drones: A challenge. arXiv preprint arXiv:1804.07437, 2018.

[16] S. Zhang, R. Benenson, and B. Schiele. Citypersons: A diverse dataset for pedestrian detection. In CVPR, 2017.

[17] M. Bonetto, P. Korshunov, G. Ramponi, and T. Ebrahimi. Privacy in mini-drone based video surveillance. In 2015 11th IEEE FG, volume 4, pages 1-6, 2015.

[18] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A Benchmark for Multi-Object Tracking. arXiv preprint arXiv:1603.00831, 2016.

[19] Caffe SSD Repository, https://github.com/weiliu89/caffe/tree/ssd/, 30-04-2019.

[20] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In CVPR Workshops, 2017.

[21] The MultiDrone Dataset, https://multidrone.eu/multidrone-public-dataset/, 30-04-2019.