Efficient Embedded Deep Neural-Network-based Object Detection Via Joint Quantization and Tiling

***Abstract*—Embedded visual AI is a growing trend in applications requiring low latency, real-time decision support, increased robustness and security. Visual object detection, a key task in visual data analytics, has enjoyed significant improvements in terms of capabilities and accuracy due to the emergence of Convolutional Neural Networks (CNNs). However, such complex paradigms require heavy computational resources that prevent their deployment on resource-constrained devices, and in particular, impose significant constraints in possible hardware accelerators geared towards such applications. In this work therefore, we investigate how a combination of techniques can lead to efficient visual AI pipelines for resource-constrained object detection. In particular we leverage an efficient search strategy based on a combination of pre-processing mechanisms, that reduce the processing demands of deep network as a counter measure for potential accuracy reduction caused by quantization. The proposed approach enables the detection of objects in higher resolution frames using quantized models, while maintaining the accuracy of full-precision CNN-based object detectors. We illustrate the impact on the accuracy and average processing time using quantization techniques and different tiling approaches on efficient object detection architectures; as a case study, we focus on Unmanned-Aerial- Vehicles (UAVs). Through the proposed methodology, hardware accelerator demands are thereby reduced, leading to both performance benefits and associated power savings.**

I. Introduction

Advances in deep-learning-based scene understanding, and convolutional neural networks (CNNs) in particular, provide remarkable opportunities for computer vision based applications [1][2][3]. In such scenarios, vision systems are deployed on resource-constrained embedded devices that support high-resolution cameras for applications beyond photography, such as environmental and infrastructure monitoring applications, search and rescue, and surveillance amongst others [4][5][6].

Deep learning algorithms are computationally very intensive, and not suitable for on-board processing on such devices such as remotely operated vehicles (ROVs) due to battery/power constraints and are usually processed via the cloud. For certain scenarios however, where the system operates in remote areas with limited connectivity, this can result in unwanted response latencies which can degrade performance, a potentially catastrophic scenario in safety-critical applications. Thus, processing information at the edge can not only eliminate unwanted lag issues but also partially handle security problems since the data is not transmitted and sensitive data cannot be intercepted.

The problem is amplified especially when considering distant objects; for example, detecting objects on ground using top-view images from unmanned aerial vehicles (UAVs) flying in higher altitudes is even more challenging due to the very small size that the objects appear at. Moreover, in a contradicting manner, deep learning based detectors usually operate on lower resolutions to reduce the computation for on-board processing, which is not feasible for detection at higher altitudes since there is significant information loss due to image downsizing. That, in turn, degrades detection accuracy. Evidently, there is a need for processing high resolution images more efficiently, and particularly when targeting small sized objects. Existing deep learning based object detection algorithms have achieved state of the art results [7][8][9]. However, they are not suitable for on-board processing on UAV platforms because of computational constraints, power consumption, memory limitations and lack of custom hardware [10]. Recent works for implementing CNN-based techniques on resource constrained devices, employ techniques such as quantization [11], network pruning [12], compression [13] and efficient network design [14]. These have focused on decreasing the computational complexity of the CNN itself while maintaining the full precision/network accuracy. These methods work mostly on small and fixed image sizes and do not consider applications which necessitate processing higher resolution images. There is still however a lot of room for improvement in CNN architecture engineering as well as data reduction techniques to further increase the efficiency for such applications.

Our contribution therefore, focuses on combining different methods for efficiently realizing deep convolutional neural network for object detection on a resource constrained device. We consider a typical visual pipeline [15] and introduce an intelligent data selection technique from our previous works[16], [17], to operate in combination with quantized single-shot detector. Specifically, in this work, we study the effect of quantization [11] on the proposed visual pipeline. We observe that the quantization process coupled with aggressive resizing of larger images to fit a smaller receptive field degrades accuracy. By incorporating however, an intelligent selection process that does not decrease the image resolution, the original accuracy is maintained while there is an overall improvement in inference time. We validate our studies via experiments carried out on a Jetson TX2, showing that the proposed combination of approaches managed to improve accuracy over 20% while maintaining comparable inference time.

II. Related Work

There are mainly two mainstream approaches for such detectors: 1) Region-based Detectors [18][2] and 2) Single-Shot Detectors [19][9]. Region-based detectors such as Faster

RCNN [18] are two staged approaches, that first propose regions for processing and then perform detection which makes them compute intensive for embedded platforms. Single-shot detectors, like YOLO [19], employ a single CNN for the whole task, which is more amicable for real-time execution [20]. In our visual pipeline, we adopt a YOLO-like custom single shot detector, which is highly optimized for low latency with low memory footprint [21].

Furthermore, using CNN based object detectors in resource-constrained devices such as UAVs, has been of high interest in recent years. In [22], the authors use UAV on-board sensors in order to intelligently decrease the number of region proposals on an existing CNN. Although there is a significant speed-up of the performance, the detector achieves just 2.3 frames-per-seconds on Jetson TX2. In [23], a framework based on YOLO is trained to recognize airplanes in fixed sized aerial images of 448×448. However, the computational budget and latency of the detector is not addressed and the system is assumed to run on a workstation. Works such as [24] use cloud computing to achieve real-time object detection for a UAV but moving the computation to the cloud can introduce unpredictable lag which can hinder the application performance. [25] embeds a light weight super resolution model to enhance image quality acquired from UAVs at higher altitudes but assumes the target is equipped with a Qualcomm neural processing chip which does not support all existing convolutional operations resulting in decreased accuracy. [26] integrates concepts of R-CNN [18] and YOLO [19] proposing a new model and an inducing layer to optimize the model and speed up convergence. Works such as [27], use a sliding-window-based detector and use a fixed region of interest to remove false positives reducing the necessary computations. However, this still does not solve the problem of having to scale the detection process to larger resolution images. To this end, in designing custom resource-constrains accelerators typically designers are focusing on decreasing the computational complexity of the CNN by adopting quantization.

Quantization is an approach used to speed up neural networks, by employing integer and/or lower-precision arithmetic for fast inference with tolerable accuracy loss. Quantizing neural networks can reduce weights, activations and sometimes gradients down to 8 or even 1 bit representation hence shrinking the model size and accelerate inference especially in customized hardware accelerators [28]. Compressing a neural network by quantization not only reduces memory requirements but also power since the network is doing millions of multiplications and additions at much lower precision and the on- and off-chip data movement is reduced with the use of lower-bit quantized data. In this work, we are using an 8-bit quantization technique on both the input image and the weights of the CNN.

III. Proposed Visual Pipeline

A. Efficient Object Detector Architecture

In our previous work [21], we developed a specialized CNN architecture, referred to as DroNet[1], designed for resource constrained devices, to accelerate the inference with minimal impact on the achieved accuracy. To this end, we designed different structures with respect to the number of filters, number of layers, image size, number of convolution and pooling layers and explored the impact on the performance. The proposed model, was able to outperform the Tiny-YOLOv2, a smaller version of the state-of-the-art YOLOv2 [29] detector using the proposed weighted score metric, which combines the frames-per-seconds, the Intersection Over Union(IoU), the sensitivity and the precision. The proposed architecture makes use of 3×3 and 1×1 convolutions, it progressively reduces the feature maps size by a factor of 2 and uses less filters at early layers. DroNet demonstrates that by designing a network from the beginning based on the application requirements, rather than adapting an existing network, can be more beneficial. To this end, we adapt the DroNet architecture and with the addition of shortcut connections, we take feature maps from earlier in the network and merge them with up-sampled features using concatenation, to further improve the detection accuracy for smaller objects. The proposed architecture is called DroNetV3 which is based on improvements also presented in YOLOV3 [30]. Furthermore, we analyze the impact on the performance for both accuracy and processing time, in combination with quantization and tiling techniques applied on both DroNet and DroNetV3, as described in Section IV.

B. Selective Tile Processing

Single-shot detectors based on small networks, suffer from reduced accuracy due to the resizing step of the input image. Resizing has a negative impact on the accuracy due to the reduction of an objects' resolution. In addition, it can potentially distort objects of interest causing a miss-prediction by the detector. In our previous work [16] we proposed an efficient approach for processing an image, without resizing it - the proposed technique works by dividing the input image into smaller regions (tiles) based on the CNN input size and the size of the image. The tiles are uniformly distributed across the input image with a constant overlap between them. This technique of course leads to a notable increase of the number of images that need to be fed into the CNN for processing. To solve the aforementioned problem, we proposed two mechanisms that utilize statistical information gathered over time in order to select only a few tiles for processing, while keeping track of the activity in non-processed tiles. The first mechanism, called *Memory Mechanism*, stores and load the position of previously detected targets in an efficient way using prior information gathered by previous processed frames. With the use of Intersection Over Union (IoU) metric, each bounding box is compared with each bounding box stored in the memory and is categorized as new or already detected.

[1] https://github.com/gplast/DroNet.

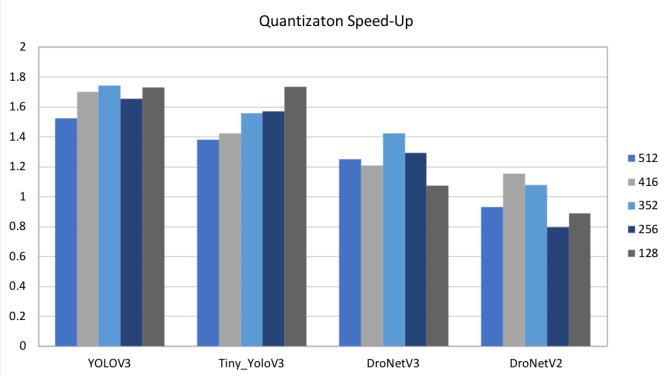

Fig. 1: Quantization Speed-up on different Single-Shot detectors running on NVIDIA Jetson TX2

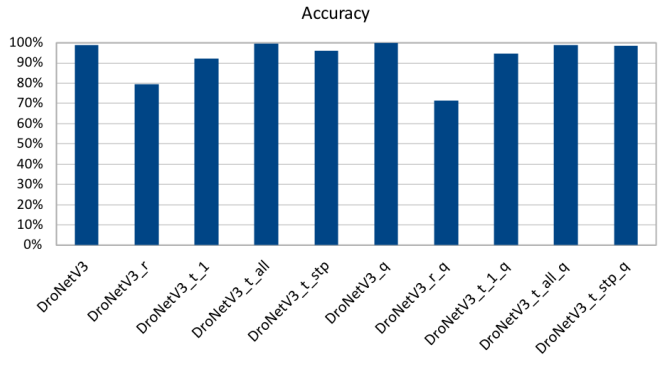

Fig. 2: Accuracy of different DroNetV3 configurations

Moreover, with the use of a memory buffer, an estimate of the position of the objects is extracted in a frame without having to again process the specific tile that contains the object. The second mechanism is an *Attention Mechanism*, which selects the tiles that need to be processed by the CNN on the next frame, by combining both memory and attention mechanism's gathered information from previous frames, such as the number of objects detected in each tile, and only few tiles are selected for processing. Moreover, we give priority to promising regions (i.e. with higher number of objects) or to regions that has not been recently selected for processing. To this end, by intelligently selecting regions of the image it is feasible to use small, fast and efficient networks for object detection, leading to real-time execution and high accuracy on embedded devices.

C. Quantization

A CNN-based object detector requires a significantly large number of multiplication and addition operations, that needs to be executed in real-time. One way to reduce the computational demands while at the same time increase the power efficiency of CNN-based detectors is with the use of quantization. Quantization is applied on both weights and the input of the CNN to reduce the number of bits needed to represent information and make use of integer or reduced-precision operations. With the use of quantization, multiply-add operations are transformed into lower-precision operations which lead to large computational gains and higher performance. In this work therefore, we are using 8-bit integer quantization on the input and weights of the CNN. Our implementation is based on Darknet, a C- and Cuda- based Neural Network framework [31], with the use of DP4A instruction [32], which performs an 8-bit integer 4-element vector dot product with accumulation. To this end, input and weight values are transformed from 32-bit floating point values, into $4 \times 8 - bit$ integer values.

IV. Evaluation of Quantization and Tiling

A. Experimental Setup

We evaluate the impact of quantization on the performance of four single-shot detectors. First, we present and compare the speed-up achieved using quantization for YOLOV3 [30], its smaller and faster version Tiny-Yolov3, DroNet and the enhanced DroNetV3. All networks are evaluated on Jetson TX2[2,3] ARM-based CPUs, since DP4A is not supported by Jetson's GPU. Moreover, we further analyze a combination of quantization and different selection techniques of tiling for DroNetV3 with respect to *Accuracy* metric which shows the percentage of the objects that were correctly detected and classified compared to the ground-truth, the *Intersection-Over-Union* (IoU) which shows the similarity between a predicted object and its ground-truth and finally the *Average Processing Time* (APT) which captures the average time spend at each image for predicting the objects. All networks were trained on the Titan Xp GPU for 200000 iterations, with an initial learning rate of 0.001, momentum=0.9 and decay=0.0005, on a constructed UAV-based pedestrian dataset consist of 1500 images and a total of 60000 pedestrians.

B. Speed-up of Quantization

Fig. 1 shows the speed-up on the performance of all the networks, tested with different input sizes for 100 iterations. Fig. 1 shows that for large networks such as YOLOV3 and Tiny-YoloV3 the speed-up is around $1.4 - 1.7\times$ compared to the smaller networks DroNetV3 and DroNetV2 which is $1 - 1.4\times$. This shows that the impact of quantization on the performance depends mostly from the network architecture (i.e. number of filters, size of filters, input size). It shows also, that in some cases, and in particular for DroNet, the quantization of the input can cause a significant performance overhead, which is a result of the extra processing time required for the quantization of the input image, leading to an increase of the processing time. To this end, in our experiments for DroNetV3 we use the input size with the highest speed-up which is 352×352.

C. Experimental Results: Quantization and Tiling

We further evaluate DroNetV3 using both tiling and quantization techniques, to investigate whether the objective of real-time inference on resource-constrained devices is met. Table I shows the collected results from different configurations of the optimized architecture using the following notation: $\boldsymbol{r}$ - resizing, $\boldsymbol{t}$ - tiling, $\mathbf{1}$ - selecting only one tile for processing,

[2]Dual-Core NVIDIA Denver 2 64-Bit CPU
[3]Quad-Core ARM® Cortex®-A57 MPCore

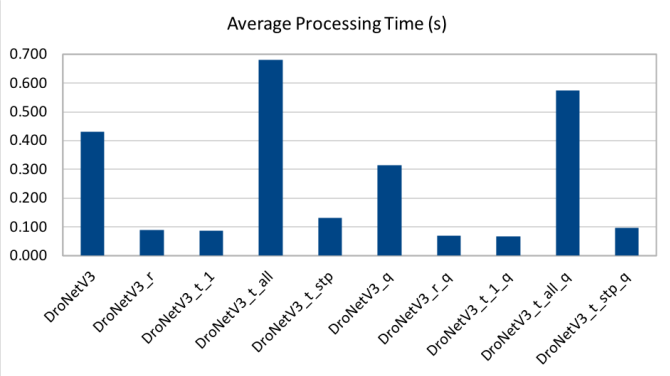

Fig. 3: Average Processing Time of different DroNetV3 configurations

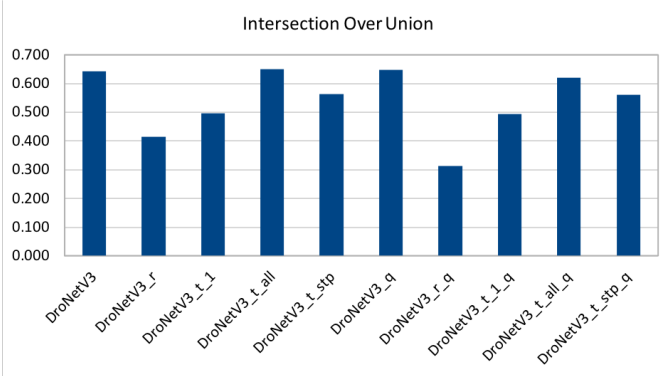

Fig. 4: IoU of different DroNetV3 configurations

all - selecting all tiles for processing, ***STP*** - using Selective Tile Processing [16], ***q*** - quantization. The input size for all networks is 352×352 except in the case where the actual image is fed in the CNN (DroNetV3, DroNetV3_q) to avoid resizing it.

Fig. 2 shows the accuracy metric for different DroNetV3 configurations. Clearly resizing the input image can lead to a significant drop on the accuracy but at the same time lead to a decrease on the average processing time. Comparing DroNetV3 and DroNetV3_r, we observe 20% drop on the accuracy and after applying quantization another 8% drop. This is expected since the resolution of the object is changing significantly along with the reduction in precision from the quantization techniques. On the other hand, resizing the image can speed-up the inference time $9\times$, from $0.432s$ to $0.048s$ as shown in Fig. 3.

The tiling approach, is a way to increase the resolution of the objects, thus as shown in Fig. 2 all of the techniques that are based on tiling, demonstrate increased accuracy compared with the DroNetV3_r. Even in the case where quantization is used, accuracy increased from 71% to $94\% - 98\%$ which is an increase of 23% to 27%. DroNetV3_t_1 has the same processing time as DroNetV3_r since the same amount of data are processed by the CNN. We observe a small increase on the processing time when using the DroNetV3_t_stp since on average the Attention Mechanism is selecting $2-3$ tiles per-frame. Moreover, since quantization has small impact on the accuracy when using the tiling approach, we also observe that DroNetV3_t_stp_q can benefit from quantization with respect to the processing time. Comparing DroNetV3_t_stp with DroNetV3_t_stp_q there is a reduction of the processing time from $0.132s$ to $0.098s$ that is closer to the processing time of DroNetV3_r. This shows that by quantizing and using the STP technique, we are able to increase the accuracy without affecting the processing time.

Moreover, the same applies for the IoU metric as shown in Fig. 4. Avoid resizing the input image, either by feeding the original image or using tiling leads to higher IoU, indicating better localization of the detected objects. Quantization also affects the IoU with a drop from 0.415 to 0.312 for DroNetV3_r and DroNetV3_r_q respectively. On the contrary, combining quantization and tiling has no impact on the IoU.

Model	Accuracy	APT	IoU
DroNetV3	98.929	0.432	0.643
DroNetV3_r	79.643	0.089	0.415
DroNetV3_t_1	92.143	0.087	0.496
DroNetV3_t_all	99.464	0.680	0.650
DroNetV3_t_stp	96.071	0.132	0.563
DroNetV3_q	100.000	0.315	0.648
DroNetV3_r_q	71.429	0.071	0.312
DroNetV3_t_1_q	94.643	0.068	0.494
DroNetV3_t_all_q	98.750	0.574	0.619
DroNetV3_t_stp_q	98.571	0.098	0.562

TABLE I: Accuracy, Average Processing Time(seconds) and IoU of different configurations of DroNetV3

V. Conclusion

In this work we investigate the use of CNN architectures targeting embedded devices, in combination with quantization and tiling techniques for real-time object detection. We determine that with the use of quantization, we are able to decrease the inference time, while maintaining the accuracy of the original full-precision detector, by installing a tiling approach. The combination of the two approaches also manages to provide the same processing time as a slightly faster, quantized network with image resizing, but with an increase of 20% to 90% in terms of its accuracy.

Acknowledgment

This work has been supported by the European Unions Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE) and from the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development. Christos Kyrkou would like to acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] D. Mahajan, R. B. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," *CoRR*, vol. abs/1805.00932, 2018.

[2] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[3] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018.

[4] C. Yuan, K. A. Ghamry, Z. Liu, and Y. Zhang, "Unmanned aerial vehicle based forest fire monitoring and detection using image processing technique," in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 1870–1875, Aug 2016.

[5] H. U. Dike, Q. Wu, Y. Zhou, and G. Liang, "Unmanned aerial vehicle (uav) based running person detection from a real-time moving camera*," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2273–2278, Dec 2018.

[6] H. Niu, N. Gonzalez-Prelcic, and R. W. Heath, "A uav-based traffic monitoring system - invited paper," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, June 2018.

[7] T. Kong, F. Sun, H. Liu, Y. Jiang, and J. Shi, "Foveabox: Beyond anchor-based object detector," *CoRR*, vol. abs/1904.03797, 2019.

[8] B. Singh, M. Najibi, and L. S. Davis, "SNIPER: efficient multi-scale training," *CoRR*, vol. abs/1805.09300, 2018.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[10] J. L. Y. H. H. L. X. L. Ying Wang, Zhenyu Quan, "A retrospective evaluation of energy-efficient object detection solutions for embedded devices," in *IEEE/ACM Proceedings of Design, Automation and Test in Europe conference*, DATE'16, 2018.

[11] Y. Guo, "A survey on methods and theories of quantized neural networks," *CoRR*, vol. abs/1808.04752, 2018.

[12] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Towards compact convnets via structure-sparsity regularized filter pruning," *CoRR*, vol. abs/1901.07827, 2019.

[13] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *CoRR*, vol. abs/1506.02626, 2015.

[14] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018.

[15] M. Elgendy, *Deep Learning for Vision Systems*. Manning, 2019.

[16] G. Plastiras, C. Kyrkou, and T. 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, (New York, NY, USA), pp. 3:1–3:6, ACM, 2018.

[17] G. Plastiras, C. Kyrkou, and T. Theocharides, "Edgenet: Balancing accuracy and performance for edge-based convolutional neural network object detectors," in *Proceedings of the 13th International Conference on Distributed Smart Cameras*, ICDSC 2019, (New York, NY, USA), pp. 8:1–8:6, ACM, 2019.

[18] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[19] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[20] 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," *CoRR*, vol. abs/1611.10012, 2016.

[21] 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 *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 967–972, March 2018.

[22] A. Kouris, C. Kyrkou, and C.-S. Bouganis, "Informed region selection for efficient uav-based object detectors : Altitude-aware vehicle detection with cycar dataset," 2019.

[23] M. Radovic, O. Adarkwa, and Q. Wang, "Object recognition in aerial images using convolutional neural networks," *Journal of Imaging*, vol. 3, no. 2, 2017.

[24] J. Lee, J. Wang, D. Crandall, S. Šabanović, and G. Fox, "Real-time, cloud-based object detection for unmanned aerial vehicles," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, pp. 36–43, April 2017.

[25] D. Gonzalez, M. A. Patricio, A. Berlanga, and J. M. Molina, "A convolutional neural network model for superresolution enhancement of uav images," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 992–997, March 2019.

[26] H. Wu, Z. Zhou, M. Feng, Y. Yan, H. Xu, and L. Qian, "Real-time single object detection on the uav," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1013–1022, June 2019.

[27] G. Maria, E. Baccaglini, D. Brevi, M. Gavelli, and R. Scopigno, "A drone-based image processing system for car detection in a smart transport infrastructure," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–5, April 2016.

[28] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *CoRR*, vol. abs/1511.00363, 2015.

[29] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, July 2017.

[30] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[31] AlexeyAB, "yolo2_light." https://github.com/AlexeyAB/yolo2_light, 2016–2019.

[32] M. Harris, "Mixed-Precision Programming with CUDA 8." https://devblogs.nvidia.com/mixed-precision-programming-cuda-8/, 2016.