Accelerating deep neural networks for efficient scene understanding in automotive cyber-physical systems

Stavros Nousias[1], Erion Vasilis Pikoulis[1,2], Christos Mavrokefalidis[1,2], Aris S. Lalos[1]
[1]Industrial Systems Institute, Athena Research Center, Patras Science Park, Greece
[2]Computer Engineering and Informatics Dept., University of Patras, Greece
Emails: nousias@isi.gr, {pikoulis,maurokef}@ceid.upatras.gr, lalos@isi.gr

***Abstract*—Automotive Cyber-Physical Systems (ACPS) have attracted a significant amount of interest in the past few decades, while one of the most critical operations in these systems is the perception of the environment. Deep learning and, especially, the use of Deep Neural Networks (DNNs) provides impressive results in analyzing and understanding complex and dynamic scenes from visual data. The prediction horizons for those perception systems are very short and inference must often be performed in real time, stressing the need of transforming the original large pre-trained networks into new smaller models, by utilizing Model Compression and Acceleration (MCA) techniques. Our goal in this work is to investigate best practices for appropriately applying novel weight sharing techniques, optimizing the available variables and the training procedures towards the significant acceleration of widely adopted DNNs. Extensive evaluation studies carried out using various state-of-the-art DNN models in object detection and tracking experiments, provide details about the type of errors that manifest after the application of weight sharing techniques, resulting in significant acceleration gains with negligible accuracy losses.**

I. Introduction

In recent years, Cyber-Physical Systems (CPSs) play an important role in modern technology [1], by interconnecting computational and physical resources. CPSs are realized by embedded computers and communication networks that govern physical actuators that operate in the physical world, while receiving inputs from sensors, thus creating a smart control loop capable of adaptation, autonomy, and improved efficiency. CPSs have impacted almost all aspects of our daily life connected with, for instance, transportation systems, healthcare devices, household appliances, electrical power grids, oil and natural gas distribution and many more. Specifically, in the field of Intelligent Transportation System (ITS) [2], the use of CPSs can lead to comprehensive systems that combine advanced technologies with conventional transportation infrastructures, improving the performance of transportation systems, enhancing travel security, fuel economy and, ultimately, enhancing the travel experience of road users.

One of the most essential operations executed at the ITS CPSs to enable the aforementioned benefits is the perception and understanding of dynamic and complex environments from multi-modal sensor data. The critical nature of the perception ability [3] in safety functions for autonomous driving is self-evident: a deviation of, for example, 30 cm in the estimated lateral position of the autonomous vehicle can make all the difference between a "correct" and an "incorrect" (and, potentially, life-threatening) maneuver initiation. One of the major challenges to be addressed, regards the highly dynamic behavior of road users (e.g., pedestrians, cyclists, cars), which can change their motion style in an instance, or start/stop moving abruptly. Consequently, prediction horizons for active perception systems are typically short; even so, small performance improvements can produce tangible benefits. For example, accident analyses [4] show that being able to initiate emergency braking 0.16 s (i.e. five frames at 33 Hz) earlier, at a time to collision of 0.66 s, reduces the chance of incurring injury requiring a hospital stay from 50% to 35%, given an initial vehicle speed of 50 km/h. The aforementioned facts clearly indicate the need for fast and effective scene understanding solutions including, among others, image classification, object detection, object tracking and semantic segmentation.

Here, we focus on DNN-based object detection and, in particular, on the application of MCA techniques on high-performance, pre-trained detectors. Employing MCA techniques can be critical for the efficient execution of the relevant deep models on embedded devices that are deployed on autonomous vehicles. In the following, first, the positioning of the paper is provided through the description of the relevant bibliography and its contribution. Then, the MCA techniques and the object detectors that are adopted for this study, are briefly described. Finally, before concluding the paper, a thorough experimental evaluation of the MCA impact on the behaviour of the adopted models is presented.

II. Relevant bibliography and contribution

Object detection has been evolved considerably since the appearance of deep convolutional neural networks [5]. Nowadays, there are two main branches of proposed techniques. In the first one, the object detectors, using two stages, generate region proposals which are subsequently classified in the categories that are determined by the application at hand (e.g., vehicles, cyclists and pedestrians, in the case of autonomous driving). Some important, representative, high performance examples of this first branch are Faster R-CNN [6], Region-based Fully Convolutional Network (R-FCN) [7], Feature Pyramid Network (FPN) [8] and Mask R-CNN [9]. In the

second branch, object detection is cast to a single-stage, regression-like task with the aim to provide directly both the locations and the categories of the detected objects. Notable examples, here, are Single Shot MultiBox Detector (SSD) [10], SqueezeDet [11], YOLOv3 [12] and EfficientDet [13].

Although two-stage detectors demonstrate better performance than the single-stage counterparts, the latter have lower computational and storage requirements which leads, generally, to faster inference time [14]. In autonomous driving, Advanced Driver Assistance Systems (ADAS) rely on embedded systems with limited resources. ADAS is responsible of executing various machine learning tasks, including object detection, meaning that efficient implementations that take into account those limitations are critical [15]. To this end, single-stage detectors have been particularly studied for autonomous driving by either proposing specialized, compact deep models (e.g., [16], SqueezeDet [11], SA-YOLOv3 [17], Mini-YOLOv3 [18]) or applying MCA techniques [19] to existing, pre-trained models (e.g.,[20] [21], [22], Efficient YOLO [23], ICME 2020 Competition [24]).

The MCA-related literature has been increasing in recent years and there are numerous surveys that provide a comprehensive overview of the area ([19], [25], [26], [27]). Without being exhaustive, some earlier works proposed parameter pruning, in which, unimportant parameters (e.g., filters [28], [29]) are removed and, hence, not considered during the inference phase of the DNN deployment. Other works focus on limiting the representation of the involved parameter by reducing their bit-width or increasing common representations via weight sharing (e.g., scalar, vector and product quantization [30], [31]). Finally, several works employ tensor / matrix decompositions on the involved quantities (e.g., filters) into factors by utilizing, for instance, low-rankness [32].

Most of the MCA techniques that have been applied for the problem of object recognition (as the ones mentioned above), belong either to pruning or scalar quantization, which currently are supported by toolboxes like the TensorFlow Model Optimization Toolkit. Here, moving a step further, we focus on more elaborate and high-performing MCA techniques that belong to weight sharing [33], [34] and study their impact on the performance of object detection for autonomous driving. The contributions of the paper are as follows:

- Two weight sharing techniques are employed for the compression / acceleration of two object detection deep models that are based on the well-known ResNet50 and on SqueezeNet DNNs [11].
- An analysis is provided on the error types that manifest after the application of weight sharing techniques.
- The results obtained on the KITTI dataset using the selected DNN models, reveal acceleration gains of up to 70% with negligible accuracy loss.

III. Weight sharing via Product Quantization

Generally speaking, the linear operation carried out by the convolutional layers can be viewed as involving the computation of dot-products between input and kernel vectors lying in an N-dimensional space, with N being the number of input/kernel channels. In the product quantization framework, the N-dimensional vector space is partitioned into S, N'-dimensional subspaces with $N' = N/S$. Accordingly, the convolution operation is partitioned into S separate sub-convolutions, each involving the N'-D sub-vectors of a particular subspace.

The goal of product quantization is to perform vector quantization to the kernel sub-vectors lying in each subspace, by clustering them into a number of clusters much smaller than the number of the original sub-vectors. This way, each sub-vector is represented by the centroid of the cluster it belongs to, and the acceleration occurs because the original dot-products between the input and kernel sub-vectors, are approximated by the ones between the input and the centroids/representatives.

Although conventionally vector quantization is treated as a clustering problem solved via the popular k-means algorithm, a recently proposed approach that treats the problem from a Dictionary Learning perspective, has shown very promising results [34], achieving up to 100 $\%$ (or, $2\times$) acceleration gain over conventional techniques, on state-of-the-art pre-trained models (VGG, ResNet, SqueezeNet) from the ImageNet competition. This superior performance is achieved by imposing a special structure to the learned representatives, which, for the problem at hand, allows more representatives than the k-means-based approaches for the same target acceleration, thus improving considerably the quantization error.

More specifically, the kernel approximation scheme incurred by the conventional (referred as VQ hereafter) approach, can be as follows:

$$\mathbf{W} \approx \mathbf{C}\mathbf{\Gamma}, \tag{1}$$

where the columns of $\mathbf{W}$, $\mathbf{C}$, and $\mathbf{\Gamma}$, contain the kernel sub-vectors (of a particular subspace), the representatives (or cluster centroids), and assignment vectors, respectively. Each column of $\mathbf{\Gamma}$ has exactly one non-zero element, equal to 1, meaning that each sub-vector (column) from $\mathbf{W}$ is approximated by exactly one of the K_{vq} representatives (columns) in $\mathbf{C}$.

On the other hand, the Dictionary-Learning based approximation (referred as DL hereafter) can be stated as:

$$\mathbf{W} \approx \mathbf{D}\mathbf{\Lambda}\mathbf{\Gamma}, \tag{2}$$

where $\mathbf{W}$ and $\mathbf{\Gamma}$ are defined as in (1), while $\mathbf{D}$ and $\mathbf{\Lambda}$ denote the dictionary and the matrix of sparse coefficients, respectively. Specifically, the columns of $\mathbf{D}$ (called dictionary atoms), are normalized, while $\mathbf{\Lambda}$ is a sparse matrix in the sense that each of its columns contains at most ρ non-zero elements, with ρ being the sparsity level. Thus, under the DL-based scheme, the original sub-vectors are approximated via K_{dl} representatives contained in $\mathbf{D}\mathbf{\Lambda}$, which, in turn, are obtained as linear combinations of at most ρ atoms from a dictionary of size L_{dl}, with $L_{dl} < K_{dl}$.

Due to the linearity of the operations performed in the convolutional layer, the sparse coefficients in $\mathbf{\Lambda}$ need only

be applied to the convolution between the input and the dictionary atoms in $\mathbf{D}$, instead of the atoms themselves. This endows the DL approximation scheme defined in (1) with the flexibility to use a number of representatives K_{dl} that is several times larger than K_{vq}, while restricting the size of the dictionary (so that $L_{dl} \ll K_{vq}$), thus, reducing the number of "heavy" convolutions. Due to this fact, the DL approximation results in significantly higher acceleration ratios (namely, the ratio of original vs accelerated computational complexities, measured by required multiply-accumulate operations) for the same quantization error, as shown in [34].

IV. Application on widely adopted DNN models

Two deep detection network architectures, namely SqueezeDet and Resnet50ConvDet, were employed for the evaluation of the presented weight sharing approach. They are fully convolutional detection networks presented by Wu et al. [11], consisting of a feature-extraction part that extracts high dimensional feature maps for the input image, and ConvDet, a convolutional layer to locate objects and predict their class. For the derivation of the final detection, the output is filtered based on a confidence index also extracted by the ConvDet layer. Figure 1 presents the overall architecture of the deep networks, the convolutional volume kernel shapes and the feature tensor shapes.

As it can be observed from Fig. 1(a), the feature-extraction (convolutional) part of SqueezeDet is based on SqueezeNet [35], which is a fully convolutional neural network that employs a special architecture that drastically reduces its size while still remaining within the state-of-the-art performance territory. Its building block is the "fire" module that consists of a "squeeze" 1×1 convolutional layer with the purpose of reducing the number of input channels, followed by 1×1 and 3×3 "expand" convolutional layers that are connected in parallel to the "squeezed" output. SqueezeNet consists of 8 such modules connected in series.

On the other hand, the backbone of ResNetDet is based on the convolutional layers of ResNet50 [36], whose building block consists of three layers, stacked one over the other, as depicted in Fig. 1(b). The three layers are 1×1, 3×3, 1×1 convolutions. The 1×1 convolution layers are responsible for reducing and then restoring the dimensions. The 3×3 layer is left as a bottleneck with smaller input/output dimensions. The convolutional part of ResNetDet consists of 13 such blocks.

V. Experimental evaluation

A. Training

Both networks were trained with the KITTI odometry dataset [37] consisting of 7477 color traffic scenes images of 1242×375 pixels. Three classes are taken into account, namely, cyclists, pedestrians and cars which were manually annotated with bounding boxes containing the objects in the scene. A significant observation regarding the dataset is that not all objects of the same class are labeled in each and every image. Such a fact plays a role in the evaluation of the detection outcome as our analysis will reveal. The dataset was split in a $80\%, 20\%$ for training and validation, respectively, resulting in $N_{tr} = 5980$ training examples and $N_{val} = 1497$ validation examples.

For the training of the SqueezeDet architecture, Stochastic Gradient Descent (SGD) was employed with the following values for the hyperparameters (determined via experimentation); batch size $B = 8$, learning rate $LR = 10^{-4}$, with a weight decay rate $D_W = 10^{-4}$, a learning rate decay rate of $D_{LR} = 2 * LR/N_e$, number of steps $N_s = 3 \times N_{tr}$ and a dropout rate of 50%, over a total of $N_e = 300$ epochs. Training and testing took place in an NVIDIA GeForce GTX 1080 graphics card with 8GB VRAM and compute capability 6.1 in a Intel(R) Core(TM) i7-4790 CPU @ 3.60Hz based system with 32GB of RAM.

Likewise, for Resnet50ConvDet, we also employed SGD with hyperparameter values as in the case of SqueezeDet. Training and evaluation of Resnet50ConvDet took place in an NVIDIA GeForce Geforce RTX 2080 with 16GB VRAM and compute capability 7.5 in a Intel(R) Core(TM) i7-4790 CPU @ 3.60Hz based system with 16GB of RAM.

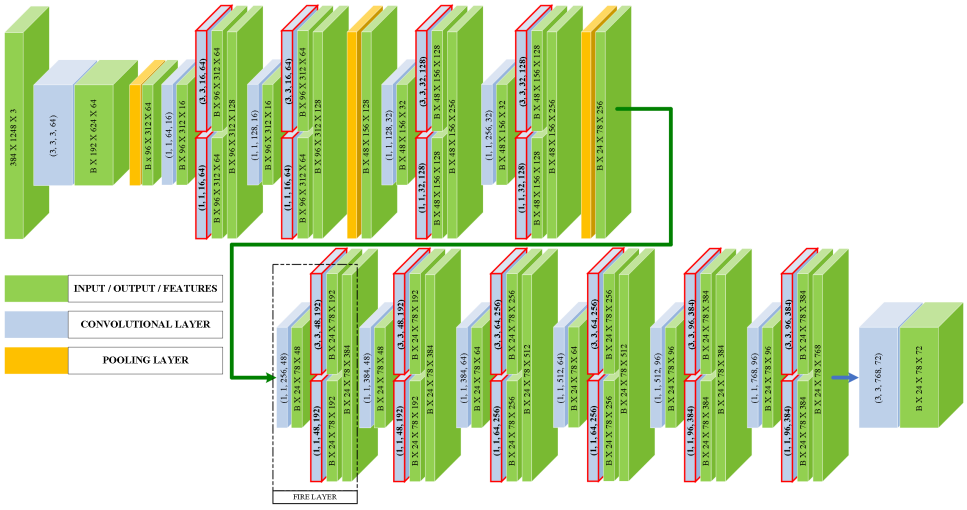
In all cases, training took place with a data augmentation scheme where the bounding boxes drift by $k_x * 150$ pixels across the x-axis and $k_y * 150$ pixels across the y-axis, where $k_x, k_y \sim U(0, 1)$. A 50% probability is also assumed to flip the object.

B. Acceleration scheme

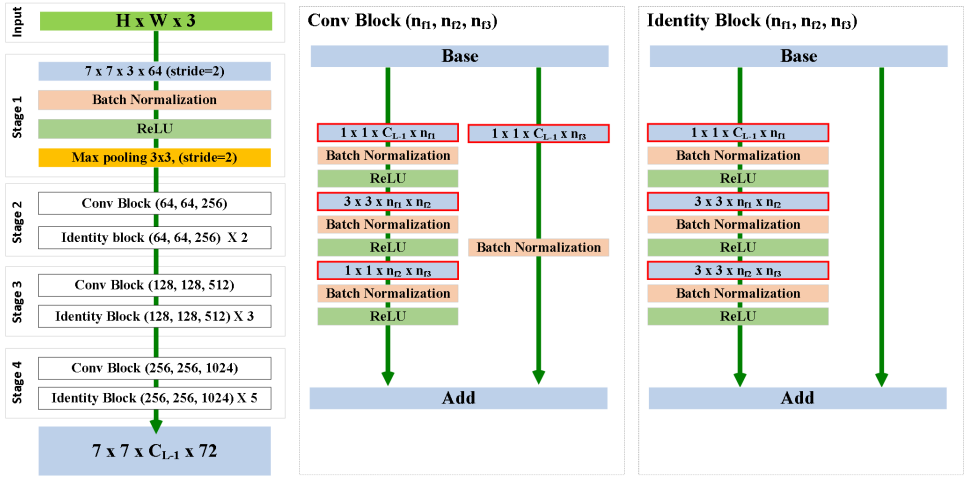
In our experiment, we apply the rival techniques to the two detection models in a "full-model" acceleration scenario. It involves accelerating multiple (or all) convolutional layers of the original models and measuring the achieved performance of the accelerated networks.

It is noted, here, that, although full-range acceleration depends heavily on the performance of the technique used for the acceleration of each layer, it also involves experimentation over the strategy used for accelerating the layers and the involved fine-tuning (re-training) of the accelerated model. Here, we follow a stage-wise acceleration approach (as proposed in [33]) with each stage involving the acceleration (and fixing) of one or more layers of the network, and, subsequently, fine-tuning (i.e., re-training) the remaining original layers. The starting point for each stage is the accelerated and fine-tuned version of the previous stage. The process begins with the original network and it is repeated until all target layers are accelerated. For fine-tuning and performance assessment, we use the training and validation datasets from KITTI, as previously explained.

a) Accelerating SqueezeDet: The feature-extraction part of SqueezeDet, namely SqueezeNet, is responsible for roughly 83% of the total 5.3×10^9 multiply-accumulate (MAC) operations required. Since SqueezeNet constitutes an already "streamlined" network, in our acceleration experiments we followed a moderate acceleration strategy only targeting the "expand", as shown in Fig. 1(a). Acceleration was performed in a one-module-per-stage fashion for a total of 8 acceleration stages. Using acceleration ratios of $\alpha = 8$, 10, 12, and 20



(a)



(b)

Fig. 1: Architectures of the employed detector networks (a) SqueeezeDet, and (b) ResNetDet. The convolutional layers highlighted by the red frames constitute the target layers in our acceleration experiments. B is the batch size, H the height and W the width of a volume kernel. C_{L-1} is the number of channels of the previous layer.

on the targeted layers, an acceleration of the SqueezeNet part by 72%, 74%, 75%, and 78%, respectively, and a total model acceleration by 59%, 60%, 62%, and 65%, respectively, were achieved.

b) Accelerating ResNetDet: The feature extraction part of ResNetDet is responsible for roughly 81% of the total 3.5×10^{10} MACs required by the network. Following the network's architecture, in our experiments with ResNetDet, we accelerated its convolutional (feature-extraction) blocks in a one-block-per-stage fashion leading to 13 total acceleration stages. Using acceleration ratios of $\alpha = 8$, 10, 12, and 20 on the targeted layers (see Fig. 1(b)), an acceleration of the feature-extraction part by 84%, 86%, 88%, and 92%, respectively, and a total model acceleration by 67%, 69%, 71%, and 74%, respectively, were achieved.

C. Metrics

For each detection, the Intersection Over Union (IOU) score is computed as the ratio of area of intersection to the area of union between the predicted and ground-truth bounding boxes. A true positive occurs when IOU>0.5 and the predicted class is the same as the ground-truth class. A false positive occurs when IOU<0.5 or a different class is detected, meaning that unmatched bounding boxes are taken as false positives for a given class. Precision, recall and mean average precision (mAP) are subsequently calculated according to [38].

D. Results

The progressive, stage-wise acceleration results for the employed networks, using both the VQ and the DL acceleration techniques for various acceleration ratios, are shown in Fig. 2. The rightmost point in every plot depicts the performance of the "fully" accelerated network, i.e., after all targeted convolutional layers have been accelerated. At each point, the performance of the detectors was assessed based on the achieved mean average precision (mAP) and recall.

As a general comment, the results presented in Fig. 2 reveal a very promising performance by the employed weight-sharing

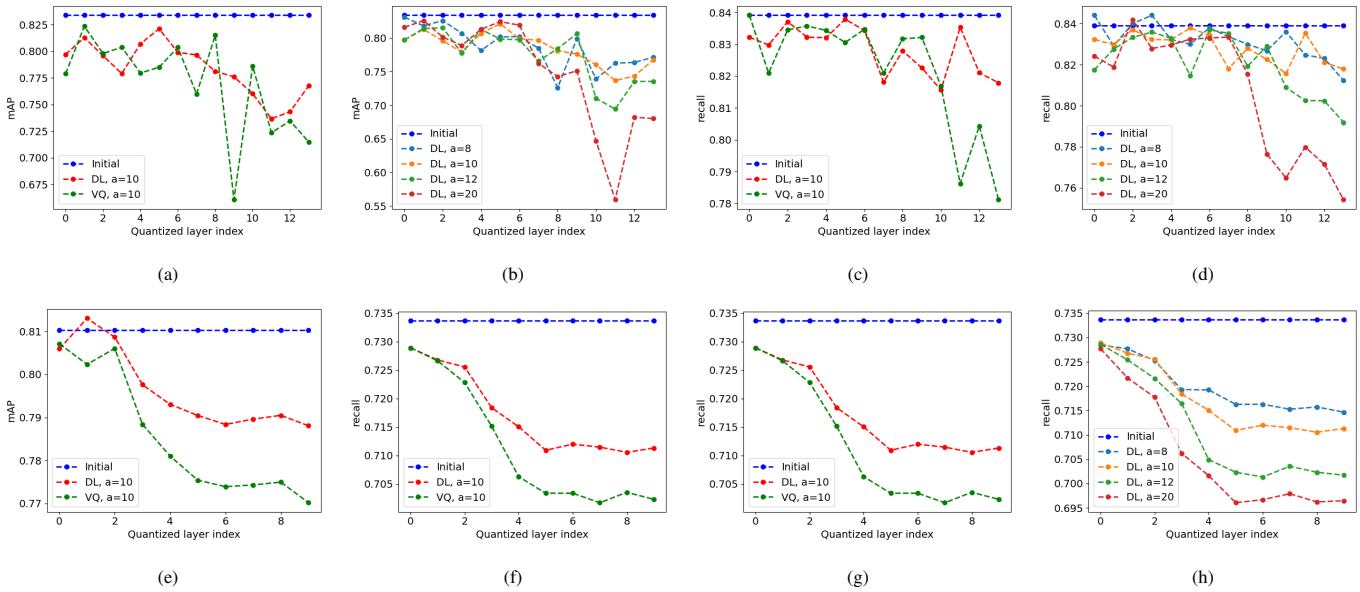


Fig. 2: Performance evaluation and comparison of DL vs VQ acceleration techniques on ResNetDet (top row) and SqueezeDet (bottom row).

techniques, and, especially so, for the DL-based one, whose application results in significantly accelerated detectors, with limited loss of their detection capabilities, as expressed by both the mAP and recall values. It should be also noted that these results could be further improved by following a more targeted acceleration strategy (e.g., via experimentation over the acceleration sequence, the acceleration ratio per layer, using a more extensive fine-tuning process, etc.) which acts as further confirmation of our conclusion. Moreover, comparatively speaking, the DL-based technique managed to generally outperform its rival in our experiments, as highlighted by the plots presented in Fig. 2, for an acceleration ratio of $a = 10$ (Fig. 2(a)&(c), and (e)&(g), for ResNetDet, and SqueezeDet, respectively).

Application instances of the accelerated versus the original networks using examples from the KITI dataset are shown in Fig. 3, respectively.

E. Error-type analysis

For a better insight on the obtained results, we performed an in-depth analysis of the error-types of the employed detectors. For this analysis, we examined 64 images containing the groundtruth annotation and the detection outcome and classified the errors into seven categories; a) object located but not labeled in dataset, b) object located buy bounding box not in place ($IOU < 0.5$), c) object located but overlapping double bounding box appeared, d) non existent object located, e) object not located due to occlusion, f) object not located at all, and g) mirrored object (i.e., on glass surface), object located but in wrong class. Furthermore, we manually classified the 64 images into clear scenes with sufficient light and no occlusions, and messy scenes with many objects some of them being occluded. The motivation behind this perspective is that the detector correctly detects an object but it is assumed as an error or the detector correctly misses an object (i.e., occlusion) but it is assumed as an error since it was originally annotated in the dataset. As we can observe 50% of the errors in the examined images, are objects that were actually found but either they were not annotated or there was a bounding box issue. The results of this qualitative analysis are summarized by the bar-charts shown in Fig. 4.

VI. Conclusions

This work investigates the acceleration benefits of weight sharing methods in deep learning based scene analysis for automotive CPSs. Best practices for optimizing the available variables and the training procedures are described based on extensive evaluations on the KITTI dataset. The presented results provide details about the type of errors that manifest, resulting in significant acceleration gains with negligible accuracy losses. By inspecting the error analysis it can be easily seen that most of the errors are attributed to annotation uncertainties. A more thorough investigation that utilizes also synthetic datasets generated from the CARLA autonomous driving simulator is currently under investigation and it is expected to alleviate the impact of the uncertainties to the training and validation errors, providing additional space for acceleration gains.

Acknowledgement

This paper has received funding from H2020 project CPSoSaware (No 873718) and the DEEP-EVIoT - Deep embedded vision using sparse convolutional neural networks project (MIS 5038640) implemented under the Action for the Strategic Development on the Research and Technological Sector, cofinanced by national funds through the Operational programme of Western Greece 2014-2020 and European Union funds (European Regional Development Fund).

(a) Open street junction

(b) Narrow street

Fig. 3: Application of accelerated vs original SqueezeDet models, using examples from the KITTI dataset. Green rectangles correspond to ground truth boxes, while red rectangles to predictions. The confidence scores are also shown in red letters. Yellow rectangles in (a) and yellow dot in (b) highlight the most obvious performance degradation of the accelerated networks, as compared to the original one.

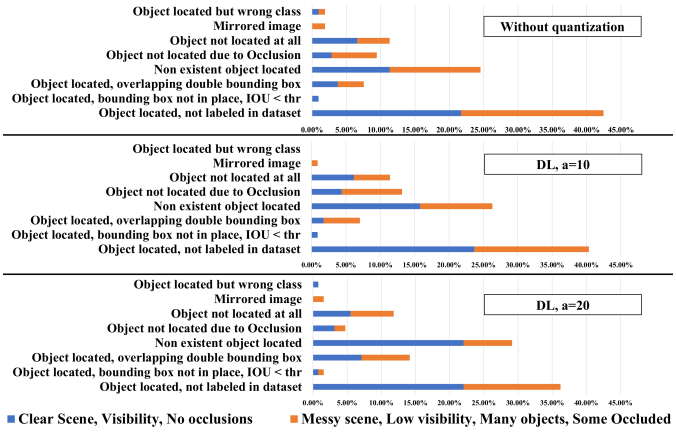


Fig. 4: Error type analysis using manually evaluated examples.

REFERENCES

[1] S. Zanero, "Cyber-physical systems," *Computer*, vol. 50, no. 4, pp. 14–16, 2017.

[2] J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, 2011.

[3] J. Zhang and D. Tao, "Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things," *IEEE Internet of Things Journal*, 2020.

[4] J. Lenard, R. Danton, M. Avery, A. Weekes, D. Zuby, and M. Kühn, "Typical pedestrian accident scenarios for the testing of autonomous emergency braking systems," in *Enhanced Safety of Vehicles (ESV) Conference, Washington, USA*, 2011.

[5] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

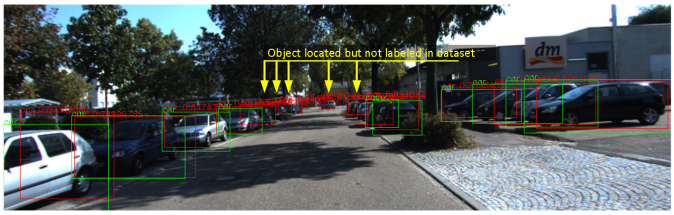
[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[7] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *arXiv preprint arXiv:1605.06409*, 2016.

[8] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

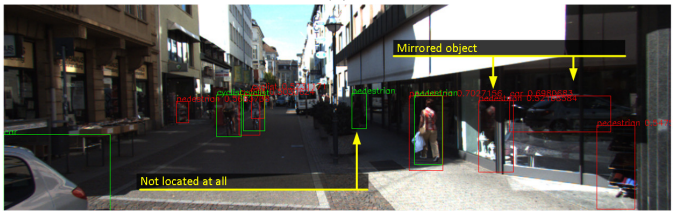
[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C.

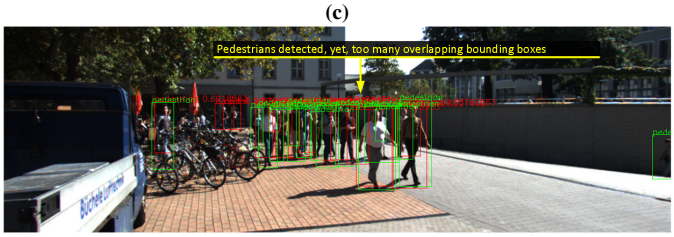


(a)

(b)



(c)



(d)

Fig. 5: Examples of error types. Green boxes refer to groundtruth data, while red boxes to detections.

Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[11] B. Wu, F. 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 129–137.

[12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[13] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[14] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[15] J. Borrego-Carazo, D. Castells-Rufas, E. Biempica, and J. Carrabina, "Resource-constrained machine learning for adas: A systematic review," *IEEE Access*, vol. 8, pp. 40 573–40 598, 2020.

[16] A. Kozlov and D. Osokin, "Development of real-time adas object detector for deployment on cpu," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2019, pp. 740–750.

[17] D. Tian, C. Lin, J. Zhou, X. Duan, Y. Cao, D. Zhao, and D. Cao, "Sa-yolov3: An efficient and accurate object detector using self-attention mechanism for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[18] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, and R.-S. Jia, "Mini-yolov3: real-time object detector for embedded applications," *IEEE Access*, vol. 7, pp. 133 529–133 538, 2019.

[19] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[20] N. Krittayanawach and P. Vateekul, "Robust compression technique for yolov3 on real-time vehicle detection," in *2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)*. IEEE, 2019, pp. 1–6.

[21] J. Xu, Y. Nie, P. Wang, and A. M. López, "Training a binary weight object detector by knowledge transfer for autonomous driving," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2379–2384.

[22] H.-H. Nguyen, D. N.-N. Tran, and J. W. Jeon, "Towards real-time vehicle detection on edge devices with nvidia jetson tx2," in *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 2020, pp. 1–4.

[23] Z. Wang, J. Zhang, Z. Zhao, and F. Su, "Efficient yolo: A lightweight model for embedded deep learning object detection," in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2020, pp. 1–6.

[24] C.-C. Tsai, Y.-H. Yang, H.-W. Lin, B.-X. Wu, E. C. Chang, H. Y. Liu, J.-S. Lai, P. Y. Chen, J.-J. Lin, J. S. Chang, *et al.*, "The 2020 embedded deep learning object detection model compression competition for traffic in asian countries," in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2020, pp. 1–6.

[25] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[26] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Sig. Proc. Mag.*, vol. 35, pp. 126–136, 2018.

[27] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019.

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv:1608.08710*, 2016.

[29] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Toward compact convnets via structure-sparsity regularized filter pruning," *IEEE transactions on neural networks and learning systems*, 2019.

[30] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International Conference on Machine Learning*, 2015, pp. 1737–1746.

[31] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv:1412.6115*, 2014.

[32] S. Bhattacharya and N. D. Lane, "Sparsification and separation of deep learning layers for constrained resource inference on wearables," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, 2016, pp. 176–189.

[33] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized cnn: A unified approach to accelerate and compress convolutional networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4730–4743, 2018.

[34] E.-V. Pikoulis, C. Mavrokefalidis, and A. S. Lalos, "A new clustering-based technique for the acceleration of deep convolutional networks," in *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020.

[35] 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.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[37] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "The kitti vision benchmark suite," *URL http://www. cvlibs. net/datasets/kitti*, vol. 2, 2015.

[38] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.