Accelerating 3D scene analysis for autonomous driving on embedded AI computing platforms

Stavros Nousias\textsuperscript{1}, Erion-Vasili Pikoulis\textsuperscript{1,2}, Christos MAVrokelafidis\textsuperscript{1,2}, Aris S. Lalos\textsuperscript{3}, Konstantinos Moustakas\textsuperscript{3}
\textsuperscript{1}Industrial Systems Institute, Athena Research Center, Patras Science Park, Greece
\textsuperscript{2}Computer Engineering and Informatics Dept., University of Patras, Greece
\textsuperscript{3}Electrical and Computer Engineering Dept., University of Patras, Greece

Emails: nousias@isi.gr, {pikoulis,maurokef}@ceid.upatras.gr, lalos@isi.gr, moustakas@ece.upatras.gr

Abstract—The design of 3D object detection schemes that use point clouds as input in automotive applications has gained a lot of interest recently. Those schemes capitalize on Deep Neural Networks (DNNs) that have demonstrated impressive results in analyzing complex scenes. The proposed schemes are generally designed to improve the achieved performance, leading however to high performing approaches with high computational complexity. To mitigate this high complexity and to facilitate their deployment on edge devices, model compression and acceleration techniques can be utilized. In this paper, we propose compressed versions of two well-known 3D object detectors, namely, PointPillars and PV-RCNN, utilizing dictionary learning-based weight-sharing techniques. It is demonstrated that significant acceleration gains can be achieved with acceptable average precision loss when evaluated on the KITTI 3D object detection benchmark. These findings constitute a concrete step towards the deployment of high-performance networks in edge devices of limited resources, such as NVIDIA’s Jetson TX2.

Index Terms—3D object detection, PointPillars, PV-RCNN, model compression and acceleration, weight sharing, NVIDIA Jetson TX2

I. INTRODUCTION

The continuously growing domain of Autonomous Vehicles (AVs) has gained interest in both academia and industry. AVs are considered as an integral component of connected intelligent transportation systems, improving performance and safety indicators of future mobility systems, providing safer transportation, efficient management of fuel consumption and upgrading the whole travelling experience.

One of the most essential operations executed at the AVs, to enable the aforementioned benefits, is the perception and understanding of dynamic and complex environments from sensor data coming from various modalities (e.g., Camera, LiDAR, etc.). Camera-based scene analysis modules are sensitive to challenging illumination or weather conditions which can significantly degrade the quality of imagery data. On the other hand, LiDAR data are less affected by environmental changes and provide depth information directly, although their distinct sparse representation characteristics, bring new challenges that need to be addressed. Traditionally, a LiDAR analysis module processes the generated point clouds for detecting objects via several operations including background/road subtraction, followed by spatiotemporal clustering and classification [1].

The advances in deep learning are considered as the main driving force towards fast and effective scene understanding solutions. Many recent works have studied the strengths and weaknesses of Deep Neural Networks (DNNs) in detecting objects in point clouds, generated by LiDAR sensors [2]–[4]. While the impressive performance of DNNs in LiDAR-based object detection is nowadays well-established, their high storage and computational costs become problematic especially in real-time applications, like the ones related to scene understanding in AVs.

In this work, we study the application of recently proposed Model Compression and Acceleration (MCA) techniques on state-of-the-art DNNs designed for LiDAR-based 3D object detection in AVs. Our comprehensive evaluation on the KITTI 3D object detection benchmark [5] involving the detection of cars, pedestrians and cyclists, demonstrates very promising results towards the goal of efficient and accurate scene understanding. Apart from a high-performance GPU-based platform, the 3D detectors were also deployed on NVIDIA’s Jetson TX2, thus, revealing a promising future direction concerning the efficient design and execution of compressed models on edge devices, utilizing dictionary learning-based MCA approaches.

The rest of the paper is organized as follows. In Section II, the related bibliography for 3D object detection and MCA techniques is presented, along with the main contributions of the paper. Section III contains a brief description of the DNNs under study and the used weight-sharing MCA techniques. Our experimental evaluation is described in detail in Section IV. Finally Section V concludes the paper.

II. RELATED WORK AND CONTRIBUTION

A. Object detection in LiDAR point clouds

3D object detection from LiDAR point clouds is mainly a data-driven task due to the lack of apparent structure in the data. Deep fully convolutional networks have been traditionally employed in the literature since 2016, with the main evolutionary elements to concern i) the transformation of the 3D point cloud, ii) the network structure and iii) the utilization of feedback loops or abstraction layers for a multiscale feature extraction approach. Initial attempts [6] projected the 3D points in a 2D level and used traditional 2D fully convolutional

This work has received funding from H2020-ICT-2019-2 project CPSSoSaware Cross-layer cognitive optimization tools & methods for the lifecycle support of dependable CPSs (Grant Agreement No. 873718) 978-1-6654-2614-5/21/$31.00 ©2021 IEEE
networks reporting accuracy of 71.0% for moderate difficulty cars. The use of 3D fully convolutional networks [7] increased the accuracy to 75.3% but also increased the computational complexity. Yan et al. [8] proposed sparse convolutional networks improving training and inference times and reaching a reported accuracy of 79.46% for moderate difficulty cars, according to KITTI benchmarks. Pointpillars [9] proposed a novel encoder that utilizes PointNets to learn a representation of point clouds organized in vertical columns (pillars) presenting an accuracy of 74.31% in the same category. 3D object detection with region proposals was introduced in PointRCNN of Shi et al. [10]. They used two stages, where the first stage generates 3D proposals and the second refines them reporting accuracy of 75.64%. An extension of PointRCNN is the Part-A2 Net [11] encompassing a part-aware proposal part and an aggregation part reporting an of 78.49%. PV-RCNN [12] is another novel approach that combines 3D sparse convolutions in a voxelized space with region proposals and an abstraction layer reporting an average precision of 81.43%.

B. Model compression and acceleration

Model Compression and Acceleration (MCA) refers to a family of approaches to produce efficient deep models in terms of the required resources (computational, storage) during the inference phase of its operation. This is achieved by transforming an original, pre-trained deep model of high complexity to a new reduced version without affecting substantially the final performance, depending, of course, on the application. The MCA literature is thoroughly described in several recent survey papers like [13], [14] and [15], where the presented techniques can be utilized on the algorithmic or the hardware level, while, currently, there are also methods that take into account both algorithmic and hardware aspects. Many of the proposed techniques focus on pruning unimportant parts of deep networks like filters [16]. Other approaches limit the representation of the involved parameter by reducing their bit-width or increasing common representations via scalar, vector and product quantization [17], [18]. In another direction, the involved quantities are modelled mathematically as tensors or matrices and decomposed into factors by exploiting inherent properties such as low-rankness [19]. Focusing on the utilization of MCA techniques on 3D object detection using point clouds, the literature is quite limited. An example can be found in [20] where the proposed deep model is transformed via a pruning technique that sets to zero unimportant parameters by exploiting the alternate direction method of multipliers [21].

C. Contribution

The efficacy/impact of more elaborate and high-performing MCA techniques has not been considered yet. In this paper, we focus on weight sharing approaches. In particular, the highlights of the paper are as follows: (i) Two recently proposed weight sharing techniques [22], [23] are utilized on two well-known 3D point-cloud object detection frameworks, namely, PointPillars [9] and PV-RCNN [12]. (ii) The results obtained on the KITTI 3D object detection benchmark [5] reveal considerable acceleration gains, with limited accuracy loss, on both examined models. (iii) Full-model acceleration of up to 9.2× in the case of PointPillars and up to 6.3× regarding the targeted part (namely, its 2D con-layers) of PV-RCNN, has been achieved. (iv) Relative performance drop of PointPillars ranges from being negligible for the class “car” (1 − 5%) to acceptable for “pedestrian” (18 − 21%), and “cyclist” (11 − 16%), across the difficulty levels of the KITTI dataset. (v) For the case of PV-RCNN, mostly negligible losses across the range of classes and difficulty levels, even proving beneficial in specific cases, were observed.

III. WEIGHT SHARING VIA PRODUCT QUANTIZATION ON 3D OBJECT DETECTION SCHEMES

Here, the two object detection schemes that will be considered, namely, PointPillars and PV-RCNN, are briefly presented. The PointPillars network [9] introduces the notion of a Pillar. Based on those Pillars, this network removes the need for 3D convolutions, which have been central to networks like VoxelNet [2] and Second [8], by utilizing strictly 2D convolutions, thus, achieving both high precision and fast inference.

The architecture of PointPillars consists of three stages as depicted in Fig. 1(a). The first stage transforms the point cloud into a pseudo-image. By grouping the points of the cloud into vertical columns, called pillars, that are positioned based on a partition of the $x − y$ plane, this stage summarizes the information of the points per pillar into 1D vectors. These vectors are rearranged appropriately to construct the pseudo-image that will feed the next stage. The second stage consists of a feature extraction backbone network that provides a high-level representation. This representation is subsequently processed by the third stage which is the adopted object detector, producing 3D bounding boxes and confidence scores for the classes of interest. In terms of computational complexity, the backbone network of the second stage, consisting of a number of 2D convolutions and 2D transpose convolutions, requires more than 95% of the involved operations and, thus, we will focus on this stage in the following for its acceleration.

The second object detector for point clouds that will be considered in this paper, is the recently proposed PV-RCNN [12]. On one hand, this system capitalizes on ideas from grid-based object detectors that transform the irregular point clouds into regular representations that can be processed efficiently by ordinary convolutional layers, limiting, however, their performance by the resolution of the adopted grid. On the other hand, PV-RCNN also exploits ideas from point-based object detection, which operates directly on the points of the cloud, removing the need for point cloud discretization with increased, however, computational complexity.

In more detail, the architecture of PV-RCNN is depicted Fig. 1(b). PV-RCNN, first, transforms the point cloud into voxels which are subsequently processed by a voxel backbone network consisting of 3D sparse convolutions. Based on its output, 3D region proposals are produced using the BEV
backbone network whose structure is depicted in Fig. 1. PV-RCNN also samples a subset of the points, named key points, and associates summary features by appropriately concatenating information extracted by the key points themselves and corresponding information extracted from different layers of the voxel backbone network. The information of the 3D region proposals along with the key-point features are appropriately merged over equispaced grid points defined in each 3D region proposal. This enhanced information is eventually processed to procedure the improved final confidence scores for the classes and the 3D bounding boxes. In the following, we will focus on the BEV backbone for studying the impact on the performance of the PV-RCNN.

Viewing the convolution operation as a summation over dot-products between input and kernel (depth- or channel-wise) vectors, Product Quantization (PQ) aims at reducing the required number of dot-products by limiting the number of allowed representations for the kernel vectors (thus sharing results between them), using a Vector Quantization (VQ) framework.

To be more specific, PQ first obtains a suitable partition of the kernel vector space into a predefined number of subspaces and then applies VQ in each of them, that is, it estimates a codebook of representatives that “best” approximate the original sub-vectors, according to some appropriate metric. Following this approximation scheme, only the dot-products between the codewords and the input need to be calculated. The obtained results are subsequently shared (substituted) accordingly among the sub-vectors represented by each codeword. It becomes obvious that the size of the used codebook represents a trade-off between the achieved acceleration/compression and the induced approximation error. Typically, codebook design is addressed by applying the $k$-means algorithm on the original sub-vectors, namely by using the $k$ centroids obtained via $k$-means as the members of the desired codebook. However, treating the problem in a Dictionary Learning framework can lead to significant improvement over the conventional approach, as it was recently shown in works concerning both the acceleration of image classification DNNs (e.g. VGG, ResNet, SqueezeNet) [23], as well as for image-based object detectors (SqueezeDet and ResNetDet) [24].

More specifically, the conventional approximation scheme (referred to as VQ hereafter) can be defined, as follows:

$$W \approx C \Gamma,$$  

where $W$, $C$, denote the matrices holding the original sub-vectors (of a particular subspace), and the codebook (cluster centroids), respectively, while the columns of $\Gamma$ are one-hot vectors indicating the codewords in $C$. On the other hand, the Dictionary Learning based approach (referred to as DL hereafter) imposes a decomposition of the obtained codebook,
as follows:

\[ W \approx DA\Gamma, \quad (2) \]

where \( W \) and \( \Gamma \) are as in (1), \( D \) denotes the dictionary while \( A \) is a sparse matrix, with a hyperparameter controlling the sparsity of its columns. The main advantage of the DL-based codebook design first presented in [23], lies in its ability to significantly increase the size of the employed codebooks, without affecting the achieved acceleration, compared to conventional techniques. This is due to the decomposition of the codebook defined in (2), which decouples the size of the codebook from the number of operations it incurs. To be more specific, the size of the codebook is determined by the size of \( A \), whose sparsity limits the number of required operations, while the main bulk of the operations is due to the dense dictionary \( D \), whose size can be controlled separately. This advantage results in a better quantization error for the same target acceleration, which is ultimately translated into better performance by the accelerated network (compared to the VQ approach).

IV. EXPERIMENTAL EVALUATION

A. Training and evaluation

Both networks were trained with the KITTI 3D object detection [5] benchmark consisting of 7481 training images and 7518 test images as well as the corresponding point clouds, comprising a total of 80.256 labelled objects. In our study, three classes are mainly examined, cyclists, pedestrians and cars annotated with bounding boxes containing the objects in the 3D scene. 3716 annotated Velodyne point cloud scenes were used for training and 3769 annotated Velodyne point cloud scenes were used for testing and validation. For the deployment and retraining of PointPillars and PV-RCNN, the OpenPCDet framework [25] was employed. For the initial evaluation, pre-trained instances were used, while for the retraining, the Adam optimizer was employed with learning rate \( l_r = 0.003 \), weight decay rate \( D_W = 10^{-2} \) and a batch size \( B = 4 \). Training took place in an NVIDIA GeForce RTX 2080 with 16GB VRAM and compute capability 7.5. Furthermore, for the Pointpillars network the detection accuracy was evaluated on NVIDIA Jetson TX2, while for the PV-RCNN network, due to model size, the detection accuracy was evaluated on the NVIDIA GeForce RTX 2080.

B. Acceleration scheme

In our experiments, we apply the VQ and DL weight-sharing techniques to the PointPillars and PV-RCNN models, targeting their convolutional layers, and measuring the performance drop induced by the acceleration, compared to the original networks. The reported acceleration ratios are defined as the ratio of the original to the accelerated computational complexities, measured by the number of multiply-accumulate (MAC) operations. To achieve our acceleration goal, we followed the stage-wise strategy presented in [22], whereby the individual layers are accelerated progressively in stages, starting from the original network. At each stage, the parameters of one or more layers are quantized using the presented techniques and fixed, and subsequently, the remaining layers are re-trained to adapt to the newly presented changes. The process is then repeated for the convolutional layers involved in the next stage, and so on until all desired layers are accelerated. The KITTI 3D object detection dataset is employed for the fine-tuning and performance evaluation, ensuring that the same training examples that were used during the initial training are also used during the fine-tuning step.

a) Accelerating PointPillars: PointPillars is a fully convolutional network with its feature-extraction part (both 2D and transposed convolution operators) being responsible for 97.7% of the total MAC operations required. In total Pointpillars network encompasses 4.835 \( \times 10^6 \) parameters and require 63,835 \( \times 10^9 \) MACs. For a good balance between acceleration and performance drop, we targeted the 2D convolutional layers of PointPillars (consuming approximately 47% of the total MACs), as well as the 4 \( \times \) 4 transposed convolutional layer of the network (responsible for 44.4% of the total MACs), depicted with the red blocks in Fig. 1(a). Acceleration was performed in 16 acceleration stages with each stage involving the quantization of a particular layer, followed by fine-tuning. Using acceleration ratios of \( \alpha = 10, 20, 30, \) and 40 on the targeted layers, lead to a reduction of the total required MACs by 82%, 86%, 88%, and 89%, or equivalently, to total model acceleration of PointPillars by 5.6\( \times \), 7.6\( \times \), 8.6\( \times \), and 9.2\( \times \), respectively.

b) Accelerating PV-RCNN: The main bulk of the operations required by PV-RCNN are consumed by the Voxel-Backbone and the BEV-Backbone blocks shown in Fig. 1(b), with the former one being composed of Submanifold Sparse 3D-Conv layers [26], while the latter consisting of regular 2D convolutional layers. Since the Sparse convolutional layers are already specialized layers that are designed to exploit the sparsity of the input to reduce their computational complexity, and keeping in mind that the number of operations required by such layers is input-dependent, in this experiment we focused only on the BEV-Backbone block of PV-RCNN, as shown in Fig. 1(b). PV-RCNN network encompasses 12,405 \( \times 10^6 \) parameters and requires 88,878 \( \times 10^9 \) MACs without taking into account the sparse convolutional layers. In this case, the targeted layers (highlighted in Fig. 1(b)) are responsible for roughly 86% of the MACs required by the BEV-Backbone block. Similarly to the previous experiment, using acceleration ratios of \( \alpha = 10, 20, 30, \) and 40 on the targeted layers, lead to a reduction of the MACs required by the BEV-Backbone block by 77%, 82%, 83%, and 84%, or equivalently, to the block’s acceleration by 4.5\( \times \), 5.5\( \times \), 6.0\( \times \), and 6.3\( \times \), respectively.

C. Metrics

The official KITTI evaluation detection metrics include bird’s eye view (BEV), 3D, 2D, and average orientation similarity (AOS). The 2D detection is done in the image plane and average orientation similarity assesses the average orientation (measured in BEV) similarity for 2D detections [27]. The KITTI dataset is categorised into easy, moderate,
TABLE I: PointPillars BEV Average Precision. The shown acceleration ratios of $\alpha = 10$, 20, 30, and 40 on the targeted layers, corresponds to a total acceleration of PointPillars by $5.6 \times$, $7.6 \times$, $8.6 \times$, and $9.2 \times$, respectively.

| Car | Pedestrian | Cyclist |
|-----|-----------|---------|
|     | Easy      | Moderate | Hard    | Easy     | Moderate | Hard    | Easy  | Moderate | Hard    |
| Original | 92.04 | 88.06 | 86.66 | 61.60 | 56.01 | 52.05 | 85.26 | 66.24 | 62.22 |
| DL, $\alpha = 10$ | 91.83 | 87.32 | 84.72 | 56.45 | 51.00 | 46.93 | 82.66 | 63.05 | 58.87 |
| DL, $\alpha = 20$ | 92.03 | 84.40 | 84.73 | 56.46 | 50.18 | 46.22 | 83.03 | 64.53 | 60.62 |
| DL, $\alpha = 30$ | 91.76 | 86.80 | 84.08 | 53.49 | 47.14 | 43.71 | 78.82 | 60.61 | 56.98 |
| DL, $\alpha = 40$ | 92.57 | 85.10 | 83.67 | 51.04 | 44.79 | 41.91 | 78.81 | 58.82 | 55.02 |
| VQ, $\alpha = 10$ | 93.08 | 87.24 | 84.54 | 56.74 | 51.02 | 47.28 | 82.78 | 63.83 | 59.89 |
| VQ, $\alpha = 20$ | 91.90 | 86.74 | 84.20 | 54.11 | 47.84 | 43.82 | 84.34 | 64.50 | 60.31 |
| VQ, $\alpha = 30$ | 91.11 | 84.77 | 81.88 | 54.64 | 47.64 | 43.67 | 76.42 | 59.30 | 55.66 |
| VQ, $\alpha = 40$ | 91.75 | 86.41 | 82.44 | 50.81 | 44.94 | 41.53 | 75.26 | 56.31 | 52.83 |

Fig. 2: Qualitative example of detection outcomes for Pointpillars. Green, yellow, and blue colors correspond to classes “Car”, “Cyclist”, and “Pedestrian”, respectively. Red box enclosed areas highlight regions of interest for this example. Accelerated and original networks generate similar errors (false positives) in the same regions of the scene.

TABLE II: PV-RCNN BEV Average Precision. The presented acceleration ratios of $\alpha = 10$, 20, 30, and 40 on the targeted layers, corresponds to a total acceleration of the BEV-Backbone block by $4.5 \times$, $5.5 \times$, $6.0 \times$, and $6.3 \times$, respectively.

| Car | Pedestrian | Cyclist |
|-----|-----------|---------|
|     | Easy      | Moderate | Hard    | Easy     | Moderate | Hard    | Easy  | Moderate | Hard    |
| Original | 94.51 | 90.19 | 88.09 | 71.12 | 64.12 | 59.89 | 91.35 | 74.63 | 69.89 |
| DL, $\alpha = 10$ | 95.67 | 90.46 | 88.40 | 63.16 | 58.64 | 56.24 | 92.81 | 76.99 | 72.39 |
| DL, $\alpha = 20$ | 94.83 | 90.21 | 88.17 | 65.03 | 58.20 | 54.67 | 92.81 | 74.18 | 69.67 |
| DL, $\alpha = 30$ | 95.63 | 90.40 | 88.37 | 66.33 | 58.74 | 54.27 | 92.83 | 75.13 | 71.24 |
| DL, $\alpha = 40$ | 95.11 | 90.27 | 88.12 | 67.95 | 61.11 | 56.96 | 91.67 | 75.93 | 71.15 |
| VQ, $\alpha = 10$ | 95.38 | 90.64 | 88.30 | 66.43 | 59.56 | 55.41 | 92.31 | 75.13 | 70.51 |
| VQ, $\alpha = 20$ | 95.11 | 90.33 | 88.31 | 71.80 | 62.99 | 57.73 | 90.94 | 75.05 | 70.26 |
| VQ, $\alpha = 30$ | 95.12 | 90.09 | 87.96 | 70.79 | 63.13 | 58.00 | 93.86 | 75.66 | 70.97 |
| VQ, $\alpha = 40$ | 95.31 | 90.32 | 88.25 | 67.78 | 59.83 | 55.34 | 92.04 | 74.44 | 69.76 |

and hard difficulties, and the official KITTI leaderboard is ranked by performance on moderate. For the sake of self-completeness, easy difficulty refers to a fully visible object with a minimum bounding height box of 40px and max truncation of 15%, moderate difficulty refers to a partially occluded object with a minimum bounding box height of 25px and max truncation of 30% and hard difficulty refers to a difficult to see an object with a minimum bounding box height of 40px and max truncation of 50%. Each 3D ground truth detection box is assigned to one out of three difficulty classes (easy, moderate, hard), and the used 40-point Interpolated Average Precision metric is separately computed on each difficulty class. It formulated the shape of the Precision/Recall curve as $AP_r = \frac{1}{|R|} \sum_{r \in R} \rho_{\text{interp}}(r)$ averaging the precision values provided by $\rho_{\text{interp}}(r)$, according to [28]. In our setting, we employ forty equally spaced recall levels, $R_{40} = \{1/40, 2/40, 3/40, \ldots, 1\}$.

The interpolation function is defined as $\rho_{\text{interp}}(r) = \max_{r' \geq r} \rho(r')$, where $\rho(r)$ gives the precision at recall $r$, meaning that instead of averaging over the actually observed precision values per point $r$, the maximum precision at recall value greater or equal than $r$ is taken.

D. Object detection

In this section, the impact of the VQ and DL acceleration techniques on the performance of PointPillars and PV-RCNN is presented, following the procedure outlined in Sec. IV. Table I, summarizes the average precision (AP) for various acceleration ratios in the case of PointPillars. For each category (namely, car, cyclist, pedestrian), the three AP correspond to the three levels of difficulty (namely, easy, moderate and hard) that the evaluation dataset provides. It is observed, as expected, that the impact on performance is greater as the acceleration
ratios are increased, a tendency also observed for other values of acceleration ratios that are not depicted here. The relative performance drop is at most 1%, 3% and 5% for the three difficulty levels (easiest to hardest) of “car”, while it is at most 18%, 21% and 21% for “pedestrian” and 11%, 15% and 16% for “cyclist”. It is observed that these results are promising when the two weight-sharing techniques are employed, as the aforementioned maximum performance drops correspond to a considerable reduction on MAC operations, while the impact is negligible for “car”. Note that these acceleration gains can be further enhanced by more tailored MCA configurations involving e.g. layer-specific compression/acceleration ratios. Finally, an example depicting false positive errors from the application of the original and the accelerated networks is shown in Fig. 2. For PV-RCNN, the performance of the model remains practically unaffected by the acceleration of the targeted layers, as shown by the results summarized in Table II. This can be attributed to the limited extent of the affected part of the network, but also to the quality of the employed acceleration techniques. Moreover, it is interesting that applying the procedure described in Sec. IV-B, which involves progressively isolating and fine-tuning specific portions of the network, has even proven beneficial for the overall network’s performance, for certain combinations of categories/acceleration ratios.

V. CONCLUSIONS

This work investigated the impact of two recently proposed weight sharing MCA techniques on the performance of two state-of-the-art 3D object detectors in the frame of automotive applications, namely Pointpillars and PV-RCNN. Specifically, this work investigated was the reduction in MAC operations and storage with respect to the deployment of the aforementioned networks to embedded AI computing platforms and more specifically NVIDIA Jetson TX2. The evaluation of the impact was performed on the KITTI 3D object detection benchmark and demonstrated significant acceleration gains while retaining to a great extent the performance of the original networks. As a next step, we are investigating their deployment on Deep Neural Network ASICS.

REFERENCES

[1] D. Fernandes, A. Silva, R. Névoa, C. Simões, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro, and P. Melo-Pinto, “Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy,” Information Fusion, vol. 68, pp. 161–191, 2021.
[2] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.
[3] B. Yang, W. Luo, and R. Urtasun, “Pixor: Real-time 3d object detection from point clouds,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7652–7660.
[4] K. Shin, Y. P. Kwon, and M. Tomizuka, “Roanet: A robust 3d object detection based on region approximation refinement,” in 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2019, pp. 2510–2515.
[5] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012, pp. 3354–3361.
[6] B. Li, T. Zhang, and T. Xia, “Vehicle detection from 3d lidar using fully convolutional network,” arXiv preprint arXiv:1608.07916, 2016.
[7] B. Li, “3d fully convolutional network for vehicle detection in point cloud,” in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 1513–1518.
[8] Y. Yan, Y. Mao, and B. Li, “Second: Sparisely embedded convolutional detection,” Sensors, vol. 18, no. 10, p. 3337, 2018.
[9] A. H. Lang, S. Vora, H. Caeser, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12697–12705.
[10] S. Shi, X. Wang, and H. Li, “Pointrcnn: 3d object proposal generation and detection from point cloud,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.
[11] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, “From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network,” IEEE transactions on pattern analysis and machine intelligence, 2020.
[12] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, “Pv-rcnn: Point-voxel feature set abstraction for 3d object detection,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10 529–10 538.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” arXiv:1412.6115, 2014.
[19] 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.
[20] S. Zhou, M. Xie, Y. Jin, F. Miao, and C. Ding, “An end-to-end multi-task object detection using embedded gpu in autonomous driving,” in 2021 22nd International Symposium on Quality Electronic Design (ISQED). IEEE, 2021, pp. 122–128.
[21] Z. Zhang, S. Ye, K. Zhou, J. Tang, W. Wen, M. Fardad, and Y. Wang, “A systematic dnn weight pruning framework using alternating direction method of multipliers,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 184–199.
[22] 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.
[23] 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.
[24] S. Nousias, E.-V. Pikoulis, C. Mavrokefalidis, and A. S. Lalos, “Accelerating deep neural networks for efficient scene understanding in automotive cyber-physical systems,” in 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), 2021.
[25] O. D. Team, “Openpcdet: An open-source toolbox for 3d object detection from point clouds,” https://github.com/open-mmlab/OpenPCDet, 2020.
[26] B. Graham and L. van der Maaten, “Submanifold sparse convolutional networks,” arXiv preprint arXiv:1706.01307, 2017.
[27] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “The kitti vision benchmark suite,” URL http://www.cvlibs.net/datasets/kitti, vol. 2, 2015.
[28] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder, “Disentangling monocular 3d object detection,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1991–1999.