An End-to-end Multi-task Object Detection using Embedded GPU in Autonomous Driving

Shanglin Zhou[1], Mimi Xie[2], Yufang Jin[2], Fei Miao[1], Caiwen Ding[1]
[1]University of Connecticut [2]The University of Texas at San Antonio
E-mail: [1]{shanglin.zhou, fei.miao, caiwen.ding}@uconn.edu
[2]{mimi.xie, yufang.jin}@utsa.edu

***Abstract*—Autonomous driving has gained popularity due to its high reliability compared to human drivers. Autonomous vehicles combine variety of sensors to perceive their surroundings, and use deep learning (DL) to extract complicated information from the sensing data. However, there are several challenges: Many DL models have explosive model sizes, and therefore not only time consuming but also power consuming when implementing on embedded systems on vehicles, further degrading the battery life-cycle. The current on-board AI treats lane detection and car location separately. In this paper, we propose an end-to-end multi-task environment detection framework. We fuse the 3D point cloud object detection model and lane detection model, with model compression technique applied. As on-board sensors forward information to the multi-task network, it not only parallel two detection tasks to extract combination information, but also reduces entire running time of the DL model. Experiments show by adding the model compression technique, the running speed of multi-task model improves more than $2\times$. Also, running time of lane detection model on Nvidia Jetson TX2 is almost $6\times$ less comparing with running on CPU, which shows reasonableness of using embedded AI computing device on autonomous vehicle.**

***Index Terms*—3D Object Detection, LIDAR Point Cloud, Lane Detection, Weight Pruning, Embedded Computing Device**

I. Introduction

Autonomous driving has been attracting public attention for over decades as a disruptive technology and opens the door to multi-billion markets [1], [2]. An autonomous vehicle depends on IoT sensors such as camera, LIDAR, ultrasonic, etc. on its external structure to obtain detail environmental "images" [3]. An algorithm is needed to convert all these collected "images" into information that can be read and understood by the on-board IoT system. This system can enable vehicle act upon these traffic, weather, or surrounding information and make informed decisions [4].

Deep learning (DL) has shown great promise in many state-of-the-arts such as image classification [5] and speech recognition [6]. Object detection a prerequisite for autonomous driving because a self-driving vehicle must be able to keep track of its surrounding moving and stationary objects. Flawlessly combining and analysis of collected environment information from on-board sensors and then sending them to the control system for decision making makes deep learning play an integral role in the autonomous driving technique. However, DL models with different focus return different environmental information. Some of the information can be directly used, while some need to be fused with others for further analysis [7]. For instance, object detection models locate neighboring vehicles, lane detection models detect lane markings. One example is obtaining the neighboring vehicles' lane indices. Fusing output of these models will then return lane indices.

To ensure accuracy and performance, deep learning models usually stack many layers with millions of parameters. VGGNet [8], a widely used deep convolutional neural network, which forms backbone for many DL models, has 15 layers and 134×10^6 parameters. YOLONet [9], the first efficient object detection network, has 25 layers and 64×10^6 parameters [10]. The maximum power consumption of Quadro RTX 6000 GPU is 295W [11]; GeForce RTX 2080 Ti GPU is 302W [12]. Running deep learning models on these machines will exceed an autonomous vehicle's battery output power. Thus, equipped a low-power and efficient device is essential. But embedded devices are light in storage and memory. Jetson X1 and X-Carrier have 16GB flash storage and 4GB memory [13], Jetson TX2 has 32GB storage and 8GB memory [14]. Reducing both computation and storage costs of evaluating a model is important [15]. Various techniques exist in this area, e.g. weight pruning, sparsity regularization, quantization, all aim to reduce model size without harming the performance of the model.

In this paper, we design an end-to-end multitask environment detection model that can output neighbouring vehicles' lane indices and their location information. This multitask environment detection model combines two models: 3D point cloud object detection model which only performs on LIDAR point cloud data, and lane detection model which only performs on camera RGB images. Two difficulties for this multitask model: 1) two models have different network structures and scales that cause different running time, so we cannot parallel run the two tasks and get result simultaneously; 2) these two deep learning models are with large model size, so that real-time detection is hard especially on embedded device.

To achieve real-time and parallel, we add the state-of-the-art Alternate Direction Method of Multipliers (ADMM)-based DNN model compression [16] on our multitask environment detection model. ADMM-based model compression can achieve very high compression rate while maintain the prediction accuracy on many DNN architectures. In this case, not only the two models are pruned to appropriate model size that can run on embedded computing device, but also running time of two models decrease to real-time level to meet autonomous driving object detection standard.

We summarize our contributions as:

978-1-7281-7641-3/21/$31.00 ©2021 IEEE

- Combine 3D point cloud object detection task and lane detection task to one multi-task model that can directly return neighbouring vehicles' information.
- Add ADMM-based model compression and realize real-time running of large scale models.
- Our multi-task model combines two networks with different scales and enables them running in parallel.

II. Related Work

A. 3D Object Detection using LIDAR point clouds

The existing 3D point cloud object detection methods mostly analyze and extract the geometric attributes, shape attributes, structural attributes or the combination of multiple attributes of the objects and then do comparison and learning, to complete the object recognition and classification. As it is an intrinsically three-dimensional problem, using 3D convolutional network to solve it becomes a natural thing [17].

PointCNN [18] directly sorts the point cloud data and then do convolution. It learns a permutation matrix to sort and weight the input data. PointNet [19] also directly deals with input data. It extracts order-independent information on feature points using symmetric functions to approximate the global information from point cloud.

However, as the amount of point cloud data acquired by LiDAR is super large, fully connection of the point cloud data will grow explosively, which bringing a great computational burden. The voxel-based approach attempts to convert irregularly distributed point clouds, or meshes, into a rasterized representation of a regular distribution [20], then uses convolution directly on the 3D data. FPNN [21] represents 3D spaces as volumetric fields, employs field probing filters to efficiently extract features from 3D point cloud. VoxelNet [20] introduces an end-to-end network that converts sparse voxel to dense vector and transforms a group of points within each voxel into a unified feature representation.

B. Lane Detection

Lane detection is a fundamental problem because lane markings are the main static component on the road that instruct the vehicles to interactively and safely drive on the way. Deep learning methods that applied to lane detection usually treat the problem as a semantic segmentation task [22]. VPGNet [23] proposes a multi-task network guided by vanishing points for lane and road marking detection. It tries to address not accurate detection under poor weather conditions. SCNN [24] tries to use visual information more efficiently by aggregating information from different dimensions via processing sliced features and adding them together one by one. Other methods focus on real-time running of the algorithm, such as SAD [25] that applies attention distillation mechanism to improve the representation learning of CNN-based lane detection models.

C. Weight Pruning

Many investigations have shown that there exists redundancy in DNN model parameters [27]–[29]. As shown in Fig. 1, effective model compression with negligible accuracy loss can be achieved using weight pruning methods. One fundamental work is [27], that uses a three-step method prunes redundant connections in DNNs. This work reduces 9× number of parameters in AlexNet on ImageNet dataset without accuracy degradation. However, indices are needed to locate which weight to pruning. This is not friendly in hardware implementations for low-performance improvement [30]. This problem is partially addressed by several works [31], [32]. Energy efficiency-aware pruning method [33], [34] facilitates energy-efficient hardware implementations [35], allowing for certain accuracy degradation, and structured sparsity learning technique is proposed for irregular network structure after pruning.

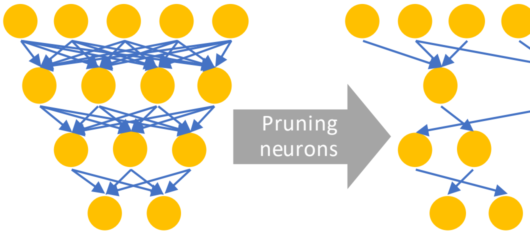

Fig. 1: Illustration of weight pruning for DNNs. Left side is network before pruning, right side is network after pruning, with several neurons and synapses being dropped [26].

III. End to End Environment Detection Network

Fig. 2 shows the end-to-end system structure and our multi-task environment detection network that deployed on the embedded AI device. It consists of two main parts: on-board sensors – including LIDAR and camera; and multi-task environment detection network – our mainly designed network that will fuse information from two detection models and outputs result to decision-making module.

On-board LIDAR captures point cloud and forwards it to the 3D point cloud object detection part of the on-board multi-task environment detection network (shown in the blue rectangle). This part processes the point cloud data and output neighbouring vehicles' information. Meanwhile, on-board camera captures images, forwards them to the lane detection part of the multi-task environment detection network (shown in the light blue rectangle). This part deals with these images, doing analysis and output lane markings information.

Output information from the two parts is then forwarded to the combination part of the detection network (in the orange rectangle). Information is fused, being post-processed and forwarded to decision-making module for further analysis.

A. 3D Object Detection using Point Cloud

The 3D point cloud object detection model deals with LIDAR point clouds as input and detects location of each neighbouring vehicle. To achieve this task, we adopt the method of PointPillars [36]. PointPillars utilizes PointNets [37], which organizes LIDAR point clouds in vertical columns (pillars) form to learn the representation. As shown in left part of Fig. 2, network structure is constructed by three main parts.

First part is Pillar Feature Net, which is a feature encoder network that converts raw point clouds to sparse pseudo-images. This is done by evenly gridding the $x-y$ plane with $P = H \times W$ pillars (H and W are height and width of $x-y$ plane), encoding points in each pillar with $D = 9$

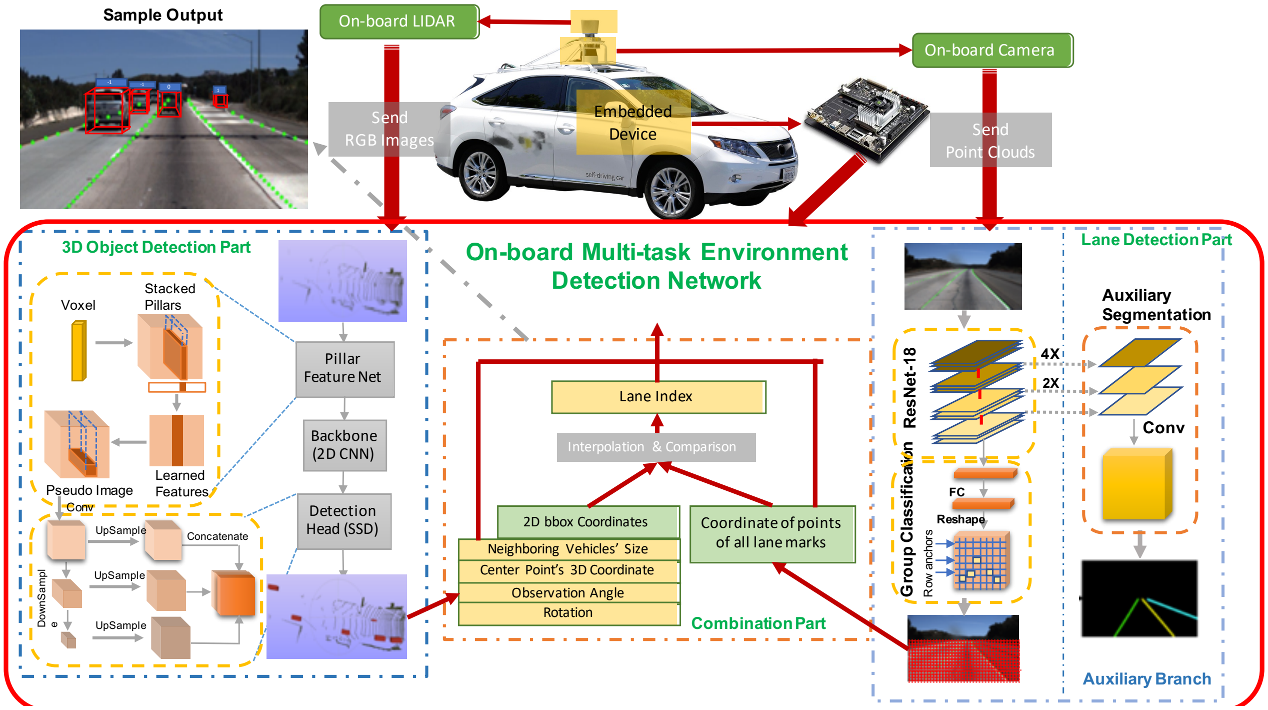

Fig. 2: Autonomous vehicle equipped with on-board sensors and embedded AI devices. On-board LIDAR and camera forward captured images to the embedded device with designed multi-task environment detection network. 3D object detection part (left) deals with point clouds and returns neighbouring vehicles' information. Lane detection part (right) deals with images and returns lane information. Combination part (middle) fuses information and returns post-processing information. Upper left shows a sample output, with neighbouring vehicles' lane indices and bounding boxes, and label of lane markings.

dimension vector $\{x, y, z, r, x_c, y_c, z_c, x_p, y_p\}$, where $\{x, y, z\}$ are original true position coordinates, r is the reflectance, $\{x_c, y_c, z_c\}$ are offsets of this point from the mean point of all points in the pillar, and $\{x_p, y_p\}$ are offsets of point with respect to pillar geometric center. Because the number of points in each pillar is different, points in pillars with more than N points are randomly sampled while points in pillars with less than N points are padding with 0. In this case, a dense tensor with size (D, P, N) is created. Then a simplified version of PointNet is applied to output a (C, P, N)-size tensor, and finally operate *max* over the channels to output the pseudo-image with size $(C, P) = (C, H, W)$.

The second part is a 2D convolution backbone that processes the pseudo-images into high-level representations [20]. Pseudo-images are continuously down-sampled to small spatial resolution features. Convolution, normalization, and non-linear layers are applied to capture these feature information at different scales. Then features are up-sampled using transposed 2D convolution and then being concatenated together.

The third part is detection head for object classes prediction and 3D bounding box calculation. Single shot detector [38] is leveraged. It uses features from the backbone network to predict 3D bounding boxes for objects.

Mean Average Precision (mAP) measures the performance of object detection model [39]. Three concepts are used while calculating mAP. First is intersection over union (IOU). It evaluates the overlap between predicted bounding box B_p and ground truth bounding box B_{gt} with equation $IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$. Second are *true positive* (TP) and *false positive* (FP). With a self-defined threshold, TP refers to correct detection that defines as detection with $IOU \geq threshold$, and FP refers to incorrect detection that defines as detection with $IOU \leq threshold$. Then $Precision = \frac{TP}{TP+FP}$ defines the ability of a model to identify only the relevant objects, and $Recall = \frac{TP}{all\ ground\ truth}$ defines the ability of a model to find all the relevant cases. Precision×Recall curve can evaluate performance of a detector. But the curve is zigzag that hard to do comparison. Hence average precision (AP) is defined as the area under the curve (AUC) of the Precision×Recall curve. Then, mAP is the average of all the classes' AP [40].

B. Lane Detection

The lane detection model deals with camera images and detect detailed coordinate of each lane markings. We applied method from Ultra-Fast-Lane-Detection [41]. Ultra-Fast-Lane-Detection uses ResNet-18 as backbone for global context. It aggregates auxiliary segmentation task that utilizes multi-scale features by extracting middle step feature maps to model local features. This method is not only light and effective that friendly to embedded computing device, but also addresses the no-visual-clue problem in the lane detection area, which means if the lane markings are blurred, affected by light, or even completely obscured, this Ultra-Fast-Lane-Detection method can still accurately detect the lane markings.

Instead of segmenting every pixel of lanes, images are decomposed to a collection of rows, which are called row anchors. Then lane detection is redefined as finding the set that contains positions of lane markings in certain rows of the image, i.e., row-based selection and classification based on global image. As shown in Fig. 3. If images have size $H \times W$, usually h row anchors are selected. Each row anchor is then divided into many grids/cells. So, detect the location of lanes can be converted to localize certain cells over these row anchors. Comparing with traditional segmentation approaches that need to deal with $H \times W$ classification problems, Ultra-Fast-Lane-Detection only needs to deal with classification problems on h rows, except that the classification on each row is W-dimensional. It simplifies the original $H \times W$ classification problems to only h classification problems. Generally, h is much smaller than the image height H.

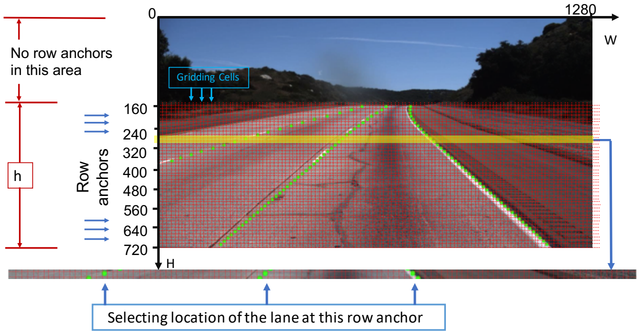

Fig. 3: Illustration of row anchors and decomposition of lanes. Row anchors are predefined as horizontally selecting rows in the images. Each lane marking is decomposed into gridding cells in row anchors.

To handle challenging scenarios such as severe occlusion and extreme lighting conditions, Ultra-Fast-Lane-Detection uses whole image as receptive field to maintain global features. Context information from other locations of the image can be utilized. Auxiliary segmentation learns from prior layers' feature maps, so prior information like shape and direction of lanes can be learned and leveraged in main classification task.

Evaluation metrics for lane detection tasks under different datasets are different. Accuracy is sometimes used, as percentage of the correctly predicted number of lane points [42]. $F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$ is another criteria [24].

C. Multi-task Model

3D point cloud object detection model returns 2D bounding box coordinate, 3D size, 3D location, observation angle and rotation of all neighbouring vehicles, while lane detection model returns detailed pixel coordinates of points on all lane markings. Lane detection model can locate up to four lane markings in one image, up to five lanes can be segmented. We mark these lanes from left to right with index -2, -1, 0, 1, 2. To get lane index of each neighbouring vehicle, we use 2D bounding box information of neighbouring vehicles (from point cloud detection model), set the centre point of the lower bound of the bounding box as the key point and get its x-y coordinate. Because y-axis coordinates of points in each lane marking are determined by predefined row anchors, we compare the key point's x-axis coordinate with all points from lane markings that with same y-axis coordinate with this key point, and use bisection method to calculate index of the lane that this key point fell between.

D. Weight Pruning to Enhance Synchronous Information

Processing time of 3D object detection using PointPillars is significantly longer (11.16×) than lane detection using Ultra-Fast-Lane-Detection. To synchronously obtain the neighbouring vehicles' location and lane markings' information, we adopt ADMM-based weight pruning technique [16], [43] on PointPillars network to reduce its running time.

Alternating Direction Method of Multipliers (ADMM) is an optimization algorithm that breaks an optimization problem into two sub-problems, each of which can be solved iteratively. It can be understood as a smart regularization technique with regularization target dynamically updated in each ADMM iteration, thereby resulting in higher performance in model compression than prior work.

If we consider a general N-layer DNN with loss function $f\left(\{\mathbf{\Theta}_i\}_{i=1}^N\right)$, the overall problem of DNN weight pruning is minimizing this loss function subject to i^{th} layer's weight belongs to $\{\mathbf{\Theta}_i \mid \operatorname{card}(\operatorname{supp}(\mathbf{\Theta}_i)) \leq t_i\}$, where t_i is desired numbers of non-zero weights. We can then use indicator function and incorporate auxiliary variables to re-formulate the loss function to Eq. 1 as

$$\begin{aligned} \underset{\{\mathbf{\Theta}_i\}}{\operatorname{minimize}} \quad & f\left(\{\mathbf{\Theta}_i\}_{i=1}^N\right) + \sum_{i=1}^N g_i\left(\mathbf{P}_i\right) \\ \text{subject to} \quad & \mathbf{\Theta}_i = \mathbf{P}_i, i = 1, \ldots, N \end{aligned} \tag{1}$$

Eq. 1 can then be decomposed into two sub-problems through application of the augmented Lagrangian [43], and can be solved iteratively until convergence.

Euclidean projection is performed during training to guarantee most weights in each layer are non-zero. Then zero-weights are masked and the network is retrained until converge.

IV. Experiments

In this section, we include detail of our experiments, including dataset, experimental settings, data augmentation. We show ADMM result applied on two detection models. Also, we compare power consumption and running speed of models on GPU, CPU, and TX2 under different compression rate.

A. Model and Datasets

For 3D point cloud object detection, we use PointPillars' pretrained model following OpenPCDet repository [1]. Experiments are conducted on KITTI object detection benchmark [44]. The KITTI dataset contains both LIDAR point clouds and camera RGB images samples. Only LIDAR point clouds are used during training. The samples are divided into 7,481 training and 7,518 testing. During experiments, the training part is then be separated to 3,712 training samples and 3,769 validation.

For lane detection, we use pretrained model from Ultra-Fast-Lane-Detection on TuSimple lane detection benchmark [2]. TuSimple dataset is collected with stable lighting conditions

[1] https://github.com/open-mmlab/OpenPCDet

[2] https://github.com/TuSimple/tusimple-benchmark

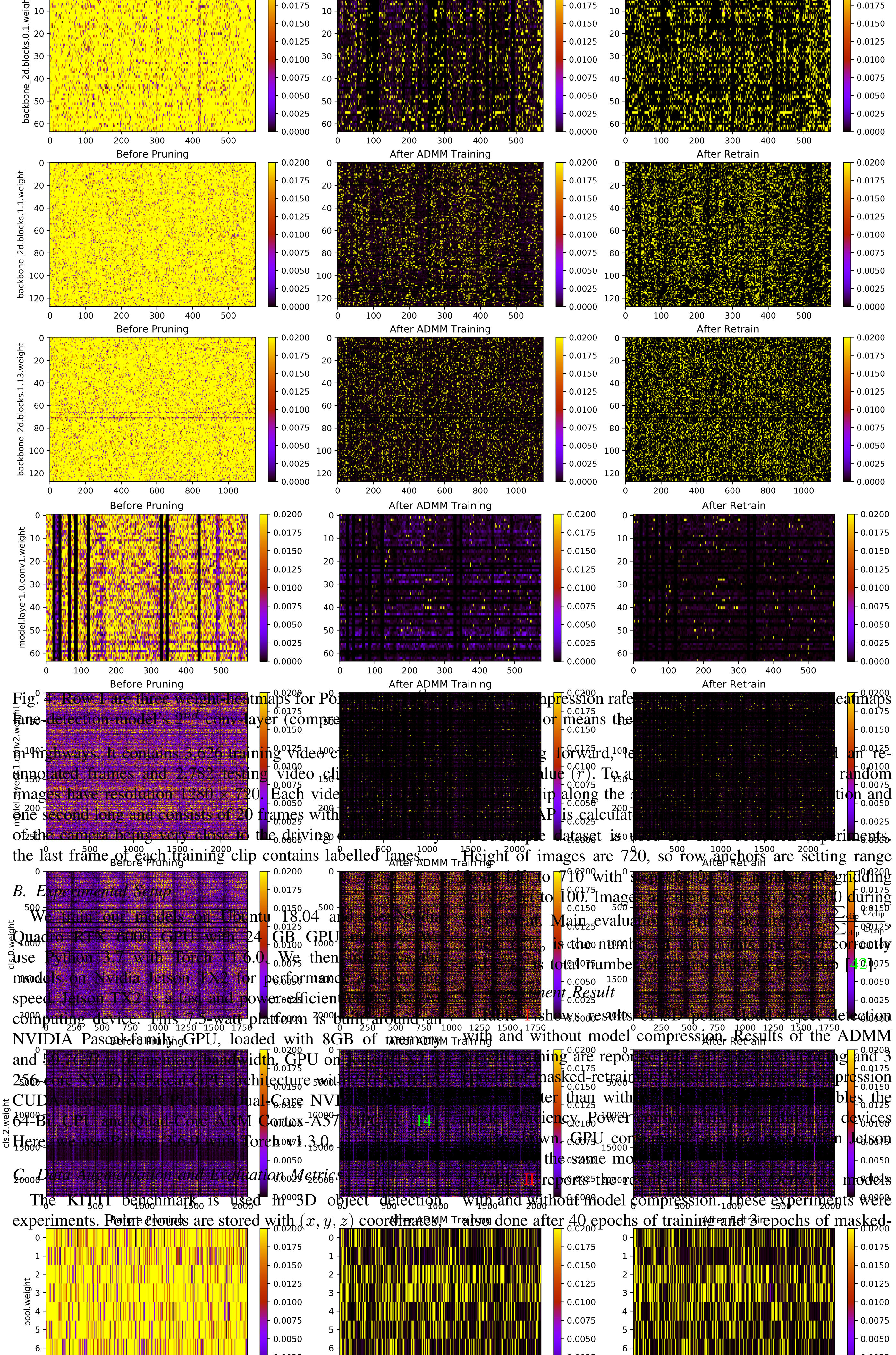

Fig. 4: Row 1 are three weight-heatmaps for Po … th … mpression rate … eatmaps lane-detection-model's 2^{nd} conv-layer (compre … or means the

in highways. It contains 3,626 training video c … annotated frames and 2,782 testing video cli … images have resolution 1280 × 720. Each vide … one second long and consists of 20 frames with … of the camera being very close to the driving … the last frame of each training clip contains labelled lanes.

B. Experimental Setup

We train our models on Ubuntu 18.04 and … Quadro RTX 6000 GPU with 24 GB GPU … use Python 3.7 with Torch v1.6.0. We then … models on Nvidia Jetson TX2 for performance … speed. Jetson TX2 is a fast and power-efficient … computing device. This 7.5-watt platform is built around an NVIDIA Pascal-family GPU, loaded with 8GB of memory and 59.7GB/s of memory bandwidth. GPU on … 256-core NVIDIA Pascal GPU architecture with 256 NVIDIA … CUDA cores, while CPUs are Dual-Core NVI … 64-Bit CPU and Quad-Core ARM Cortex-A57 MPCore [14]. Here we use Python 3.6.9 with Torch v1.3.0.

C. Data Augmentation and Evaluation Metrics

The KITTI benchmark is used in 3D object detection experiments. Point clouds are stored with (x, y, z) coordinates,

… g forward, le … d an re- … lue (r). To a … random … ip along the … tion and … AP is calculat … TuSimple dataset is used for lane detection experiments. Height of images are 720, so row anchors are setting range from 160 to 710 with step of 10. The number of gridding cells is set to 100. Images are then resized to 288 × 800 during experiment. Main evaluation metric is accuracy $\frac{\sum_{\text{clip}} C_{\text{clip}}}{\sum_{\text{clip}} S_{\text{clip}}}$, where C_{clip} is the number of … correctly … is total number … [42].

D. Experiment Result

Table I shows results of 3D point cloud object detection with and without model compression. Results of the ADMM and … pruning are reported … and 3 epochs of masked-retraining. Models … compression … ter than with … bles the … model efficiency. Power consumption … devices … is also shown. GPU con … n Jetson … the same mo …

Table II reports the results for the Lane Detection models with and without model compression. These experiments were also done after 40 epochs of training and 3 epochs of masked-retraining. Models with model compression are 3× faster than without compression, with only 1.4% of accuracy drop. CPU running time on different models is also reported. Under the same model, CPU needs 6× time than Jetson TX2. For power consumption, GPU consumes 33× more power than Jetson TX2 under the same model.

Effectiveness of Weight Pruning: Fig. 4 shows heatmaps of a randomly selected layer in the two models. In both of the models, weights are non-zero before pruning. After ADMM pruning applied, most of the weights become zero. After retrain, some of the weights are back to non-zero values, but the majority are left being zero. This states that by applying ADMM pruning on the two models, most of the weight will be pruned to zero so that running time of the model will reduce. This shows the effectiveness of applying weight pruning.

V. Conclusion

In this paper, we propose an end-to-end multitask environment detection model that can be used in embedded AI computing device for autonomous driving. This multitask model combines 3D point cloud object detection and lane detection. With input being LIDAR point cloud and camera images from onboard IoT sensors, the network can efficiently combine outputs, do analysis and deliver comprehensive information like lane indices of neighbouring vehicles. As 3D object detection model is much slower as the bottleneck of the multitask network, we apply ADMM model compression. Experiment shows running time of 3D object detection model is reduced 2× with slightly mAP drops. We also compare running speed and power consumption on GPU, CPU and Jetson TX2. Experiments show speed on CPU is 6× than on TX2, power consumption of GPU is 33× than TX2. This proves efficient and effective of embedded computing device.

Acknowledgment

This work was supported in part by the NSF S&AS-1849246 grant.

References

[1] McKinsey & Company, "Ten ways autonomous driving could redefine the automotive world," https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world. 1

[2] Shanglin Zhou, Bingbing Li, Caiwu Ding, Lu Lu, and Caiwen Ding, "An efficient deep reinforcement learning framework for uavs," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 323–328. 1

[3] Anil Gupta, "Machine learning algorithms in autonomous driving," https://iiot-world.com/artificial-intelligence-ml/machine-learning/machine-learning-algorithms-in-autonomous-driving/, 2020. 1

[4] Rilind Elezaj, "How do self-driving cars work?," https://www.iotforall.com/how-do-self-driving-cars-work, 2020. 1

[5] Alex Krizhevsky and et al., "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. 1

[6] Geoffrey Hinton and et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012. 1

[7] Ekim Yurtsever and et al., "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, 2020. 1

[8] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1

[9] Joseph Redmon and et al., "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788. 1

[10] Li Liu and et al., "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, 2020. 1

[11] Nvidia, "Quadro rtx 6000," https://www.nvidia.com/en-us/design-visualization/quadro/rtx-6000/, 2020. 1

[12] Chris Angelini, "Aorus geforce rtx 2080 ti xtreme 11g review: In a league of its own," https://www.tomshardware.com/reviews/gigabyte-aorus-geforce-rtx-2080-ti-xtreme-11g,5953-4.html, 2020. 1

[13] Chris Angelini, "Jetson tx1 module," https://developer.nvidia.com/embedded/jetson-tx1, 2020. 1

[14] Nvidia Developer, "Jetson tx2 module," https://developer.nvidia.com/embedded/jetson-tx2, 2020. 1, 5

[15] Wei Niu, Zhenglun Kong, Geng Yuan, Weiwen Jiang, Jiexiong Guan, Caiwen Ding, Pu Zhao, Sijia Liu, Bin Ren, and Yanzhi Wang, "Achieving real-time execution of transformer-based large-scale models on mobile with compiler-aware neural architecture optimization," *arXiv preprint arXiv:2009.06823*, 2020. 1

[16] Tianyun Zhang and et al., "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 4

[17] Martin Engelcke and et al., "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1355–1361. 2

[18] Yangyan Li and et al., "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, 2018. 2

[19] Charles R Qi and et al., "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. 2

[20] Yin Zhou and Oncel 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. 2, 3

[21] Yangyan Li, Soeren Pirk, Hao Su, Charles R Qi, and Leonidas J Guibas, "Fpnn: Field probing neural networks for 3d data," *Advances in Neural Information Processing Systems*, vol. 29, pp. 307–315, 2016. 2

[22] Jihun Kim and Minho Lee, "Robust lane detection based on convolutional neural network and random sample consensus," in *International conference on neural information processing*. Springer, 2014. 2

[23] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In So Kweon, "Vpgnet: Vanishing point guided network for lane and road marking detection and recognition," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1947–1955. 2

[24] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," *arXiv preprint arXiv:1712.06080*, 2017. 2, 4

[25] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1013–1021. 2

[26] Ao Ren and et al., "Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 925–938. 2

[27] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, pp. 1135–1143, 2015. 2

[28] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017. 2

[29] Yijue Wang, Chenghong Wang, Zigeng Wang, and et al., "Mcmia: Model compression against membership inference attack in deep neural networks," *arXiv preprint arXiv:2008.13578*, 2020. 2

[30] Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding, "Efficient transformer-based large scale language representations using hardware-friendly block structured pruning," *arXiv preprint arXiv:2009.08065*, 2020. 2

[31] Tianyun Zhang, Xiaolong Ma, Zheng Zhan, and et al., "A unified dnn weight compression framework using reweighted optimization methods," *arXiv preprint arXiv:2004.05531*, 2020. 2

[32] Yifan Gong, Zheng Zhan, Zhengang Li, and et al., "A privacy-preserving-oriented dnn pruning and mobile acceleration framework," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 119–124. 2

[33] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5687–5695. 2

[34] Deniz Gurevin, Shanglin Zhou, Lynn Pepin, Bingbing Li, Mikhail Bragin, Caiwen Ding, and Fei Miao, "A surrogate lagrangian relaxation-based model compression for deep neural networks," *arXiv preprint arXiv:2012.10079*, 2020. 2

[35] Bingbing Li, Santosh Pandey, Haowen Fang, Yanjun Lyv, Ji Li, Jieyang Chen, Mimi Xie, Lipeng Wan, Hang Liu, and Caiwen Ding, "Ftrans: energy-efficient acceleration of transformers using fpga," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, pp. 175–180. 2

[36] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12697–12705. 2

[37] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927. 2

[38] Wei Liu and et al., "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37. 3

[39] Mark Everingham and et al., "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010. 3

[40] R. Padilla and et al., "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020. 3

[41] Zequn Qin, Huanyu Wang, and Xi Li, "Ultra fast structure-aware deep lane detection," *arXiv preprint arXiv:2004.11757*, 2020. 3

[42] TuSimple, "Tusimple competitions for cvpr2017," https://github.com/TuSimple/tusimple-benchmark, 2020. 4, 5

[43] Stephen Boyd, Neal Parikh, and Eric Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Now Publishers Inc, 2011. 4

[44] Andreas Geiger and et al., "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012. 4