Energy-efficient Edge Approximation for Connected Vehicular Services

Dewant Katare
Department of Engineering Systems and Services
Delft University of Technology
d.katare@tudelft.nl

Aaron Yi Ding
Department of Engineering Systems and Services
Delft University of Technology
aaron.ding@tudelft.nl

***Abstract*—Connected vehicular services depend heavily on communication as they frequently transmit data and AI models/weights within the vehicular ecosystem. Energy efficiency in vehicles is crucial to keep up with the fast-growing demand for vehicular data processing and communication. To tackle this rising challenge, we explore approximation and edge AI techniques for achieving energy efficiency for vehicular services. Focusing on data-intensive vehicular services, we present an experimental case study on the high-definition (HD) map using the model partition approach. Our study compares the AI model energy consumption using multiple approximation ratios over embedded edge devices. Based on experimental insights, we further discuss an envisioned approximate Edge AI pipeline for developing and deploying energy-efficient vehicular services.**

***Index Terms*—3D maps, Approximation, Data Compression, Energy Efficiency, Edge AI, HD map, Model compression**

I. Introduction

Connected autonomous vehicles rely on sensory technologies and computational complex algorithms to process large amounts of data. Usual approaches include supervised, unsupervised and semi-supervised learning [16]. Embedded deployment of collision warning systems, object detection and segmentation have been presented in recent years [10], [11]. However, the implemented AI models are dense in size and consist of millions of parameters, thus requiring heavy processing units for real-time processing, which makes these models high resource and energy-consuming. The requirement of high-performance computational resources and onboard energy consumption will increase when applications and services such as SLAM, path planning and vehicular communication are deployed together [20]. At present, these applications have been proposed to be deployed by processing data at the vehicle onboard computing unit using CNN/DNN models and offloading onboard memory in a planned interval [15].

Analytic studies performed on energy consumption for fully connected autonomous vehicles suggest that overall energy consumption from the vehicular ecosystem can be divided into three categories: 1) Onboard energy consumption (sensors and computing devices). 2) Energy consumption from communication and networking. 3) Energy consumption from the infrastructure sensors, backend units such as Edge-servers and the central data-server maintaining legacy data/models [13], [18]. With the current practices, energy consumption on an autonomous vehicle is around 550-750 Wh/100 km of driving, and the predicted energy consumption, by considering communication between vehicles and computation at the backened units is more than 2500 Wh/100 km of driving. The increase in data amount from the sensors, also the computation and communication required by data-intensive vehicular services are contributing to the increase in high power consumption. To efficiently and economically deploy future connected vehicular services and use cases [15], this rising computational and energy challenge demands an effective design that can enable energy-aware mechanisms for connected vehicular applications. Popular connected vehicular services and future use cases include audio-video streaming, HD map, traffic monitoring and traffic optimization [1]. Some use-case (e.g. traffic monitoring) may require sharing of onboard processed data or model weights from the vehicles, cloud or the data-server (an illustration of connected vehicle ecosystem is shown in Figure 1). Solutions involving edge devices and computing have been proposed [1], [15] by using the connected vehicular ecosystem and by bringing inference to the edge [1]. Distributed machine learning approaches using edge computing and intelligence have also been proposed to tackle the computation on resource-constrained embedded/edge devices [21]. However, the consideration should be also given to manage the high volume data and optimized deployment of AI models through approximate techniques resulting in onboard energy-efficient approaches. The contributions of this paper are as follows:

- We investigate the trade-off of approximation for HD map application and share experimental insights on the accuracy impact for the DNN model.
- We further implement DNN model partition and analyze energy measurements on embedded/edge devices using vehicular dataset and propose an envisioned approximate Edge-AI pipeline for energy-efficient vehicular services.

II. Background

Maps have been regularly used by the automotive industry, to assist the driver navigate from source to destination. In today's vehicle, online mapping services are being used, and are delivered using geographic information system through internet. In autonomous vehicle, this trend will continue in the form of High-definition map, which will be constructed using the automotive sensors such as cameras and lidar [23]. Several techniques of HD map creation, update and change detection have been proposed in the last two years.

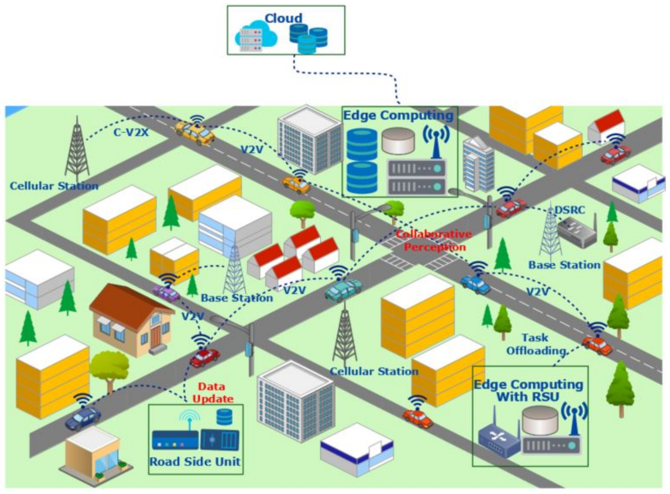

Fig. 1. Future vehicular ecosystem consisting of vehicle, edge and cloud.

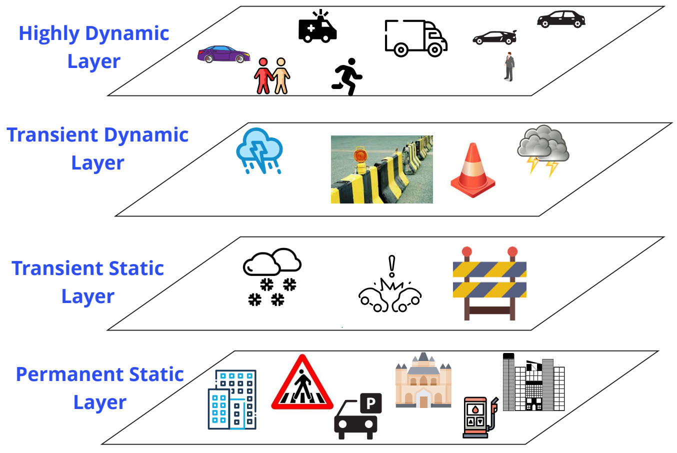

Fig. 2. HD map Layer representation in vehicular ecosystem

Approximate techniques have been implemented on software, hardware and architecture. An approximate operation establishes a trade-off between performance parameters to achieve favourable gains. The following subsection discusses the fundamentals of HD map and approximate techniques.

A. *HD map*

It contains street information in 3d scenarios, overlaid with various information such as traffic signs, speed limits, roadside information, and lane markings, with high precision that allows the vehicle to achieve precise localization. Because of their semantic nature, HD or 3D maps have often been categorized as "Mobility as a Service". Different HD or 3D map versions have been released in the last few years [23]. These maps contain geometric and semantic information from the vehicles surrounding sensed using LiDar, camera, GPS, and IMU, which is further labelled for map creation and development using AI algorithms. Researchers have also proposed aggregating data from several connected cars to construct HD maps in highway driving or special use-cases such as platooning. Future connected and automated vehicles will need precise localization abilities to place themselves into the readily mapped vehicular environment and to localize themselves beyond-line-of-sight [3]. It is not a significant concern if the vehicle is not precisely localized in the current online maps or if the map is not highly accurate or updated as humans still control the vehicle. However, future connected autonomous vehicles should be precisely mapped within the environment using highly accurate and updated maps. It is also important to note that an increasing number of connected autonomous vehicles generates high-volume data, which should be processed onboard and exchanged with the other vehicles and devices within the vehicular ecosystem. Therefore, there is a need to develop data processing pipelines and energy-aware mechanisms that can efficiently handle the large volume of data and vehicular services such as HD maps [3], [15].

Definitions of HD maps may vary depending upon the organization and type of data and information used for the creation and annotation of HD map could be different as well, as the organization has different data collection strategies based on sensor platforms. The automotive edge computing consortium has also described the HD map's operational behaviour for future connected vehicles and associated data and network traffic based on current sensors data-rate [3].

1) High Definition Map Application: At present, there is no particular standard for the data and information that should be stored within the HD maps. The most complete definition based on multiple use-cases is provided by automotive edge computing consortium (AECC). The AECC version of HD map [3], consists of static and dynamic information, that is classified into four layers based on the time intervals (Figure 2). The layer map is inspired from the existing concept of a Local Dynamic Map, which is standardized by European Telecommunications Standards Institute (ETSI). Layers information in the map are as follows:

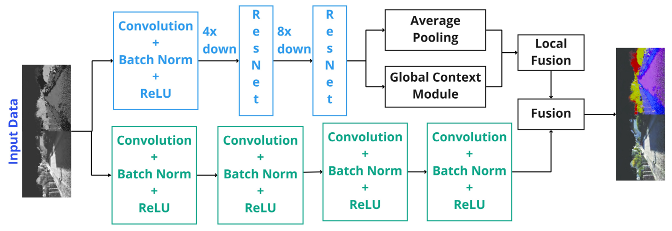
Input Data
Convolution + Batch Norm + ReLU
4x down
ResNet
8x down
ResNet
Average Pooling
Global Context Module
Local Fusion
Fusion
Convolution + Batch Norm + ReLU
Convolution + Batch Norm + ReLU
Convolution + Batch Norm + ReLU
Convolution + Batch Norm + ReLU

Fig. 3. DNN architecture used for Semantic Map

Permanent Static Layer: consists of information that can change within the interval of one day or longer. It can consist of information from traffic lanes, traffic signals, buildings and the 3D scenario of the road. The permanent static layer can be described as a static map. In **Transient Static Layer**, change may occur within a few to several hours. As shown in Figure 2, it consists of information on road construction, accidents and snowfall. The **Transient Dynamic Layer**, see changes in the information very frequently. The change may occur at an interval of a few minutes, which can be because of local weather, such as heavy rain, storm road obstacles and unexpected objects. **Highly Dynamic Layer** contains information within the interval of few minutes to several seconds. The highly dynamic layer can help localize vehicles precisely. It includes the position of moving objects such as other vehicles, bicycles, motors, and pedestrians. Information that requires frequent updates within a second interval in an HD map has been excluded in this section.

B. Approximate Techniques

Two systems or models can be called approximate versions of each other if they can be replaced in error-tolerable applications using approximate operations. An approximation is generally performed to achieve favourable gains in one of the performance parameters by trading the other. Techniques such as model compression and data compression have been implemented to allow fast/real-time inference on resource-constrained devices. Popular compression techniques proposed for vehicular services are pruning, low-rank approximation, quantization, knowledge-distillation, and sketching.

Low-rank Approximation, have been used to compress neural networks used for object detection tasks. Factorization [17], and decomposition [7] of neural networks filters and layers have helped reduce the parameters from the NN. The popular approaches include singular value decomposition [7], [8], Tucker decomposition [12], and canonical polyadic decomposition [2]. The decomposition approach is applied to the parameters for the overall dimension reduction by targeting a channel through decomposing the relevant filter. Similarly, **Pruning** has been used to reduce the overall number of parameters in two forms: removal of weights [14] and removal of neurons [26]. Removal of weights from NN maintains benchmark accuracy as only those weights are removed, which have a magnitude close to zero. Idea of **Quantization** is directly inspired by the human nervous system, where the information is stored in a discrete or detached manner [19]. Currently practised forms of quantization include uniform and non-uniform, static and dynamic, and granular (layer, group, and channel). For vehicular applications, NN quantization can be implemented while training a model or during inference (post-training). In the uniform quantization approach [4], [6], the quantized values are uniformly distributed over the space using a linear approach, while in the non-uniform quantization, the quantized values are non-uniformly distributed using the logarithmic or exponential approach. Non-uniform quantization of DNN based on interval learning is proposed in [9].

C. Model Partition

To tackle the computing and resource challenges, a large DNN model can be partitioned into smaller models and distributed to embedded/edge devices for inference [1]. Model partition approaches are generally inspired by split computing which is based on resource allocation schemes. In these cases an algorithm is used to calculate the computing resources available at the participating devices [21]. Based on the available computing resources, a large DNN model can be split into smaller forms for collaborative training and inference. Each split or partitioned form of the model includes model parameters and weights, which perform the processing individually and transmit the calculated weights to the central device for overall convergence [21].

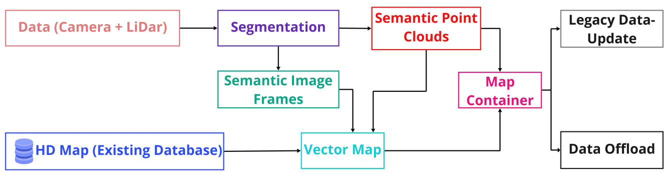

Fig. 4. HD map architecture for vehicular applications

III. Method

This section discusses the model approximation approach for a DNN model. The approximation discussed in this paper specifically covers the convolutional layer in a CNN. The motivation to choose convolutional layers is because of their compute-intensive nature. This discussion is followed by the model partition approach used for the HD Map model.

A. Model Approximation

The semantic model used in this paper is based on BiSeNet architecture [25] first proposed in 2018 for performing real-time segmentation on images. This architecture uses two blocks: Spatial and Context. The Spatial block consists of convolution, batch normalization, and the ReLU layer. The purpose of the Spatial block is to store the high-quality features and spatial information from the input and preserve the output for feature-based fusion with the output of the Context block. In semantic tasks, the receptive field incorporates features into convolutional networks. The Context block is responsible for providing a receptive field in BiSeNet architecture. It consists of average pooling, convolution, ReLU, Sigmoid block, and a global average pooling, thus providing a maximum receptive field for features and lightweight architecture. The computation cost of tasks is tackled using a refinement module, which is responsible for refining the feature of an input image at each stage. This is implemented using average pooling and vector maps to store the learned feature. This can be further easily integrated for feature-based fusion without any up-sampling process, thus requiring less computation. High-level representation is shown in Figure 3.

Since the development in the area of separable convolution, depth-wise convolution, and depth-wise separable convolution, implementation of DNN models on the resource-constrained embedded device has been very popular [14]. These implementations have enabled complex task deployment in an optimized condition. An efficient model/architecture to perform semantic mapping in an autonomous vehicle is BiSeNet [5], which is a comparatively lightweight architecture and is used for real-time applications. The layers of BiSeNet architecture are shown in Figure 3. In this paper, approximate operation/functions have been implemented on the convolution layer of architecture to further approximate/compress the baseline architecture. As defined in [24], the relative error between two systems (original and approximated) can be used as a measure of the difference between them. For a convolution layer, the relative error between original(f_x) and approximated system/model (g_x) can be calculated as:

$$E(g, f) = \sum_{n} \left| \frac{g[x] - f[x]}{f[x]} \right| \tag{1}$$

Since the architecture consists of arbitrary layers of convolutional filters, the above-mentioned relative error should be considered along with the probability distribution functions on the inputs. Therefore, the above equation can be revised as:

$$E(g, f) = \int_{\epsilon} \int_{\eta} \left| \frac{F(\epsilon, \eta) - G(\epsilon, \eta)}{F(\epsilon, \eta)} \right| .\mathcal{P}_x(\epsilon)\mathcal{P}_w(\eta) d(\epsilon) d(\eta) \tag{2}$$

Based on the above equation, $\mathcal{P}_x(x)$ and $\mathcal{P}_w(w)$ are assumed as probability density function of x and w. The principle behind successfully implementing approximation is to minimize the $E(g, f)$. One of the basic operations used in the architecture is convolution, and the convolution of an image can be explained as $Z = X \circledast W$ [11]. Here, X is an image, and W is the kernel. Z can be further calculated as:

$$z_{m,n} = \sum_{i=1}^{k_h} \sum_{j=1}^{k_w} x_{m+i,n+j} . w_{i,j} \tag{3}$$

Here, (m,n) are the pixel coordinates of an image, and k_h, k_w, is the corresponding height and width of the kernel/filter used. As it can be seen from the formula, the procedure includes multiplication. A form of approximation for convolution by [24], has been described as:

$$z_{m,n} \approx \sum_{i=1}^{k_h} \sum_{j=1}^{k_w} \mu_{|w|} . min(x_{m+i,n+j}, \hat{w}_{i,j}) (4)$$

Here, $\mu_{|w|}$ is the expected value of $|w|$ from equation 3. This approximate function has been implemented for the convolution operation. Implementing approximation on the convolution layer is because of its property to occupy maximum computation time during inference. Table 1 corresponds to the approximate ratio of the baseline architecture and the performance metrics. Approximation in the form of ratio has been implemented, starting from 10% up to 45%. Respective model size in megabytes is also shown in the table. The training process consists of forward propagation, which also includes an approximate function for convolution as shown in equation 4, backward propagation, and updating the parameters with the learning rate. In this case, adam update rule and learning rate of 0.001 is used.

TABLE I
PERFORMANCE OF DNN MODEL WITH DIFFERENT APPROXIMATE RATIO. FIRST ROW IS BASELINE ARCHITECTURE, AND FOLLOWING ROWS ARE WITH APPROXIMATE PERCENTAGE OF THE BASELINE (FOR EXAMPLE: APPROX-10 IS 10% (SIZE OF BASELINE.)

Approximate Ratio/Per	Size (MB)	Recall	Accuracy	IoU
Baseline	134.7	85.27	88.05	0.86
Approx-10	120.6	82.71	84.58	0.85
Approx-20	107.1	78.14	77.03	0.79
Approx-25	100.3	71.04	72.22	0.73
Approx-35	86.8	64.51	65.09	0.61
Approx-45	73.2	55.68	53.66	0.57

B. Model Partition

To partition or split the model for vision-based applications, it is important to account for parallelization. The operation in a deep neural network can be described as the calculation of output tensors. As the input and output tensor consists of several dimensions, the partition of the tensor can be used to parallelize this operation. In the case of the DNN model consisting of convolutional operations which basically computes feature/activation maps by calculating the channel, height and width of given input. Partitioning this three outputs can help to execute the operation in a parallel manner and further results into a partitioned model.

Model Training: When the baseline model is trained with a feature-rich dataset such as Argoverse [22], the output model file is approximately 134.7MB, as shown in Table 1. Deployment of such a model on resource-constrained devices brings several computing challenges. Five versions of the baseline architecture were trained on the PyTorch framework in high-performance computer clusters using the approximate operation on the convolution layer. Model performance parameters such as recall, accuracy, and intersection-over-union were computed during training and validation. On the validation set for semantic vector maps, the baseline implementation resulted in an accuracy of 88.05, with a recall score of 85.27 and IoU of 0.86. Further, an approximation of 10% is implemented on the baseline model, which resulted in the model size of 120.6 MB with recall, accuracy, IoU of 82.71, 84.58, and 0.85, respectively. An approximation of 25 and 35 percent from the baseline can be exploited to develop applications using balanced trade-offs (energy and accuracy) for applications that can tolerate degradation in performance. For the experiments, Argoverse dataset is used for training, and testing [22]. It is very large and includes 200 test scenarios. Therefore, a split approach is used to train and test the model.

IV. EXPERIMENTS AND RESULTS

This section discusses model deployment on Nvidia Jetson-Nano and energy measurement with the monsoon high voltage power monitor using data compression algorithms. Jetson-Nano performs in memory/compute-constrained settings, making it ideal for testing model/architecture designed for resource-constrained embedded devices.

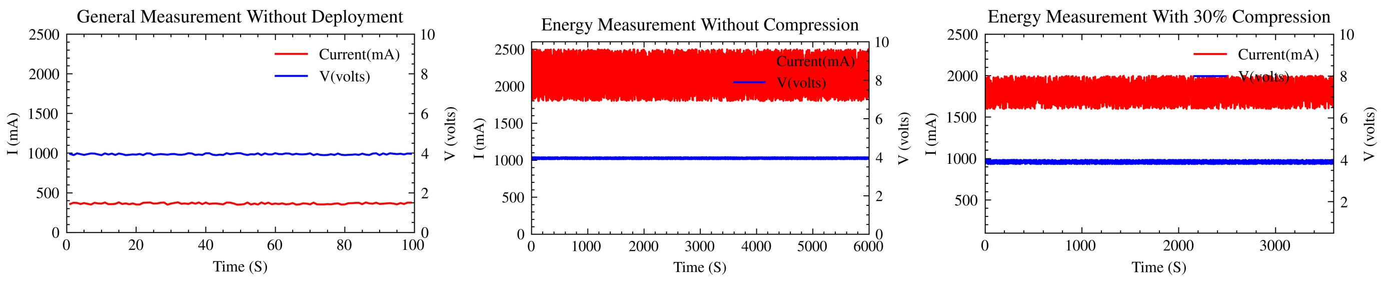

Fig. 5. Energy measurement on HD map DNN model (Approx-35 mentioned in Table 1), Figure (a) shows idle state of the device with default applications active, Figure (b) shows DNN deployment without data compression and Figure (c) shows measurement with 30 % compression on data

Jetson Nano: Python and Pytorch framework is used to deploy the DNN model in the pre-built Linux-OS provided by Nvidia. API provided by Argo is further integrated within the framework for easy data access and visualization.

HV Power Monitor: It is an energy measurement device from Monsoon solutions used for voltage and current measurements on embedded and computing devices. For the experiments, the monsoon device is controlled using raspberry-pi (python + monsoon HV Power monitor API) to provide flexibility and automate measurements.

To test energy consumption of the AI model for a semantic map application, the Approx-35 model mentioned in Table 1 is used. During the measurements, current values are recorded against voltage and time (as shown in Figure 5). Figure 5a) shows measurements on the jetson-nano device in an idle state, with pre-built applications (background apps) and a connected graphic port. The current measure for this default setting is between 350-380 mA. Figure 5a) shows the measurement recording for 100 seconds. The Approx-35 model is used with the Argo validation set for the second scenario. Since the validation set is large, the measurements are captured for around 2 hours, and the current readings are recorded. As mentioned in the dataset description, depending upon the size of the validation set, the current measurements were in the range of 1820-2400 mA. In the third scenario, a data compression approach is implemented on the test data before it is passed to the DNN model for validation. Figure 5c) shows the current readings for this scenario, which is between 1600-1880 mA. As expected, there is a significant drop in power usage when the data is compressed.

Edge AI and Energy-efficient Vehicular Services

Connected autonomous vehicles are driven using several services. Identifying the services/applications that allow a trade-off between energy and accuracy is essential. Depending upon the services and their fundamentals, such as data-processing approach, computational complexity, onboard, and communication latency, these services can be further divided on the possibility of relaxing the need for full precise operation through approximation. An approximation can also be implemented on the sensed data, using data compression and offloading mechanisms to enhance the onboard energy-saving. In vehicular services that share data, computing and energy optimization can be achieved using data aggregation and resource allocation mechanisms, which are very popular in federated and distributed learning settings.

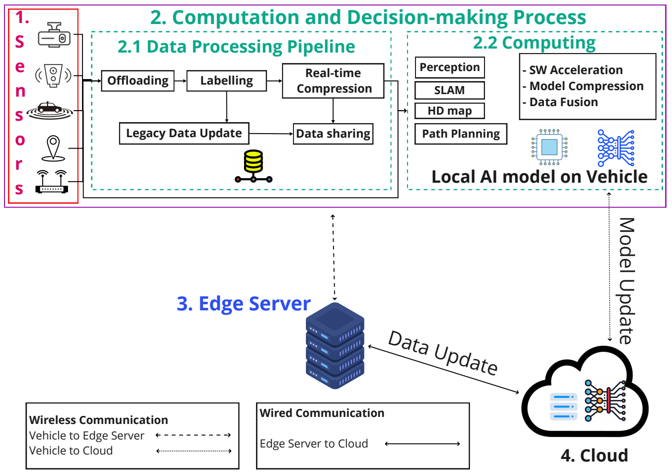

Fig. 6. Edge-AI Pipeline

As the future vehicular ecosystem will consist of distributed edge devices, sensors, servers, roadside units, and heterogeneous networks [15], we can further explore the distributed (machine) learning mechanisms across these devices to save vehicle's onboard energy and use the distributed computation functionality [21]. An envisioned Edge-AI pipeline is shown in Figure 6. The pipeline consists of onboard sensors such as camera, LiDar, radar, GPS, and communication devices in the vehicle. Further, the pipeline includes computation and decision-making, divided into data processing and computing. The data processing pipeline can perform tasks such as offloading, labeling, real-time compression, and data sharing for latency tolerable applications. The following block is computing that performs onboard computation. The computing block also implements model approximation and software acceleration. For low latency applications (e.g., SLAM), the input from sensors can be directly processed (by-passing data processing mechanisms block) at the computing block (vehicles onboard computing unit). In addition, this pipeline also consists of Edge-server and devices placed within the vehicular ecosystem. For latency, tolerable and collaborative applications (e.g., traffic monitoring), sensed data from several vehicles and infrastructure sensors could be offloaded (using vehicular communication) to the edge-server, which also deploys the computation and decision-making block. Furthermore, it consists of a cloud or remote server that maintains the global DNN model, legacy database, and services. Using such a pipeline can open insights for multi-model data processing approaches, efficient video streaming for in-vehicle infotainment, energy-aware adaptive mechanisms, and Edge-AI techniques to facilitate the vehicular services collaboratively within the connected vehicle-Edge ecosystem.

V. Concluding Remarks

This paper explores energy efficiency mechanisms using approximate functions for data-intensive vehicular services at the edge/onboard computing unit. Focus is given to services such as semantic/HD maps, as the task requires heavy volume data processing via AI models, which makes them compute intensives. We explored the potential of AI model approximation using an approximate function for the convolutional layer. Using a convolutional layer/filter is a trend for perception/semantic tasks in connected vehicles, and this layer also occupies the majority of processing power during inference. Our further study can target approximating the fully connected layer and balanced approximation of the whole model. These investigations can further reveal the impact of approximation on the model accuracy and favorable trade-off for energy-accuracy on the resource-constrained devices.

Acknowledgment

The authors gratefully acknowledge funding from European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska Curie grant agreement No. 956090 (APROPOS: Approximate Computing for Power and Energy Optimisation).

References

[1] Peter Arthurs, Lee Gillam, Paul Krause, Ning Wang, Kaushik Halder, and Alexandros Mouzakitis. A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[2] Marcella Astrid and Seung-Ik Lee. Cp-decomposition with tensor power method for convolutional neural networks compression. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 115–118. IEEE, 2017.

[3] Boris Bučko, Martin Michálek, Katarína Papierniková, and Katarína Zábovská. Smart mobility and aspects of vehicle-to-infrastructure: A data viewpoint. *Applied Sciences*, 11(22):10514, 2021.

[4] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[5] Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. Rethinking bisenet for real-time semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9716–9725, 2021.

[6] Liu Fangxin, Zhao Wenbo, Wang Yanzhi, Dai Changzhi, and Jiang Li. Ausn: Approximately uniform quantization by adaptively superimposing non-uniform distribution for deep neural networks. *arXiv preprint arXiv:2007.03903*, 2020.

[7] Jianbo Guo, Yuxi Li, Weiyao Lin, Yurong Chen, and Jianguo Li. Network decoupling: From regular to depthwise separable convolutions. *CoRR*, abs/1808.05517, 2018.

[8] Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. *arXiv preprint arXiv:1910.13984*, 2019.

[9] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019.

[10] Dewant Katare and Mohamed El-Sharkawy. Collision warning system: embedded enabled (rtmaps with nxp blbx2). In *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 1–6. IEEE, 2018.

[11] Dewant Katare and Mohamed El-Sharkawy. Real-time 3-d segmentation on an autonomous embedded system: using point cloud and camera. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 356–361. IEEE, 2019.

[12] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

[13] Michael Krail, J Hellekes, U Schneider, E Dütschke, M Schellert, D Rüdiger, A Steindl, I Luchmann, V Waßmuth, H Flämig, et al. Energie-und treibhausgaswirkungen des automatisierten und vernetzten fahrens im straßenverkehr. *Final report of the study on behalf of the German Federal Ministry of Transport and Digital Infrastructure. Karlsruhe, Germany*, 2019.

[14] Jiachen Mao, Huanrui Yang, Ang Li, Hai Li, and Yiran Chen. Tprune: Efficient transformer pruning for mobile devices. *ACM Trans. Cyber Phys. Syst.*, 5(3):26:1–26:22, 2021.

[15] Weisong Shi and Liangkai Liu. Smart infrastructure for autonomous driving. In *Computing Systems for Autonomous Driving*, pages 173–190. Springer, 2021.

[16] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1007–1015, 2018.

[17] Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank factorization for deep neural network compression. *Neurocomputing*, 398:185–196, 2020.

[18] Morteza Taiebat, Samuel Stolper, and Ming Xu. Forecasting the impact of connected and automated vehicles on energy use: a microeconomic study of induced travel and energy rebound. *Applied Energy*, 247:297–308, 2019.

[19] James Tee and Desmond P Taylor. A quantized representation of probability in the brain. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 5(1):19–29, 2019.

[20] Ruijun Wang, Liangkai Liu, and Weisong Shi. Hydraspace: Computational data storage for autonomous vehicles. In *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pages 70–77. IEEE, 2020.

[21] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, 2019.

[22] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023.

[23] Kelvin Wong, Yanlei Gu, and Shunsuke Kamijo. Mapping for autonomous driving: Opportunities and challenges. *IEEE Intelligent Transportation Systems Magazine*, 13(1):91–106, 2020.

[24] Xuecan Yang, Sumanta Chaudhuri, Lirida Naviner, and Laurence Likforman. Quad-approx cnns for embedded object detection systems. In *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 1–4, 2020.

[25] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

[26] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. NISP: pruning networks using neuron importance score propagation. volume abs/1711.05908, 2017.