2022 IEEE International Workshop on Metrology for Automotive (MetroAutomotive) | 978-1-6654-6689-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/MetroAutomotive54295.2022.9855110

A TinyML Soft-Sensor for the Internet of Intelligent Vehicles

Thommas Flores*†, Marianne Silva*† , Pedro Andrade*†, Jordão Silva*† , Ivanovitch Silva*†,
Emiliano Sisinni‡,Paolo Ferrari‡ and Stefano Rinaldi‡
* Postgraduate Program in Electrical and Computer Engineering (PPgEEC)
†Federal University of Rio Grande do Norte - Natal, Brazil
‡University of Brescia - Brescia, Italy
Emails:*{thommas.flores.101, marianne.silva.086, pedro.meira.055, jordao.cassiano.009, }@ufrn.edu.br,*
ivanovitch.silva@ufrn.br and ‡{emiliano.sisinni, paolo.ferrari,stefano.rinaldi}@unibs.it

***Abstract*—The increased number of sensors in modern cars offers the opportunity to develop algorithms that can monitor and diagnose vehicle performance more efficiently. We present the results of applying and deploying a TinyML model into a typical OBD-II automotive scanner to serve as a soft-sensor and estimate carbon dioxide emissions. A TinyML workflow based on TensorFlow, TensorFlow Lite, and Micro was designed to a 32-bit microcontroller target (Machhina A0™) and considering different quantization methods and Multi-layer Perceptron Regressors (MLP). Train, test, and validation were conducted using real-world data fetched from several kinds of vehicles through an emission measurement system. The results suggest that the soft-sensor can estimate Carbon Dioxide emissions with a Mean Absolute Percentage Error (MAPE) of approximately 27% and processing time averages around 37 to 173 microseconds (depending on activation functions adopted) in the target hardware and using intake manifold absolute pressure, intake air temperature, and vehicle speed as independent variables. The results of this study also demonstrated quantization has a major impact on memory usage. On average, 10 to 17 times less memory is required to achieve the same result on MAPE.**

***Index Terms*—Soft-Sensor, MLOps, TinyML, Quantization, Intelligent Vehicles, Regression, OBD-II.**

I. Introduction

The Internet of Intelligent Vehicles (IoIV) is a set of innovative vehicles connected to the cloud employing vehicle-to-everything (V2X) communications. IoIV applications will allow vehicles to become intelligent objects equipped with sensing platforms, computing facilities, control units, and storage. Furthermore, these vehicles will be connected to any entity (other vehicles, roadside stations, charging/gas stations, cloud servers, among others) via V2X communications [1]. Thus, the IoIV will be built as a multi-tier system of different entities with different roles that can take both client and server positions in taking or providing extensive data services. In this way, IoIV can lead to numerous new applications in various domains such as assisted or autonomous driving and platooning; it is also possible to secure information sharing and learning or traffic control and optimization [2].

Intelligent On-Board Units (OBUs), such as those with advanced driver assistance systems (ADAS), are focusing on the Internet of Vehicles (IoV) space. For example, the OnStar system used by General Motors provides safety information services such as a rescue system, remote fault, and global positioning system. The G-BOOK adopted by Toyota Corporation of Japan uses wireless network technologies such as the Vehicle Information Communication System (VICS) to establish communication links between vehicles and central stations [3]. In addition, Ford and Microsoft have jointly developed a SYNC system for navigation, data transmission, and voice calls using a user's cell phone [4].

There are two different types of sensors that are used for automotive applications. Embedded sensors are also called dedicated or discrete sensors. Those devices that do not have onboard units contain in their design a set of sensors connected to an Electronic Control Unit (ECU) [5]. The ECU information is acquired through the Data Link Connector (DLC), a diagnostic connection port for motorcycles, cars, and trucks. The device connecting to this port is the On-Board Diagnostic (OBD-II), which is used to acquire vehicle parameters such as speed, engine and water temperature, battery charge rate, and error codes for fault detection [6], [7].

Whether native or not, the automotive diagnostic devices need to be processed in various applications for the information to become valuable data for the drivers. In most applications, however, this step is performed in an external environment and is currently often in the cloud [8]. Although many IoT applications use large bandwidth networks, for example, 5G, this technology in critical systems such as Advanced Driver Assistance Systems (ADAS) requires that data be processed in the early stages reducing latency and association [9].

Within this context and with the advent of the technological advancement of microcontrollers systems and the quantization techniques of machine learning (ML) algorithms, a new paradigm called TinyML emerged. The innovation of TinyML is the capacity to perform data inference through ML algorithms in ultra low power devices, typically in the range of *mW*, and thus break the traditional limitation of the need for high computational power and minimal hardware setup for the implementation of these algorithms [10]. In addition, performing inference on the device allows for more excellent responsiveness and privacy while avoiding the energy cost associated with wireless communication [11].

This paper proposes methods to use machine-learning algorithms in embedded systems that collect data from vehicles' onboard diagnostics systems. Furthermore, the proposed methods use soft sensors to estimate the amount of carbon dioxide generated during the combustion reaction in the vehicle engine. All these techniques help analyze various vehicle data.

Considering the contributions of the mentioned work, it is possible to highlight: 1) Development of a workflow and methodology for the implementation and deployment of a TinyML model into a typical OBD-II automotive scanner; 2) A performance evaluation considering different quantization techniques and multi-layer perceptron regressors to train, validate, test and deploy TinyML models to a typical OBD-II automotive hardware; 3) Design, implement and validate a soft-sensor regressor to estimate carbon dioxide, deployed in an accurate OBD-II automotive scanner, using intake manifold absolute pressure, intake air temperature, and vehicle speed as independent variables.

The remainder of this paper is organized as follows. Section II presents the related works whereas Section III describes the process of developing a soft sensor to estimate CO_2 emissions. Section IV discusses the results and the main achievements of the study, and finally, Section V remarks on conclusions and future studies.

II. Related Works

After reviewing the literature, several papers were found that influenced the research carried out and, therefore, contributed to the development of the proposed solution.

During the past decades, modern vehicles have been equipped with a system called On-Board Diagnostics (OBD) that is connected with Engine Control Unit (ECU) [12]. OBD gathers sensor data and monitors the performance of the emission control systems or components. When the performance degrades, this is reported by malfunctions indicators to the vehicle drivers [13].

Several researchers have given the OBD-II data collected from cars a machine learning upgrade. The vehicles send the information via networks and can be interpreted to perform vehicle maintenance or security tasks. For example, in [14], the authors create an infrastructure to monitor vehicular pollution based on real-time crowd sensed data processing. Not only that, but this mechanism also shares information about CO_2 emissions, potentially providing valuable data for city planning and management.

In [15], the authors applied two types of unsupervised machine learning methods to classify drivers based on fuel efficiency and driving behavior. They used data from an On-Board Diagnostics-II (OBD-II) device transmitted via 4G to a cloud platform. In the first phase, spectral clustering was used to study the macroscopic relationship between driving behaviors and fuel consumption. In the second phase, driving behaviors were integrated with environmental information to generate a model of the relationship between driving behavior and corresponding fuel consumption characteristics. The authors found that their method could effectively identify relationships between driving behavior and fuel consumption.

Finally, in the work [16], the authors use two methods to monitor the amount of carbon dioxide emitted by vehicles: air mass flow and velocity density. The results show that their solution could be helpful for traffic control systems in cities because areas with the highest concentration of pollution usually comprise crossings, traffic lights, and congested regions. However, their method has one disadvantage: it depends on the car engine's mechanical characteristics and the fuel's chemical composition.

The use of machine learning algorithms in-vehicle diagnostic systems has provided various benefits. However, the need for an internet connection to transfer data from the vehicles limits their use in regions with low bandwidth. To overcome this problem, this work proposes a new methodology for estimating CO_2 levels using machine learning algorithms embedded into a typical OBD-II automotive scanner. The proposed approach is predicted to be more efficient than existing methods that process information off-board in terms of latency, accuracy, and memory savings.

III. TinyML Vehicular Soft Sensor

This section describes developing a soft-sensor to estimate CO_2 using a TinyML regressor model embedded into an OBD-II automotive scanner known as Macchina A0. This hardware enables software customization, making it possible to deploy the TinyML model and investigate the impact of the soft-sensor on memory, latency, and performance of the inferences. An overview of the target hardware architecture and the coupling with the vehicle is described in Figure 1. Note that only three variables are used in the proposed TinyML regressor model.

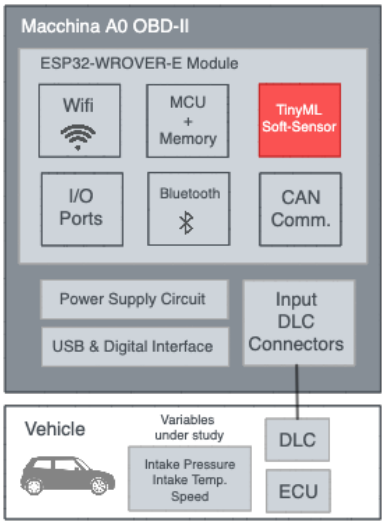


Fig. 1. The hardware architecture used to deploy the TinyML regressor model (soft-sensor).

The study was carried out in an eight-stage workflow, as described in Figure 2, under the best practices of Machine

Learning Operations (MLOps). Each stage of the workflow will be explained in detail next.

A. Collect Data

Four vehicles were selected for this data collection exercise. The course covered was a 5 km loop on a section of road in Natal, Brazil. This road included paved and asphalt stretches and had 2:00 pm to 4:00 pm as the period for the testing experiment. An OBD-II automotive scanner was connected to each vehicle. In addition, the Torque OBDPro app [1] was configured to collect data from intake pressure, intake temperature, speed, and mass airflow (MAF) every 1 second.

B. Process Data

The target variable for the regressor model is the CO_2 emission. However, this variable is not natively exposed to the OBD-II system. Fortunately, CO_2 emission can be derivate from other variables according to Eq. 1:

$$CO_2(g/s) = \frac{MAF(g/s)}{AFR \cdot Density_{Fuel}(g/l)} \cdot CO_2PL \qquad (1)$$

assuming petrol as the fuel, the Air Fuel Ratio (AFR) is configure to 14.7, $CO_2PL(g/L)$ to 2310 and $Density_{Fuel}(g/l)$ to 737 [5].

Note that the MAF values are not always available in vehicles; thus, a demand to derivate CO_2 emission from other variables emerges. In our proposal, the prediction of emissions is based on intake pressure, intake temperature, and speed variables. An additional pre-processing stage is necessary to normalize the variables and avoid the dominant influence of them with a higher values range. For convenience, a min-max normalization was adopted. A sample of the clean dataset is shown in Figure 3.

C. Design a Model

The baseline model used in the experiments is a Multilayer Perceptron Neural Network with three input layers, a single hidden layer with five neurons, and one output layer. The evaluation scenario will vary the number of hidden layers (1 to 16), the number of neurons (1 to 80), and the activation functions (relu or sigmoid) to investigate their impact on the memory, model performance, and inference time of the model embedded in an OBD-II automotive scanner.

D. Train a Model

For training, 70% of the dataset was selected and randomly scrambled. The optimizer used was Adamax, with a learning rate equal to 0.001, β_1 equal to 0.9, β_2 equal to 0.99 and ϵ equal to 10^{-7}. The batch size was configured to 20 and the model trained by 100 epochs. The convergence of the train was based on Mean Square Error (MSE) loss function. In contrast, the evaluation metric was set to Mean Absolute Percentage Error (MAPE). In the end, 0.2% (1121 instances) of the dataset was also used for validation purposes. All these parameters were chosen by convenience after empirical experimentation.

[1] https://torque-bhp.com/

E. Evaluate & Optimize

Two quantization methods provided by the TensorFlow library were adopted in this stage: OPTIMIZE_FOR_SIZE and EXPERIMENTAL_SPARSITY. The former uses a representative dataset version under training to quantize biases and activations functions. The target is to reduce the size, latency, and loss functions. On the other hand, the latter permits optimization by taking advantage of the sparsitivity of the weights from a pruning strategy. Again, the target is to reduce the size and latency of the model artifact, but without prioritizing the loss functions.

F. Convert Model

The model created with TensorFlow is now converted to a lightweight version from the TensorFlow Lite. The quantization approaches mentioned in the previous stage are responsible for optimizing and converting the model into a tiny format. After that, the model artifact is then transformed into a TensorFlow Lite Micro object. Next, each scalar-tensor is encapsulated in a 2D array and wrapped into a 32-bit float list. Finally, the file is converted into a C++ file.

G. Deploy model

The deployment of the tinyML model into the target hardware (Machhina A0) is the last step before making performance evaluations. For the sake of understanding, Figure 4 describes an overview of the end-to-end process of deploying the model. All previous stages of the data workflow were implemented in the cloud using Google Colaboratory. They are represented by step 1 in Figure 4. The next step is to export the model artifact into a C++ object (step 2). Finally, this artifact is transmitted from a serial port where it is deployed to the Macchina A0 (step 3). An ESP32-WROVER-E module supports the target hardware. Additionally, an Data Link Connection (DLC) interface is required to establish communication with the vehicle ECU. The main specifications of the target hardware are shown in Table I.

TABLE I
MACCHINA A0 SPECIFICATION.

Items	Specifications
License	Creative Commons 4.0 Int.
Main module	ESP32-WROVER-E
Microprocessors	Xtensa® 32-bit LX6
Integrated crystal	40 MHz
Integrated SPI flash	4 MB
Integrated PSRAM	8 MB
Clock frequency	up to 240 MHz
Performance	up to 600 DMIPS
Cores	2

H. Make Inferences

The final stage of the proposed workflow aims to evaluate the model's inference performance in the target hardware. In order to conduct experiments, an OBD-II Emulator (step 4 in Figure 4) was configured to emulate the generation of the vehicle data (step 5). Next, the emulator is connected

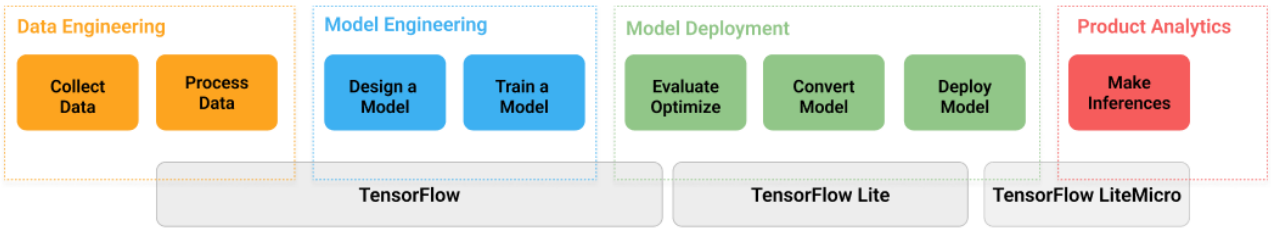


Fig. 2. The proposed eight-stage workflow used to develop and deploy the TinyML regressor model (soft-sensor).

	pressure	temperature	speed	emission
1720	0.617284	0.285714	0.531646	3.624733
2122	0.839506	0.071429	0.898734	3.624733
1213	0.962963	0.464286	0.658228	2.985075
428	0.506173	0.571429	0.000000	1.066098
1621	0.716049	0.000000	0.848101	3.198294

Fig. 3. A sample of the normalized and clean dataset considering the independent (intake pressure, intake temperature, speed) and target (CO_2 emission) variables.

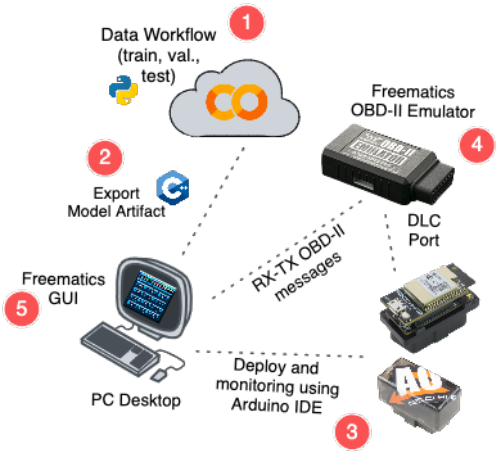


Fig. 4. An end-to-end process for deploy a tinyML model into an OBD-II automotive scanner.

to the target hardware using a DLC port. The Freematics GUI application makes the generation of the data. Finally, a serial port connecting the PC Desktop and the target hardware monitors the inferences and collects all experiment results.

IV. Results and Discussion

This section discusses the results obtained from implementing the proposed workflow to deploy a tinyML into an OBD-II automotive scanner.

Figure 5 describes a comparative evaluation of different models with a single hidden layer, variations in the number of neurons, activation functions, and quantization methods. In all scenarios, the sigmoid activation function demanded more memory resources. The explanation is because its definition is based on a more complicated mathematical formula when compared with relu. In addition, it is also observed that the quantization methods reduced the tinyML model sizes on average by 92% when compared to TensorFlow. In particular, the sparsity quantization becomes less efficient when the complexity of the network increases.

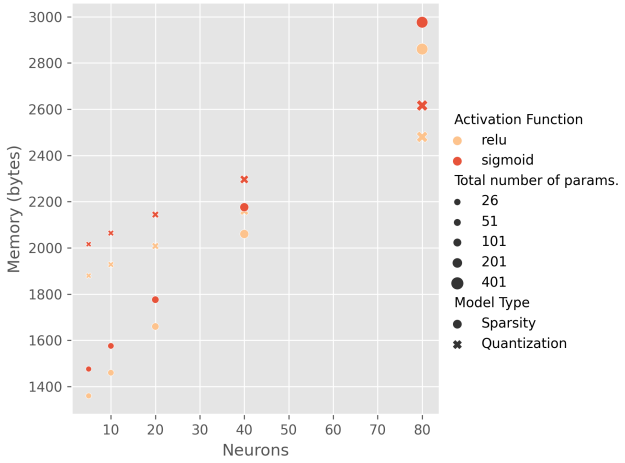


Fig. 5. Comparison between the number of neurons and the use of memory considering different activation functions and quantization approaches for a single hidden layer network.

In order to evaluate the inference time is necessary to define a setup of the deployed model. In our experiments, the sparsity quantization was adopted because it presented the average best result in terms of memory usage. This is confirmed by results of Figures 5 and 8.

A first performance evaluation of the inference is described by Figure 6. It shows a study of varying the number of neurons in a single hidden layer and its influence over the MAPE for different activation functions. The results show that the relu reduces, on average, the inference time by 81.95% and the MAPE by 50.5% compared to sigmoid. However, as expected, the computational demand increase as more neurons is added to the network.

Figure 7 shows the quantification of the error using as a reference the test dataset. This evaluation scenario assesses whether there are accuracy losses when quantizing the TensorFlow model. Results clarify for both activation functions a marginal quantization error. This behavior occurred because the change of tensors from float64 to float32 was insignificant.

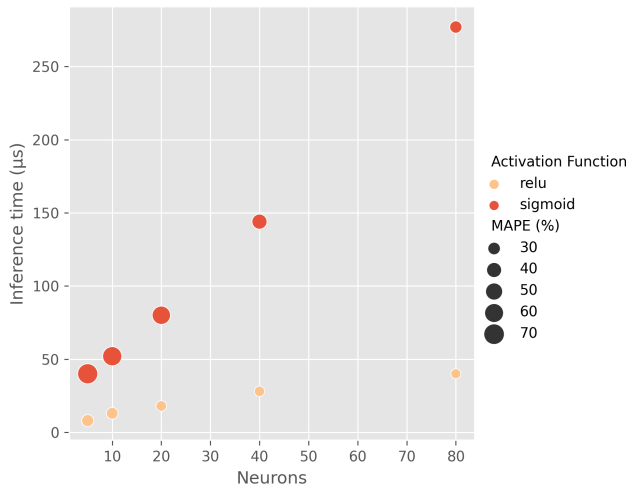
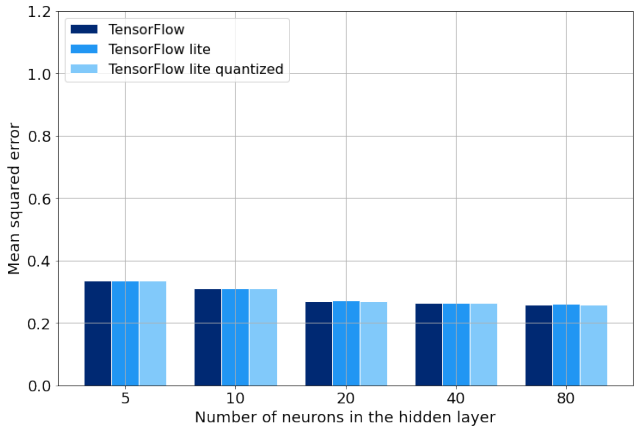
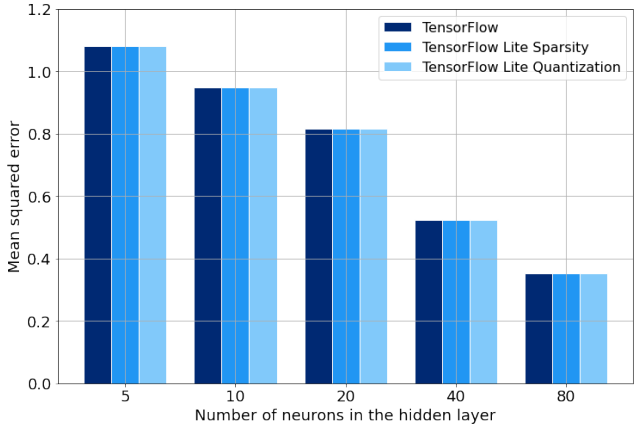


Fig. 6. Comparison between the number of neurons and the inference time for the deployed artifact (TensorFlow Lite Sparsity) considering different activation functions and MAPE performance in a single hidden layer network.



(a) Relu activation function.



(b) Sigmoid activation function.

Fig. 7. Comparison between TensorFlow and quantized models for different numbers of neurons in the hidden layer and activation function.

In addition, the use of relu decreased the MSE by an average of 61.22%.

A different evaluation scenario is described in Figure 8. This study compares the network size (number of layers) and memory use, considering different activation functions and quantization approaches. The study setup supposes using five neurons in each hidden layer. Results show that the sparsity quantization is more efficient regarding memory usage as the number of hidden layers increases, independently of the activation function. Comparing the two quantization methods, the average reduction is 16.37%.

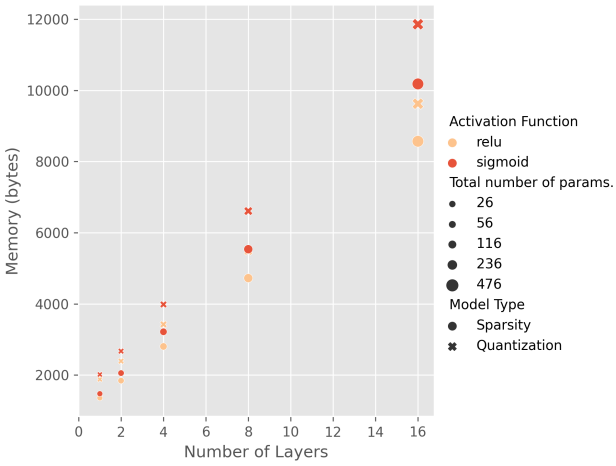


Fig. 8. Comparison between the network size (number of layers) and the use of memory considering different activation functions and quantization approaches.

Figure 9 shows the comparison between the network size (number of layers) and the inference time for the deployed artifact (TensorFlow Lite Sparsity) considering different activation functions and MAPE performance under the use of relu and sigmoid activation functions. The results show that the relu activation function reduces, on average, the processing time by 78.3% and the MAPE by 49% compared to sigmoid, and the performance difference increases the more hidden layers are added.

Overall, this last study is coherent with the previous evaluation scenarios. The adoption of the sigmoid activation function in the envisaged problem has degraded the performance even though it increases the model complexity. On the other hand, relu presented promising results using only two hidden layers. That performance was similar to the use of 16 hidden layers. This result is fascinating because using only two hidden layers reduces memory use more than four times compared to using 16 hidden layers.

V. Conclusion

This work proposes a soft sensor to estimate CO_2 emission in vehicles using TinyML paradigms. The proposed data workflow follows the best practices defined by the MLOps literature. First, the raw data was fetched from an OBD-II automotive scanner connected to the Torque APP and

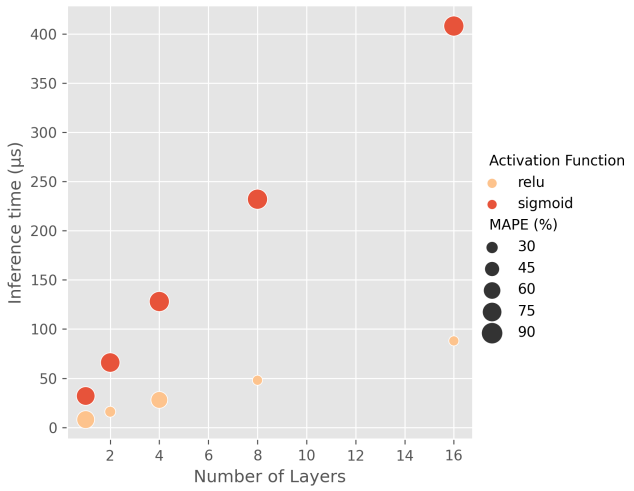


Fig. 9. Comparison between the network size (number of layers) and the inference time for the deployed artifact (TensorFlow Lite Sparsity) considering different activation functions and MAPE performance.

compiled into an actual route and in an urban scenario. Then, after a pre-processing and cleaning procedure, the raw data was served into a training pipeline where a TinyML model was generated. Finally, the model artifact was deployed into a customized OBD-II automotive scanner called Macchina A0.

The proposed workflow experimented with different configurations, including the adoption of a sort of quantization method. Results demonstrated that using quantization methods could potentially increase the efficiency of compression of models reducing the size of deployed artifacts on average by 92%.

Results have also highlighted the influence of activation functions on the performance of the deployed model in the target hardware. The scenarios where the relu activation function was adopted improved paved results compared to sigmoid counterparts. It was observed that improvements were directly proportional to the number of neurons, i.e., the greater the number of neurons, the more significant the difference between the inference time. Furthermore, it was verified that relu reduced the MAPE metric by a factor of 50% when compared to sigmoid.

Results also elucidated interesting issues regarding the influence of quantization methods on the estimation error. For example, models based on pure TensorFlow artifact (without quantization) demonstrated marginal difference regarding estimation error compared to scenarios where sparsity and size quantizations were adopted. A deep exploration brought to light the explanation due to the subtle change in the numerical types of tensors from float64 to float32. This behavior was observed during all experiments and intensified with increasing network complexity.

Another important issue investigated by the envisaged study was the inference time referenced to deployed model in the target hardware. Due to performance results, only the model based on sparsity quantization was deployed. Experiments demonstrated that the use of relu activation function for the problem under study was significantly less sensitive to the increase of network complexity when compared to the sigmoid counterpart.

Overall, the experimental results demonstrated auspicious directions and established answers to the hypotheses and research questions raised in the paper. In future works, it is paved the exploration of more complex network scenarios using convolutions and recurrent architectures.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, with grateful too the Brazilian fostering agency CNPq (National Council for Scientific and Technological Development), Process No. 435683/2018-7.

REFERENCES

[1] P. Ferrari, E. Sisinni, P. Bellagente, D. F. Carvalho, A. Depari, A. Flammini, M. Pasetti, S. Rinaldi, and I. Silva, "On the use of lorawan and cloud platforms for diversification of mobility-as-a-service infrastructure in smart city scenarios," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–9, 2022.

[2] X. Shen, R. Fantacci, and S. Chen, "Internet of vehicles [scanning the issue]," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 242–245, 2020.

[3] P. Stahl, B. Donmez, and G. A. Jamieson, "Supporting anticipation in driving through attentional and interpretational in-vehicle displays," *Accident Analysis & Prevention*, vol. 91, pp. 103–113, 2016.

[4] X. Chang, H. Li, J. Rong, Z. Huang, X. Chen, and Y. Zhang, "Effects of on-board unit on driving behavior in connected vehicle traffic flow," *Journal of Advanced Transportation*, vol. 2019, 2019.

[5] P. Andrade, I. Silva, M. Silva, T. Flores, J. Cassiano, and D. G. Costa, "A tinyml soft-sensor approach for low-cost detection and monitoring of vehicular emissions," *Sensors*, vol. 22, no. 10, 2022.

[6] B. Bánhelyi and T. Szabó, "Data mining and analysis for data from vehicles based on the obdii standard," in *ICAI*, 2020.

[7] D. G. Costa, A. Damasceno, and I. Silva, "Cityspeed: A crowdsensing-based integrated platform for general-purpose monitoring of vehicular speeds in smart cities," *Smart Cities*, vol. 2, no. 1, pp. 46–65, 2019.

[8] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–6, IEEE, 2020.

[9] O. Oviedo-Trespalacios, J. Tichon, and O. Briant, "Is a flick-through enough? a content analysis of advanced driver assistance systems (adas) user manuals," *Plos one*, vol. 16, no. 6, p. e0252688, 2021.

[10] P. Warden and D. Situnayake, *TinyML*. O'Reilly Media, Incorporated, 2019.

[11] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, *et al.*, "Benchmarking tinyml systems: Challenges and direction," *arXiv preprint arXiv:2003.04821*, 2020.

[12] A. Zeb, K. Khattak, A. Aga, Z. H. Khan, M. Sethi, and A. Khan, "On-board diagnostic (obd-ii) based cyber physical system for road bottlenecks detection," *Journal of Engineering Science and Technology*, vol. 17, 04 2022.

[13] G. Signoretti, M. Silva, P. Andrade, I. Silva, E. Sisinni, and P. Ferrari, "An evolving tinyml compression algorithm for iot environments based on data eccentricity," *Sensors*, vol. 21, no. 12, 2021.

[14] J. Oliveira, J. Lemos, E. Vieira, I. Silva, J. Abrantes, D. Barros, and D. G. Costa, "Co2 catcher: A platform for monitoring of vehicular pollution in smart cities," in *2017 IEEE First Summer School on Smart Cities (S3C)*, pp. 37–42, 2017.

[15] P. Ping, W. Qin, Y. Xu, C. Miyajima, and K. Takeda, "Impact of driver behavior on fuel consumption: Classification, evaluation and prediction using machine learning," *IEEE Access*, vol. 7, pp. 78515–78532, 2019.

[16] M. Silva, G. Signoretti, J. Oliveira, I. Silva, and D. G. Costa, "A crowdsensing platform for monitoring of vehicular emissions: A smart city perspective," *Future Internet*, vol. 11, no. 1, p. 13, 2019.