Green Deep Learning: Comparative Study of Road Object Detectors between Jetson Boards and PC

Fabio Sánchez-García, Felipe Arango, Carlos Gómez-Huélamo, Manuel Ocaña, Pedro Revenga, and Luis M. Bergasa

RobeSafe research group, Electronics Department, Universidad de Alcalá, Spain
fabio.sanchezg@edu.uah.es
{juanfelipe.arango, carlos.gomezh, manuel.ocanna, pedro.revenga, luism.bergasa}@uah.es

Abstract. Recent advancements in deep learning have provided powerful tools for intelligent vehicle tasks, particularly in the field of perception. However, achieving real-time performance with low power consumption remains a challenge included in the hot research topic known as green deep learning. In this paper, we present a comparative analysis of various YOLOv5 weights trained on the KITTI and SHIFT datasets using two platforms with different power consumption profiles: the NVIDIA Jetson AGX Xavier and a desktop computer equipped with a NVIDIA GTX 1080 Ti. Our analysis focuses on the average inference time and precision metrics for road objects detection, a key task for intelligent vehicles. Additionally, we apply TensorRT to optimize and accelerate the architecture on both platforms, resulting in significant speed improvements, particularly on the low-power Jetson AGX Xavier (30W). Our ultimate goal is to implement our whole autonomous driving architecture on several Jetson AGX Xaviers connected to a PC where the hyper-realistic CARLA simulator, is replicating the real-world autonomous vehicle environment. We obtain compelling validation results on KITTI and CARLA, achieving real-time performance on a lightweight Jetson AGX Xavier with a powerful object detector such as YOLOv5m.

Keywords: green deep learning, algorithm acceleration, embedded systems, TensorRT

1 Introduction

Deep learning (DL) has suffered huge agitation in the last decade thanks to the introduction of new techniques that improve the precision of detections, classifications or semantic segmentation from sensor data. However, these new techniques usually require massive computations during model training and inference, which have a surprisingly large energy consumption, with its corresponding carbon footprint, as well as having negative effects on the deployment on real-world applications. Green deep learning [2] is an increasingly hot topic

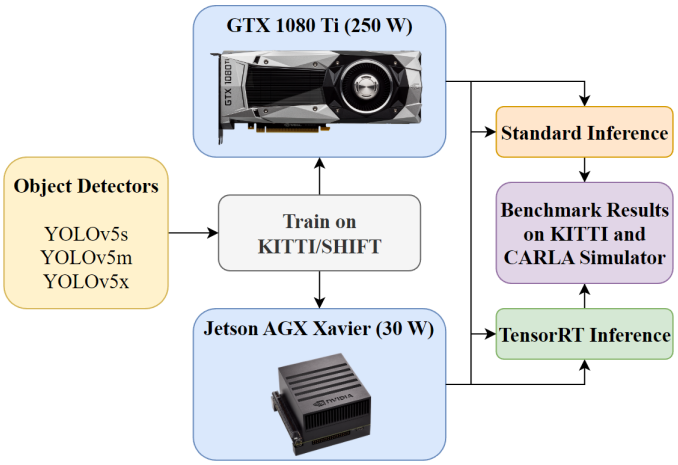

Fig. 1. Work baseline. Inference process summary of platforms and used models.

that appeals to researchers to pay attention to energy usage and carbon emissions during models implementation. The target is to yield competitive results with lightweight and efficient technologies. This paper focuses on energy-efficient DL inference in the context of intelligent vehicles.

Energy conservation is critical in electric autonomous vehicles, as their batteries must maximize the number of kilometres driven. However, reducing power consumption often entails limitations in computational capability. While state-of-the-art DL algorithms prioritize complexity and outcomes, they frequently overlook some practical aspects: achieving real-time performance in real applications and in an energy-efficient way. Having an extensive driving architecture that runs at 30 FPS on a high-powered device is futile if it cannot be implemented on a vehicle's embedded system. Introducing large GPUs like Nvidia's high-end clusters (DGX) is impractical due to their high power consumption and size, necessitating the exploration of alternative embedded platforms for running inference models in automotive applications.

Taking advantage of acceleration techniques can narrow the barrier between hardware with different power consumption, carrying the battle of efficiency to common ground. Thanks to tools such as TensorRT we can increase the FPS of a model running on a Jetson AGX Xavier to unimaginable numbers able to compete on equal terms with more powerful and power-consuming hardware in an efficient way.

In this work, we bring two devices face-to-face with each other to analyse their performances on running different YOLOv5 weights for object detection in autonomous driving environments. To center the focus in autonomous vehicles, the used weights are trained and evaluated using the KITTI [3] and SHIFT [4] datasets. The acceleration is performed using TensorRT (v8.4.1), which requires of CUDA (v12.0).

With that in mind, our contribution lies in providing a comprehensive analysis of weights and hardware platforms that can be beneficial for energy efficient intelligent vehicle design. This analysis highlights the potential to achieve model throughput across diverse devices with varying power consumption, through the implementation of acceleration techniques based on TensorRT. As a result, we present a study that assesses the feasibility of utilizing lightweight Jetson boards in automotive applications onboard vehicles. Comparative results are obtained over two public datasets (KITTI and SHIFT) and while our whole autonomous driving stack [5] is moving in the hyper-realistic CARLA simulator environment [6] (Fig. 1).

2 Background

2.1 Nvidia Jetson AGX Xavier

JAGX is an embedded device designed to be used on autonomous machines. It excels in high-performance DL tasks, thanks to its multiple data-parallel accelerators. Powered by an octa-core Nvidia Carmel ARMv8.2, it has a Volta architecture GPU with 512 Nvidia CUDA cores and 64 Tensor cores and two DL accelerators (DLA). These DLAs are fixed-function accelerators that target deep learning operations supporting typical DL layers such as convolution, pooling, activation or fully connected.

JAGX has a shared memory between GPU and CPU, which leads to faster access between processing units. This board also allows to switch among different power modes (10W, 15W and 30W) or even turn on and off different CPU cores, giving total freedom to the user when it comes to power management. However, the reduction in power consumption entails a diminution of performance, which is a trade-off that always needs to be considered.

2.2 Nvidia TensorRT

TensorRT (TRT) is a SDK (Software Development Kit) for high-performance DL inference on Nvidia devices. It is capable of performing different runtime optimisations lowering the latency and enhancing the throughput. When a model is accelerated, a hardware-dependant *engine* is created. Afterwards, the engine is used to perform the inference. This engine is created using five different strategies offered by Nvidia:

- **Precision reduction**. This technique, also called quantization, consists of transforming the precision of the model in a smaller size (FP32, FP16 or INT8). The reduction allows the GPU to perform operations faster, which generally speeds up the inference.
- **Kernel auto-tuning**. TRT is capable of choosing the best data layers and algorithms depending on the GPU platform where the acceleration is taking place. This is one of the reasons why TensorRT applications are hardware dependant, meaning that the same model will have different engines in different hardware architectures.

- **Layer fusion**. TRT can fuse layers vertically or horizontally, so that when repeated structures are found, they are fused in a single operation, improving the inference speed of the engine. This is useful since most models consist of repeating an specific structure (such as convolution, batch normalisation and activation) several times.
- **Buffer reutilization and tensor dynamic memory**. TensorRT allows to reuse memory localisation on some embedded systems to reduce the memory footprint of the model.
- **Multi-stream execution**. TRT allows the execution of several inferences asynchronously in multiple streams. This enhances the performance since the threads executed in a single CUDA core are increased.

3 Related Work

Acceleration methods are some of the most used green technologies and have reunited increasing attention in recent years due to their immense potential. TensorRT emerged onto the scene in 2018, increasing its popularity gradually. However, it was in 2019 when its importance started to rise. Researchers are increasingly focusing their efforts on integrating TRT into their models, leading to an exponential growth of this instrument.

TRT has a remarkable versatility, making it applicable across a wide range of DL implementations, from tasks like image classification to semantic segmentation and object detection, being this last the main focus of this work. Furthermore, multiple researchers have dedicated their efforts to integrate theses acceleration techniques with Jetson devices for efficient model deployment.

3.1 Acceleration and speed-up

A research on the acceleration effect is performed in [7], where a first approach to the optimisation and its explanation are exposed. The results observed support the speeding up. In [8], the methodology behind the acceleration is analysed while exploring PyTorch, ONNX and TensorRT for various ResNet and SqueezeNet models, obtaining accelerations by a factor of 2. In [9], a two-stage facial direction detection is proposed, where they compare the different results on accelerating the first stage, the second or both, which is an interesting option for multi-stage models.

In [10], they present a modified YOLOv5 architecture obtaining around 47 FPS, but losing some accuracy in the process, which does not happen when using TensorRT. They also implement their network in CARLA.

3.2 Deployment on embedded devices

In [11], they propose a real-time approach for lane detections using ResUNet++, CARLA simulator and NVIDIA Jetson Nano, obtaining inference times around 22 ms. [**?**] implements different MOT algorithms such as DeepSORT in RTX 2080Ti, Jetson TX2 and JAGX, with FPS around 46, 10 and 16 respectively. This shows again how embedded systems are far behind GPUs in workstations.

3.3 Acceleration and deployment on embedded devices

This is the primary focus of state-of-the-art research. Accelerating a model for a high-end device is not as impactful as optimizing it for a low-power system. Hence, many researchers are trying to enhance classical or custom architectures to achieve real-time performance in embedded systems.

An end-to-end methodology to optimise DL inference models on Jetson boards is analysed in [13]. This work consists of a long procedure to improve and increase performance of some different YOLO versions. A detection system for dirty-eggs based on a YOLOv4 accelerated with TensorRT is proposed and deployed in a Jetson Nano in [14], with impressive results for such a low consuming device reaching 2.3 FPS. An evaluation of YOLOv3-tiny and EfficientDet-Lite is performed in [15], where the hardware used is a Jetson TX2 and the results are evaluated on the KITTI dataset, aiming for real-time vehicle detection, with FPS between 21.5 and 36.9 for 640x640 images. A review on the possibility to introduce deep neural networks on single board computers (such as Jetson Nano) is investigated in [16], where they obtain around 25 FPS using YOLOv4-tiny thanks to the addition of TensorRT.

To the best of our knowledge, so far, there has not been any project that has studied the performance of YOLOv5 on a JAGX operating on the CARLA simulator.

4 Methodology and Comparison

4.1 Hardware Platforms

Since the acceleration is hardware-dependant, hardware is crucial when using TensorRT. As previously mentioned, our comparison focuses on two platforms: a PC desktop with a NVIDIA GTX 1080 Ti and the Jetson AGX Xavier. In Table 1, we observe that the GPU is twice as fast, but it drops down when it comes to FP16 performance. On the contrary, JAGX works better with quantized models, while the GPU copes better with generalist models, typically generated using FP32 data. It is worth noting that the power consumption of the GPU is between eight and nine times higher compared to the Jetson module. These results, along the power consumption data, highlight the importance and potential of the JAGX as an embedded system for autonomous vehicles. These factors significantly impact the final results.

4.2 Comparison Metrics

Evaluation metrics play a crucial role in giving significance to the results. In this case, we consider two key metrics: *mean Average Precision* (mAP) and inference time. This first metric can be evaluated depending on the chosen confidence intersection over union (IoU) thresholds. On the one hand, we evaluate 50 % IoU, commonly known as mAP@0.5, and a compound mAP, denoted as

Table 1. Hardware Comparison. Power consumption, and TFLOPs in FP32 and FP16 of the hardware studied.

			Perform. (TFLOPs)	
Platform	**Power**	**GPU Clock**	**FP32**	**FP16**
PC (GTX 1080 Ti)	250 W	1481 MHz	11.3	0.177
Jetson AGX Xavier	<30W	854 MHz	1.4	2.8

mAP@[0.5:0.95], which provides an average across different IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05.

Furthermore, the inference time, another critical factor, is also measured. To obtain accurate measurements, we use CUDA Events, since CUDA is asynchronous and all operations are performed in GPU. The inference time is calculated as an average of multiple iterations measured in milliseconds. The inverse of the inference time is commonly referred to as frames per seconds (FPS) of hertzes (Hz), providing a practical measure of the system's real-time performance.

4.3 Inference Methods

The main source used in this work is the ultralytics repository from YOLOv5 [17]. This allows us to perform different export options, such as a TensorRT engine, controlling several parameters like image size, precision, IoU thresholds or configuration file among others. The repository also contains validation files that enable the evaluation of the model's performance on certain dataset.

To adapt the detectors to the work environment, we fine-tune the original pretrained COCO weights of YOLO on KITTI, a dataset that focuses on offering real-world computer vision benchmarks for autonomous driving, and on SHIFT, a synthetic dataset which contains CARLA images. In this context, we perform 50 epochs utilizing the 7481 training images from KITTI and 7000 images from SHIFT, with both datasets split into 70% training images and 30% validation images. The labeled classes consist of car, pedestrian and cyclist. Additionally, Table 2 shows the features of the weights.

To cover a wide range of possibilities, we utilized various weight configurations. The original checkpoints include YOLOv5s, YOLOv5m and YOLOv5x, which represent small, medium and large variations, respectively. Naturally,

Table 2. Weights comparison. Number of layers, parameters and operations required of the weights studied.

Weights	**Layers**	**Parameters**	**GFLOPs**
YOLOv5s	157	7018216	15.8
YOLOv5m	212	20861016	47.9
YOLOv5x	322	86186872	203.8

larger weight configurations tend to obtain better results. This will be demonstrated in the final section of the work.

After retrieving the pretrained weights, we proceed with fine-tuning them and creating accelerated models for both platforms, utilizing quantization in both FP32 and FP16. This step is important as it is crucial to note that the model accelerated for the GTX 1080 Ti is not compatible with the Jetson AGX, and vice versa. Once this process is completed, the following steps involve performing validation across all weights configuration in both datasets.

Following the validation phase over open-source datasets, which offers valuable and noteworthy results, we take a step further by integrating our autonomous vehicle architecture into the Robot Operating System (ROS) [18]. This integration allows us to visualize the ego vehicle, cameras, and other agents in the simulated environment using the RVIZ simulator [19]. With this in mind, we conduct an experiment by running our stack's architecture with YOLOv5 under two distinct conditions. To achieve a better domain adaptation in CARLA, we utilize the fine-tuned weights with the SHIFT dataset.

1 Whole architecture and simulator launched on PC.

The initial phase of developing an autonomous driving architecture involves simulating its functionality under ideal conditions. In this stage, all layers of the architecture, including perception layer where the YOLOv5 operates, are executed on the PC. However, it is important to notice that this setup is far from deployment on an actual autonomous vehicle, because incorporating a PC into a vehicle is inefficient. Nonetheless, it is essential to study this preliminary phase for drawing conclusions that can guide the next steps of the project.

2 Perception layer executed on JAGX and simulator launched on PC.

This approach is more efficient and realistic, resembling a real-world scenario. The architecture of our modular autonomous driving architecture consists of various layers, including the perception layer. In this last layer, sensors gather information about the surrounding environment and transmit it to their respective processing units. In this configuration, the simulated world and sensors in CARLA are executed on the PC, while the collected information (i.e. images) is transmitted to the perception processing unit, running in a Jetson. The Jetson processes this information and retrieves the bounding boxes of detected objects in the environment. In our final version, the whole layers will be running on several JAGXs.

The CARLA scenario used in this additional experiment is based on Euro-NCAP [20] protocols. Specifically, we have implemented Car to Pedestrian Farside (CPF) scenario, in which a pedestrian crosses the street at a predetermined speed. To ensure consistency, all experiments have been performed under identical conditions within the CARLA Town 4 environment for maintaining uniformity across all the results.

Once this has been made, we measure the inference time of both conditions using CUDA Event, to compare it with the times obtained in the validation. The results obtained are thoroughly analysed in the following section.

5 Results

This section presents the inference results for the different experiments conducted in this study, regarding the validation in KITTI and SHIFT, and the test with CARLA simulator.

– **Validation with KITTI and SHIFT.**

Table 3 provides a comprehensive review of all the obtained results. It is worth noting that these results were generated using images of size 640x640, ensuring consistency across all the experiments.

The observations from the table confirm the initial hypothesis of this study. One crucial finding is that the accelerated models (TRT) demonstrate negligible accuracy loss regarding to not accelerated ones (GPU). This is important because if the precision was significantly compromised, the acceleration would have limited practical value. The fine-tuned weights achieve mAP@0.5 scores around 0.9 in KITTI and 0.85 in SHIFT, while mAP@0.5:0.95 scores range between 0.6 to 0.7 in KITTI and 0.55 to 0.65 in SHIFT. The slight loss of performance on SHIFT is attributed to the challenging nature of this dataset, which includes day and night data and various weather scenarios. As expected, YOLOv5x, with its heaviest weights, achieves the highest accuracy results. These findings validate the effectiveness of the acceleration techniques employed while maintaining satisfactory levels of precision. It is also important to note that, since we are performing 640x640 image size inference, the inference times do not vary for the same weights. This demonstrates that the inference time is independent of the dataset when the input data is resized to the same size.

Furthermore, the fastest inference times of each model are observed on the PC. In this case, FP32 inference is slightly faster than FP16 in the PC. The gap between FP32 and FP16 inference times is not significant, contradicting the theory that FP16 should be faster. However, as demonstrated in Table 1, the PC is not optimized for FP16 inference since its general usage typically involves FP32. Therefore, in this specific hardware configuration, the acceleration to FP16 does not involve significant improvements in terms of inference speed.

On the contrary, the Jetson boards exhibits inference times that are approximately five times slower compared to the PC. This outcome is anticipated due to the disparity in power consumption and GPU clock frequency between the two devices. However, the results are astonishing when we consider the inference times in FP16. The gap between the PC and JAGX is significantly reduced, thanks to the specialized nature of the JAGX in FP16 inference. The JAGX system is designed specifically for FP16 acceleration, resulting in rapid inference speed that can rival a powerful GPU like the 1080 Ti. Additionally, it is worth noting that the JAGX is much more energy efficient and is smaller in size, making it a viable option for integration into a real vehicle.

Table 3. Validation of all the different weights used in the work in both datasets: KITTI and SHIFT.

	PC KITTI Inference											
	GPU				TRT - FP32				TRT - FP16			
Weights	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95
YOLOv5s	6.8	147.1	0.895	0.608	**3.9**	**256.4**	0.894	0.608	3.9	256.4	0.894	0.608
YOLOv5m	10.5	95.2	0.918	0.666	**7.1**	**140.8**	0.917	0.662	7.4	135.1	0.917	0.661
YOLOv5x	26.9	37.2	**0.929**	**0.697**	**22.5**	**44.4**	0.927	0.696	22.6	44.2	0.927	0.694
	Jetson AGX KITTI Inference											
	GPU				TRT - FP32				TRT - FP16			
Weights	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95
YOLOv5s	30.9	32.4	0.895	0.608	20.3	49.3	0.894	0.608	**16.8**	**59.5**	0.894	0.608
YOLOv5m	51.5	19.4	0.918	0.667	40.4	24.8	0.917	0.662	**19.4**	**51.5**	0.917	0.661
YOLOv5x	149.0	6.7	**0.929**	**0.697**	141.2	7.1	0.927	0.696	**45.9**	**21.8**	0.927	0.694
	PC SHIFT Inference											
	GPU				TRT - FP32				TRT - FP16			
Weights	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95
YOLOv5s	6.9	161.3	0.806	0.566	**3.8**	**263.2**	0.805	0.566	4.0	250.0	0.805	0.566
YOLOv5m	10.4	96.2	0.855	0.637	**7.3**	**137.0**	0.854	0.634	7.6	131.6	0.854	0.633
YOLOv5x	26.7	37.5	**0.881**	**0.666**	**22.1**	**45.2**	0.879	0.665	22.9	43.7	0.879	0.664
	Jetson AGX SHIFT Inference											
	GPU				TRT - FP32				TRT - FP16			
Weights	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95	Validation Inference Time (ms)	FPS	mAP @0.5	mAP @.5:.95
YOLOv5s	31.0	32.3	0.806	0.566	20.2	49.5	0.805	0.566	**17.0**	**58.8**	0.805	0.566
YOLOv5m	51.3	19.5	0.855	0.637	39.9	25.1	0.854	0.634	**19.3**	**51.8**	0.854	0.633
YOLOv5x	149.6	6.7	**0.881**	**0.666**	141.8	7.1	0.879	0.665	**46.5**	**21.5**	0.879	0.664

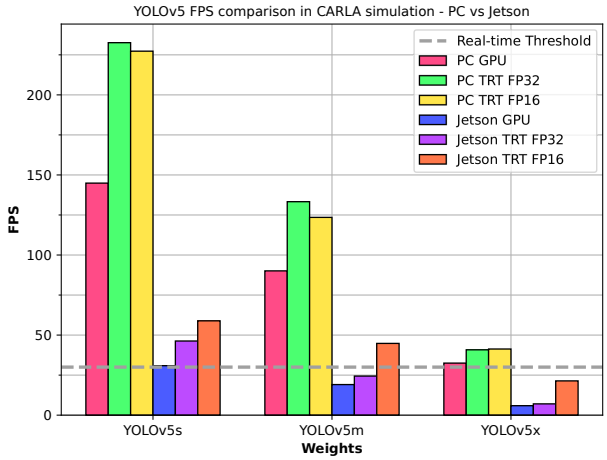

Fig. 2. FPS for CARLA. FPS obtained in the experiments with the CARLA simulator. The dashed line marks a strict real-time in 30 FPS.

– **Additional experiment in CARLA.**

As previously mentioned, we perform an additional experiment in both conditions explained in Section 4. In this situation, Fig. 2 shows the FPS obtained in CARLA for the different weights on each platform using the SHIFT fine-tuned weights for a better domain adaptation. It is important to note that the FPS obtained are very similar to the ones obtained in SHIFT validation, since we are measuring YOLOv5 inference. This figure provides valuable insight for understanding the accelerations achieved.

For each weight, the first three bars represent the PC, while the last three bars represent the JAGX. It is evident from the graph that the PC achieves the highest FPS in all cases. Another worthy observation is that both devices achieve real-time performance with YOLOv5s. However, when it comes to YOLOv5m, the JAGX lags behind. Nevertheless, thanks to the FP16 acceleration, the JAGX is able to handle the same weights in real-time. A similar trend is observed with YOLOv5x, where the accelerated FP16 version matches the unaccelerated FPS of the previous weight. This indicates that accelerations allows us to use higher-order weights without sacrificing speed or accuracy, potentially resulting in an improvement of over 0.06 mAP@0.5:0.95, which can be significant depending on the application.

The observed accelerations are remarkable, achieving acceleration factors of up to 2 times the base speed. This indicates that both PC and JAGX exhibit significant over-performance, competing with more expensive, powerful and power-consuming hardware. This can be viewed as a hardware upgrade, as TensorRT effectively enhances model performance through hardware-specific optimizations which play a crucial role in embedded systems where efficiency is key.

6 Conclusions and Findings

In this work, we proposed a comparative analysis of inference time and accuracy between different YOLOv5 weights executed on two different platforms with different power consumption and throughput: GTX 1080 Ti and Nvidia Jetson AGX Xavier. To adapt the weights to the real world and the environment in the CARLA simulator, we fine-tuned the pretrained weights using the KITTI and SHIFT datasets.

First, we obtain temporal and accuracy metrics performing validation over both datasets in both their accelerated and unaccelerated versions. YOLOv5m, which is the model with the best balance between inference time and accuracy metrics, obtains 0.918 mAP@0.5 in KITTI and 0.855 in SHIFT. When it comes to inference time, YOLOv5m reaches around 95 FPS in the GPU, but only achieves 19.5 FPS in the JAGX. However, this changes after applying acceleration, with YOLOv5m reaching 51.5 FPS in FP16 precision in the Jetson module.

After evaluating the results, we have used the simulation-oriented weights fine-tuned in SHIFT in our autonomous driving architecture in two different scenarios: on a PC running CARLA and on a Jetson AGX Xavier connected to a PC running the simulation. In this situation, the inference times observed are closer to a real implementation. We demonstrate that the FPS obtained are similar to the datasets validation inference times, with only a small deceleration attributed to computational overload in the hardware due to other processes.

In summary, our CARLA implementation showcases the performance of the JAGX platform for efficient deployment of embedded systems in autonomous vehicles. Results show the promising capability of YOLOv5m with TensorRT acceleration, achieving nearly 50 FPS without compromising accuracy. Results also highlight the potential of JAGX embedded system enabling accurate and efficient inference for autonomous driving tasks with impressive real-time results and high accuracy. The combination of these findings underscore the importance of deploying embedded systems alongside accelerated inference to ensure efficient and accurate behaviour in autonomous driving, conclusions aligned with the green DL deal.

References

1. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. 147, 195?197 (1981). doi:10.1016/0022-2836(81)90087-5
2. J. Xu, W. Zhou, Z. Fu, H. Zhou, and L. Li, "A survey on green deep learning," arXiv preprint arXiv:2111.05193, 2021.
3. A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," International Journal of Robotics Research (IJRR), 2013.
4. T. Sun, M. Segu, J. Postels, Y. Wang, L. Van Gool, B. Schiele, F. Tombari, and F. Yu, "Shift: a synthetic driving dataset for continuous multi-task domain adaptation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 21 371–21 382.

5. C. Gómez-Huélamo, A. Diaz-Diaz, J. Araluce, M. E. Ortiz, R. Gutiérrez, F. Arango, Llamazares, and L. M. Bergasa, "How to build and validate a safe and reliable autonomous driving stack? a ros based software modular architecture baseline," in 2022 IEEE Intelligent Vehicles Symposium (IV), 2022, pp. 1282–1289.
6. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," 2017.
7. Z. Song and K. Shui, "Research on the acceleration effect of tensorrt in deep learning," Scientific Journal of Intelligent Systems Research Volume, vol. 1, no. 01, 2019.
8. Y. Zhou and K. Yang, "Exploring tensorrt to improve real-time inference for deep learning," in 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys). IEEE, 2022, pp. 2011–2018.
9. H.-L. Chen, K.-H. Chen, Y.-T. Hwang, and C.-P. Fan, "Acceleration study of two-stage and deep-learning based facial direction detection on gpu-based edge device," in 2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech), 2022, pp. 429–430.
10. T.-H. Wu, T.-W. Wang, and Y.-Q. Liu, "Real-time vehicle and distance detection based on improved yolo v5 network," in 2021 3rd World Symposium on Artificial Intelligence (WSAI), 2021, pp. 24–28.
11. R. Yousri, M. Mostafa, R. Yasser, M. Shawki, A. Khaled, Z. Mostafa, A. Soltan, and M. S. Darweesh, "A real-time approach based on deep learning for ego-lane detection," in 2021 9th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC). IEEE, 2021, pp. 9–13.
12. G. Chen, Y. Lin, M. Sun, and T. Ik, "Managing edge ai cameras for traffic monitoring," in 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), 2022, pp. 01–04.
13. E. Jeong, J. Kim, and S. Ha, "Tensorrt-based framework and optimization methodology for deep learning inference on jetson boards," ACM Trans. Embed. Comput. Syst., vol. 21, no. 5, oct 2022. [Online]. Available: https://doi.org/10.1145/3508391
14. X. Wang, X. Yue, H. Li, and L. Meng, "A high-efficiency dirty-egg detection system based on yolov4 and tensorrt," in 2021 International Conference on Advanced Mechatronic Systems (ICAMechS). IEEE, 2021, pp. 75–80.
15. T.-H. Wu, T.-W. Wang, and Y.-Q. Liu, "Real-time vehicle and distance detection based on improved yolo v5 network," in 2021 3rd World Symposium on Artificial Intelligence (WSAI). IEEE, 2021, pp. 24–28.
16. R. Ildar, "Increasing fps for single board computers and embedded computers in 2021 (jetson nano and yovov4-tiny). practice and review," 2021.
17. G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, TaoXie, J. Fang, imyhxy, Lorna, Yifu), C. Wong, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, V. Sonck, tkianai, yxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," Nov. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926
18. Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org
19. H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: A toolkit for real domain data visualization," Telecommun. Syst., vol. 60, no. 2, p. 337–345, oct 2015. [Online]. Available: https://doi.org/10.1007/s11235-015-0034-5
20. M. van Ratingen, The Euro NCAP Safety Rating, 08 2017, pp. 11–20.