Optimizing Convolution Operations for YOLOv4-based Object Detection on GPU

Fatima Zahra Guerrouj[1,2,*], *Sergio* Rodríguez Flórez[1], *Abdelhafid* El Ouardi[1], *Mohamed* Abouzahir[2], and *Mustapha* Ramzi[2]

[1]Université Paris-Saclay, ENS Paris-Saclay, CNRS, SATIE, 91190, Gif-sur-Yvette, France
[2]Systems Analysis, Information Processing and Industrial Management Laboratory,
Higher School of Technology of Sale, Mohamed V University, Rabat, Morocco

Abstract. Real-time object detection is crucial for autonomous vehicles, and YOLO (You Only Look Once) algorithms have demonstrated their effectiveness for this purpose. This study examines the performance of YOLOv4 [3] for real-time object detection on an embedded architecture. We focus on optimizing the computationally intensive convolution operations by employing the cuDNN library to achieve efficient inference. The evaluation assesses critical performance metrics, including object detection accuracy in terms of Mean Average Precision (mAP) and inference latency on the embedded architecture. We conduct a comparative analysis using the publicly available KITTI [7] database. The reported results establish a benchmark between the parallelized YOLOv4 model and the baseline implementation, assessing the advantages of cuDNN acceleration for real-time object detection on resource-constrained devices.

1 Introduction

Autonomous vehicles (AVs) have the potential to transform the transportation sector by providing safer, more efficient, and easily accessible mobility solutions. However, for AVs to operate safely and reliably, they need to be able to perceive and understand their environment in real-time with a high degree of accuracy. Achieving this real-time perception involves combining various sensors and robust processing algorithms.

Real-time object detection is a crucial part of this processing pipeline, involving identifying and locating objects of interest (e.g., pedestrians, vehicles, traffic signs) in the environment captured by sensors such as cameras. Convolutional neural networks (CNNs), particularly "You Only Look Once" (YOLO) algorithms, have emerged as a leading approach to object detection due to their powerful feature learning capabilities, offering a good balance between accuracy and computational complexity, and being well-suited to real-time applications such as autonomous driving [2]

However, deploying CNN-based object detectors on resource-constrained embedded architectures, typically integrated in AVs, presents a significant challenge due to limited processing power, memory, and high energy consumption. Optimizing object detection models for efficient inference on embedded architectures is crucial for deploying AVs in the real world [1].

Therefore, this work is developed within a hardware-software codesign methodology [17]. It examines the performance of YOLOv4, a state-of-the-art object detection algorithm, on an embedded architecture for real-time object detection in autonomous vehicles. YOLOv4 shares key computational characteristics with its successors, such as General Matrix Multiplication (GEMM) operations, making our findings relevant to newer versions like YOLOv5 through YOLOv8. The focus is on using the cuDNN library, a powerful tool for accelerating computationally intensive convolution operations in the YOLOv4 model, to quantify the benefits of cuDNN acceleration by analyzing object detection accuracy, inference latency, and resource utilization on the embedded architecture. Ultimately, this study will highlight the feasibility of deploying YOLOv4 for real-time object detection on resource-constrained embedded architectures, paving the way for safer and more reliable autonomous vehicles.

The primary contributions of this study can be summarized as follows:

- **Performance evaluation of YOLOv4 on embedded architectures:** This study provides a detailed analysis of the performance of the YOLOv4 object detection algorithm when deployed on resource-constrained embedded architectures, focusing on real-time applications in autonomous vehicles.
- **Acceleration using cuDNN:** The study explores using the cuDNN library to accelerate the computationally intensive convolution operations in the YOLOv4 model. It quantifies the benefits of using cuDNN in terms of object detection accuracy, inference latency, and resource utilization.
- **Feasibility Analysis for Real-Time Object Detection:** By examining the trade-offs between accuracy, speed, and resource consumption, the research highlights the feasibility of using YOLOv4 for real-time object detection on embedded architectures used in autonomous vehicles.

*e-mail: fatima-zahra.guerrouj@universite-paris-saclay.fr

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

The paper's organization is structured as follows: Section 2 provides a comprehensive review of current object detection algorithms, focusing on their implementation on embedded architectures. Section 3 details the YOLOv4 object detection algorithm, including its architecture and functional blocks. In Section 4, the paper synthesizes techniques for parallelizing convolutional operations to achieve real-time performance on embedded architectures. Section 5 analyzes and discusses the experimental results. Finally, Section 6 summarizes the essential findings and conclusions drawn from the study.

2 Related Work

Recent advancements in deep convolutional networks have revolutionized object detection, particularly in autonomous vehicles where efficiency and accuracy are paramount. One notable contribution by [19] The research investigates the efficacy of different deep learning object detection frameworks on various NVIDIA Jetson architectures, including Nano, TX2, NX, and AGX. It assesses critical performance indicators such as accuracy (mAP and AP), speed (FPS), and resource utilization. Notably, the study emphasizes the pivotal role of the TensorRT library in enhancing inference speed and efficiency while maintaining accuracy. It focuses explicitly on person detection by comparing one-stage and two-stage detectors. The findings underscore the ability of architectures like TX2 and NX to handle intricate models with appropriate optimizations effectively. However, the analysis also acknowledges challenges stemming from memory limitations on specific Jetson architectures, thereby underscoring the significance of optimization techniques such as TensorRT for achieving peak performance.

Furthermore, [20] delves into developing an efficient system for real-time detection of pedestrians and vehicle priority signs. The system is built on the NVIDIA Jetson Nano architecture, known for its low-power capabilities, and is equipped with cameras to capture the surrounding environment. An LCD is also integrated to relay pertinent information to the driver. A custom-trained Convolutional Neural Network (CNN) named SSD-MobileNet was developed for object detection. The model underwent training using diverse road images obtained under varying environmental conditions. The research posits that the system holds promise in enhancing road safety by delivering reliable and cost-effective Advanced Driver-Assistance Systems (ADAS). The system exhibits an average accuracy exceeding 90% in object detection and correct identification (recall rate). However, the processing speed is noted to be suboptimal, at 8.7 frames per second, necessitating enhanced speed for real-time applications. Therefore, further refinements are essential to render the system practical.

Additionally, [13] presents a novel low-latency accelerator architecture tailored to enhance real-time object detection performance. This architecture features a finely-grained column-based pipeline with a padding skip technique to minimize pipeline startup time, a double signed multiplication correction circuit optimizing DSP efficiency, and a pioneering pooling unit with a shared buffer to reduce memory costs specifically for the pooling layer. Demonstrated with the implementation of YOLOv2-tiny on a development board, the architecture achieves remarkable results—a 20.7% reduction in BRAM consumption and a latency reduction of 2.125x to 2.34x compared to previous FPGA accelerators for YOLOv2-tiny, boasting a DSP efficiency of 95.2%. These advancements collectively push the boundaries of real-time object detection capabilities, which is crucial for advancing autonomous vehicle technologies.

Moreover, [14] evaluates the performance of two object detectors, EfficientDet-Lite and YOLOv3-tiny, on the Nvidia Jetson TX2 mobile embedded architecture. Their comprehensive assessment includes various network configurations, leveraging EfficientNet-Lite backbone models and exploring TensorRT optimization alongside post-training quantization techniques. The findings underscore the suitability of YOLOv3-tiny and YOLOv3-tiny-3l models for achieving accurate real-time object detection on the Jetson TX2. In contrast, the computational demands of EfficientDet-Lite models pose challenges, with only the EfficientDet-Lite0 model showing potential for near real-time processing.

The studies concentrate on enhancing hardware and software efficiency, accuracy, and integration. They also showcase notable progress in real-time object detection for self-driving cars. These works focus on improving the speed and reliability of object detection, specifically tailored for embedded architectures in autonomous vehicles. While GPU-based architectures, like the Jetson AGX Xavier, offer substantial performance advantages, alternative hardware options such as Digital Signal Processors (DSPs) and Field Programmable Gate Arrays (FPGAs) play an equally significant role. These architectures offer distinct benefits regarding power efficiency and customization capabilities, making them crucial for optimizing performance and energy consumption in resource-constrained environments. Integrating these diverse hardware solutions can result in a more balanced and efficient embedded architecture, particularly for applications related to autonomous vehicles and similar domains.

However, achieving optimal performance requires a comprehensive understanding of the nuances of the object detection model. When deployed on resource-constrained embedded architectures, this entails a detailed analysis of their architectural designs, component functionalities, and performance characteristics. Understanding these factors is crucial for refining object recognition pipelines and ultimately advancing the goal of ensuring safe and reliable autonomous vehicles. This work represents the initial phase of a hardware-software codesign approach that utilizes GPU-CPU hybrid architectures. By harnessing the power of GPUs for quick prototyping and identifying potential software improvements, this approach lays the groundwork for future development stages. The next phase will incorporate FPGA-CPU hybrid solutions, which offer additional power efficiency and performance optimizations. This two-part strategy emphasizes the importance of iterative design, ensuring that the final embedded solution

maximizes both speed and energy efficiency for real-time applications in autonomous vehicles.

3 Methods

This section provides an in-depth examination of the principal components and functional blocks of YOLOv4. Subsequently, the evaluation metrics employed to measure the network's predictive performance are surveyed. Following this, the dataset utilized for training the object detection models is detailed. Lastly, a comprehensive overview of the experimental setup implemented in this study is presented.

3.1 YOLO Algorithm

The YOLO (You Only Look Once) algorithm family has revolutionized real-time object detection by combining object classification and localization in a single pass through a Convolutional Neural Network (CNN). This approach diverges from conventional methods by predicting bounding boxes and class probabilities for objects in an image simultaneously. This feature provides a significant speed advantage, making YOLO well-suited for real-time applications such as autonomous vehicles.

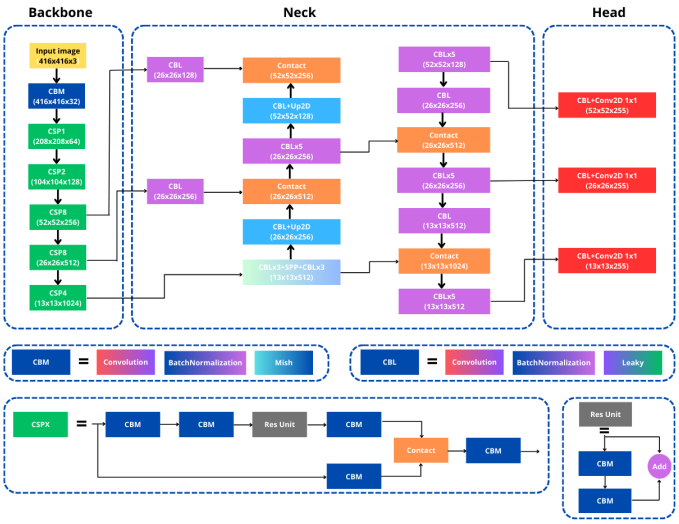

Figure 1. YOLOv4 Functional Blocks [18].

YOLOv4 [3] builds upon its predecessors while addressing their limitations. The architecture and functional blocks of YOLOv4 are illustrated in Figure 1. By leveraging robust backbone networks such as CSPDarknet53 and incorporating Feature Pyramid Networks (FPNs) with Path Aggregation Network (PAN) for enhanced multi-scale feature extraction, YOLOv4 employs multiple prediction heads to detect objects of various sizes. Additionally, it utilizes the Mish activation function to further enhance performance. Regularization techniques, such as CutMix, within the "Bag of Freebies" (BoF) framework are implemented to improve generalization, while the Focus loss function addresses class imbalance issues. Through the integration of these components, YOLOv4 achieves an optimal balance between speed and accuracy, making it a powerful choice for real-time object detection tasks on embedded architectures, including applications in autonomous vehicles.

However, to obtain a thorough understanding of YOLOv4's performance on embedded architectures, it is essential to evaluate its efficiency metrics [4], which include latency and object detection accuracy.

3.2 Evaluation metrics

To assess object detection models effectively, it is necessary to consider various performance metrics beyond accuracy alone. This evaluation specifically emphasizes three key metrics: Mean Average Precision (mAP), Average Precision (AP), and execution time, thereby providing a comprehensive assessment of the models' effectiveness.

Precision [5] serves as an important measure of a model's ability to effectively identify true positives (TP), i.e., accurately recognized objects, while minimizing false positives (FP), i.e., incorrectly classified objects. It is calculated as:

$$Precision = \frac{TP}{(TP+FP)} \tag{1}$$

Recall [5] is also a key metric, as it measures the percentage of actual positive cases that the model correctly identifies, with FN representing False Negatives.

$$Recall = \frac{TP}{(TP+FN)} \tag{2}$$

When assessing object detection models with multiple object classes, Average Precision (AP) offers a detailed evaluation by considering precision and recall across various confidence score thresholds for each detection. The model generates a Precision-Recall curve, and AP summarizes the performance for a single class by averaging precision across all thresholds.

Mean Average Precision (mAP) expands on the concept of AP by computing the AP for each class in the dataset and then averaging them to produce a singular metric representing the model's overall performance across all object categories.

$$mAP = \sum AP_{class}/N \tag{3}$$

Where N is the number of classes.

Both mAP and AP are essential for evaluating a model's ability to detect objects while minimizing false positives accurately.

Execution time, inference time, or latency [6] measures the duration required for a model to process an image and infer detection. This metric is critical for real-time applications, such as autonomous vehicles, where swift and accurate object detection is essential. Typically, execution time is measured in milliseconds (ms) or frames per second (FPS).

By considering mAP, AP, and execution time collectively, we gain a comprehensive understanding of an object

detection model's effectiveness for a specific task. This assessment ensures that models are not only accurate but also efficient and practical for real-world applications.

3.3 Dataset

Training and evaluating datasets that accurately represent real-world driving conditions is essential for developing practical object detection algorithms for autonomous vehicles. The KITTI Vision Benchmark Suite is widely recognized as a premier dataset [7].

The KITTI dataset provides extensive high-resolution stereo camera images from vehicles navigating diverse urban environments. It encompasses various lighting conditions, weather variations, and complex traffic scenes, making it an excellent resource for training object detection models.

In our research, we utilize the KITTI dataset to evaluate the performance of YOLOv4 for real-time object detection on embedded architectures in autonomous vehicles. We focus on detecting three object classes essential for safe navigation: cars, pedestrians, and bicycles. This approach ensures that our evaluation addresses the key elements necessary for the safety and reliability of autonomous driving systems.

3.4 Experimental setup

The NVIDIA Jetson Xavier AGX [8] is an advanced computing module designed to bring artificial intelligence (AI) capabilities to the edge of the network. It features a robust NVIDIA Volta™ GPU architecture with 512 cores, 8-core Carmel ARM CPUs operating at 2.26 GHz, and 16 GB of memory. This combination delivers the computational power necessary to run complex deep learning models like YOLOv4, while maintaining a compact form factor that is ideal for embedded architectures.

4 Parallelization process

Convolutional Neural Networks (CNNs) have significantly transformed the field of computer vision, leading to notable progress in tasks such as object detection. However, the substantial computational resources required for convolution operations present a persistent challenge.

At the heart of a CNN lies the pivotal convolution operation, responsible for discerning image features by applying a filter across the input data [9]. While effective, this process experiences escalating computational demands with increasing input size and filters, particularly in larger models like YOLOv4. The traditional approach, involving nested loops for element-wise multiplications and summations, necessitates heightened parallelism. This inefficiency can present considerable challenges for contemporary processors geared toward optimizing parallel execution. To combat this issue, several optimization techniques have been developed. This study focuses on two critical methods for parallelizing convolution operations to enable real-time object detection in resource-constrained environments, such as autonomous vehicles.

1. General Matrix Multiplication (GEMM): This approach reformulates the convolution operation as a matrix multiplication problem, as shown in Figure 2. By utilizing highly optimized libraries like cuDNN, GEMM significantly speeds up the convolution process [10].

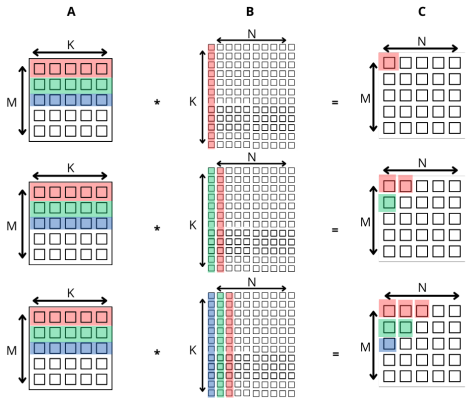

Figure 2. GEMM Operation

2. im2col (image to column): An essential component of GEMM-based convolution, the im2col technique restructures the input data by converting the input image into a column matrix, as shown in Figure 3. Each column of this matrix represents a local region that the filter processes [11]. This transformation enables the entire convolution operation to be carried out as a single matrix multiplication, fully leveraging the optimized GEMM routines.

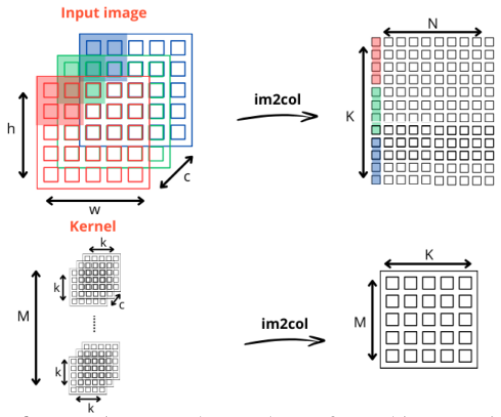

Figure 3. Input image and Kernel transformed into matrix, with N=h*w and K=k*k*c.

The combined application of GEMM and im2col facilitates parallel execution of convolution operations, which is advantageous on hardware architectures such as GPUs and specialized accelerators with multiple processing units. By reducing computational latency and increasing throughput, parallelization makes real-time processing achievable for critical applications like autonomous driving, where timely and accurate object detection is crucial.

This study delves into implementing and optimizing YOLOv4, an advanced object detection algorithm, for deployment on embedded architectures. By harnessing the power of GEMM and im2col for parallel convolution operations, we aim to significantly enhance the performance of YOLOv4 on resource-constrained architectures, ensuring it meets the stringent real-time processing requirements necessary for autonomous vehicles.

5 Results and discussion

This section will comprehensively evaluate General Matrix Multiplication (GEMM) performance on the NVIDIA Jetson AGX Xavier architecture, focusing on CPU and GPU implementations. Initially, we will analyze the performance of a naive GEMM approach on the CPU and GPU to establish a baseline. Following this, we will assess the performance of GEMM optimized by the cuDNN library, a GPU-accelerated library for deep neural networks, to illustrate the substantial gains in efficiency and speed that cuDNN offers for embedded architectures.

5.1 Performance analysis

In this study, we trained the model on a workstation featuring an Nvidia Quadro RTX 6000 GPU with 24 GB of memory and 4608 CUDA cores. Additionally, the workstation contained an Intel® Xeon(R) W-2265 CPU operating at 3.50GHz. The KITTI dataset was divided into a training set, which included 70% of the data (5237 images) and a test set of 30% (2244 images) [10]. To maximize GPU training efficiency, we chose a batch size of 64. The training process encompassed 6000 iterations to ensure the model's stability and prevent overfitting, utilizing a fixed input size of 416x416. Following the training phase, we evaluated the model's inference performance on the Nvidia Jetson AGX Xavier architecture to assess its effectiveness in a real-world embedded environment.

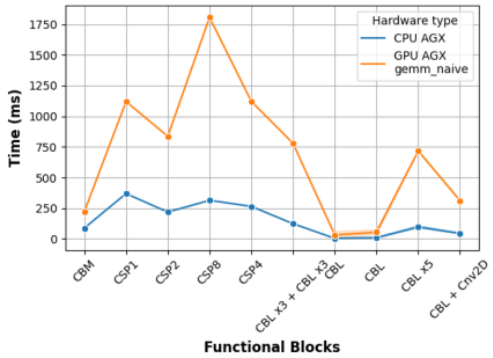

Figure 4. Time consumption of YOLOv4 functional blocks with the GEMM naive.

The provided Figure 4 presents the time consumption analysis of different functional blocks in the YOLOv4 model on both the CPU and GPU of the Nvidia Jetson AGX architecture, with a specific focus on the naive General Matrix Multiplication (GEMM) implementation.

The analysis indicates that the CPU (depicted by the blue line) consistently demonstrates low and relatively stable time consumption across all functional blocks. In contrast, the GPU (illustrated by the orange line) with the naive GEMM implementation exhibits significantly higher time consumption, particularly in computationally intensive blocks such as CSP8 and CBL x5. This disparity emphasizes the inefficiency of the naive GEMM approach on the GPU, which fails to effectively utilize the parallel processing capabilities, resulting in substantially higher latency. Notably, blocks like CSP8 show a drastic spike in time consumption, further accentuating the necessity for optimized GEMM implementations like cuDNN, which is essential for achieving efficient real-time performance.

The overall comparison between the CPU and GPU without optimization suggests that the GPU's performance is subpar in these tasks compared to the CPU, highlighting the crucial importance of employing optimized libraries to fully unleash the potential of GPU hardware in embedded architectures for real-time object detection.

5.2 cuDNN Implementation

In the analysis presented in section 5.1, it was observed that the naive General Matrix Multiplication (GEMM) implementation on the GPU resulted in significant inefficiencies, leading to higher time consumption than the CPU for different functional blocks of the YOLOv4 model. To mitigate this issue, the cuDNN (CUDA Deep Neural Network) library, known for its high optimization for deep learning on GPUs, was integrated to showcase its efficacy in processing these blocks on the GPU [12].

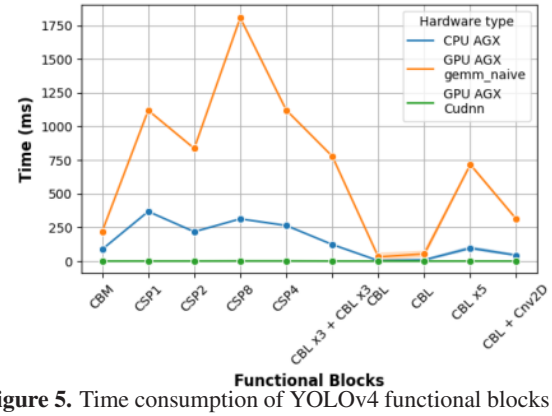

Figure 5. Time consumption of YOLOv4 functional blocks with cuDNN.

Figure 5 illustrates the time consumption of the functional blocks of YOLOv4 on the CPU (depicted by the blue line), GPU with naive GEMM (depicted by the orange line), and GPU with cuDNN (depicted by the green line). The findings are substantial: while the naive GEMM implementation on the GPU exhibited notable latency, incorporating cuDNN significantly reduced the time consumption across all functional blocks. In most scenarios, the cuDNN-optimized GPU (green line) performed equivalently to or even outperformed the CPU, demonstrating significant performance enhancements achieved through optimization. Notably, in computationally intense blocks such as CSP8 and CBL x5, the time consumption with cuDNN was almost negligible compared to the naive GEMM implementation. Leveraging cuDNN illustrates the efficient utilization of the GPU's parallel processing capabilities, rendering it an optimal solution for real-time object detection applications in embedded architectures with restricted resources.

Upon completing the time consumption analysis, our next step involved assessing the precision of YOLOv4 for three crucial classes: cars, pedestrians, and bicycles, as well as its frames per second (FPS) on the GPU AGX architecture while utilizing cuDNN. This supplementary assessment offers a comprehensive insight into not only the

computational efficiency but also the accuracy and real-time processing capabilities of YOLOv4 when optimized with cuDNN on an embedded architecture. This underscores its suitability for autonomous vehicle applications.

Table 1. Performance Analysis of YOLOv4 on the KITTI Dataset.

Model	mAP	AP car	AP pedestrian	AP bicycle	FPS
YOLOv4	83.53%	94.15%	72.22%	84.20%	11.7

The results of Table 1 show that YOLOv4 achieves an mAP of 83.53%, indicating overall solid accuracy in detecting objects. However, despite these promising accuracy metrics, the FPS is recorded at 11.7, which falls short of the real-time processing requirement typically set at 30 FPS or higher for autonomous vehicle applications. This shortfall in FPS highlights the need for further optimizations to enhance the model's real-time processing capabilities on the AGX architecture. Techniques such as model pruning and quantization could reduce computational overhead and improve inference speed, making YOLOv4 more suitable for deployment in real-time scenarios in autonomous vehicles.

6 Conclusion

In this study, we examined the performance of the YOLOv4 object detection model for real-time applications on embedded architectures in autonomous vehicles. Our evaluation of YOLOv4 on the KITTI dataset using the NVIDIA Jetson Xavier AGX architecture and cuDNN acceleration yielded promising results. YOLOv4 demonstrated a high mean average precision (mAP) of 83.53% and average solid precision (AP) values for specific object classes such as cars, pedestrians, and bicycles, showcasing its effectiveness in object detection.

However, a significant challenge lies in achieving real-time processing speeds. The current FPS (11.7) falls short of the 30 FPS threshold typically required for autonomous vehicles, highlighting the need for further optimization to balance accuracy with real-time performance. Future work will prioritize enhancements in critical areas to better adapt YOLOv4 for real-time object detection in autonomous vehicles: Model Compression, Hardware-Software Co-design, and Advanced Parallelization Techniques.

Acknowledgment

This work was partially funded by the Ministry of Europe and Foreign Affairs (Eiffel 116724T) and the CNRST of Morocco (30UM5R2021).

References

[1] J. Wei et al., "Enhanced object detection with deep convolutional neural networks for advanced driving assistance," *IEEE Trans. on Intelligent Transportation Systems*, 2019.

[2] J. Redmon et al., "You only look once: Unified, real-time object detection," *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[3] A. Bochkovskiy et al., "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[4] O. Rainio et al., "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, 2024, Nature Publishing Group UK London.

[5] P. Fränti et al., "Soft precision and recall," *Pattern Recognition Letters*, 2023, Elsevier.

[6] C. Mwitta et al., "Evaluation of inference performance of deep learning models for real-time weed detection in an embedded computer," *Sensors*, 2024, MDPI.

[7] A. Geiger et al., "Are we ready for autonomous driving? the kitti vision benchmark suite," *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012.

[8] S. Groth et al., "Estimating the Execution Time of CNN Inference on GPUs," *MBMV Workshop*, 2024.

[9] X. Zhao et al., "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, 2024, Springer.

[10] F. Z. Guerrouj et al., "Efficient GEMM implementation for vision-based object detection in autonomous driving applications," *Journal of Low Power Electronics and Applications*, 2023, MDPI.

[11] R. D. C. Lera et al., "Hardware-efficient convolution algorithms for CNN accelerators: A brief review," *Anais do XX Encontro Nacional de Inteligência Artificial e Computacional*, 2023.

[12] Z. Nazir et al., "Interpretable ML enhanced CNN Performance Analysis of cuBLAS, cuDNN and TensorRT," *ACM/SIGAPP Symposium on Applied Computing*, 2023.

[13] J. Zhang et al., "A low-latency FPGA implementation for real-time object detection," *IEEE Int. Symposium on Circuits and Systems*, 2021.

[14] H.-H. Nguyen et al., "Towards real-time vehicle detection on edge devices with NVIDIA Jetson TX2," *IEEE Int. Conf. on Cons. Elect.*, 2020.

[15] A. Anupreetham et al., "High Throughput FPGA-Based Object Detection via Algorithm-Hardware Co-Design," *ACM Trans. on Reconfigurable Technology and Systems*, 2024, ACM New York, NY.

[16] J. Zhai et al., "FPGA-based vehicle detection and tracking accelerator," *Sensors*, 2023, MDPI.

[17] Y. Wang et al., "Hardware-Software Co-design for Deep Neural Network Acceleration," *Int. Conf. on Service Science*, 2023.

[18] S. Ali et al., "Improved YOLOv4 for Aerial Object Detection," *Signal Processing and Communications Applications Conf.*, 2021.

[19] O. Byzkrovnyi et al., "Comparison of Object Detection Algorithms for the Task of Person Detection on Jetson TX2 NX Platform," *2024 IEEE Open Conf. of Electrical, Electronic and Inf. Sciences*, 2024.

[20] K. Sarvajcz et al., "AI on the Road: NVIDIA Jetson Nano-Powered Computer Vision-Based System for Real-Time Pedestrian and Priority Sign Detection," *Applied Sciences*, 2024, MDPI.