POLITECNICO DI TORINO
Repository ISTITUZIONALE

Comparative Survey of Embedded System Implementations of Convolutional Neural Networks in Autonomous Cars Applications

Original
Comparative Survey of Embedded System Implementations of Convolutional Neural Networks in Autonomous Cars Applications / Cheshfar, M., Hossein Maghami, M., Amiri, P., Gharaee Garakani, H., Lavagno, L.. - In: IEEE ACCESS. - ISSN 2169-3536. - 12:(2024), pp. 182410-182437. [10.1109/ACCESS.2024.3510677]

Availability:
This version is available at: 11583/2995313 since: 2024-12-13T09:41:19Z

Publisher:
IEEE

Published
DOI:10.1109/ACCESS.2024.3510677

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

17 November 2026

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY SECTION

Received 26 October 2024, accepted 30 November 2024, date of publication 4 December 2024, date of current version 12 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3510677

RESEARCH ARTICLE

Comparative Survey of Embedded System Implementations of Convolutional Neural Networks in Autonomous Cars Applications

MOHAMMAD CHESHFAR[1,2], MOHAMMAD HOSSEIN MAGHAMI[2], PARVIZ AMIRI[2], HOSSEIN GHARAEE GARAKANI[3], AND LUCIANO LAVAGNO[1], (Life Senior Member, IEEE)

[1]Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy
[2]Faculty of Electrical Engineering, Shahid Rajaee Teacher Training University, Tehran 16785-163, Iran
[3]Department of Information Technology, ICT Research Institute, Tehran 1439955471, Iran

Corresponding authors: Luciano Lavagno (luciano.lavagno@polito.it) and Mohammad Hossein Maghami (mhmaghami@sru.ac.ir)

This work was supported in part by the SPACE Project–funded by European Union–Next Generation EU within the PRIN 2022 program (D.D. 104 - 02/02/2022 Ministero dell'Università e della Ricerca); in part by the Key Digital Technologies Joint Undertaking under the REBECCA Project under Grant 101097224; in part by the European Union, Greece, Germany, Netherlands, Spain, Italy, Sweden, Turkey, Lithuania, and Switzerland; and in part by the Italian ICSC National Research Centre for High-Performance Computing, Big Data and Quantum Computing in the context of the NextGenerationEU Program.

ABSTRACT In the rapidly evolving field of autonomous cars, advanced deep learning systems have ushered in a new era of innovation, enabling the integration of unique features into vehicles. These advancements span various areas, including pedestrian and vehicle detection, recognition of road signs and driving patterns, identification of drivable roads and scenes, and improved mapping and routing techniques. However, the high computational requirements of deep learning networks present a significant challenge, especially for embedded systems like FPGAs (Field-Programmable Gate Arrays) that have limited capacity. Addressing this challenge, this article presents a comprehensive survey of the methodologies employed in implementing Convolutional Neural Networks (CNNs) on resource-constrained processors, within the domain of self-driving car applications. Our survey encompasses a thorough review of the existing literature in the field of deep learning applied to autonomous cars, from perception to localization, with a specific emphasis on implementations utilizing embedded hardware such as FPGAs and embedded GPUs. Furthermore, we present and analyze results that elucidate the intricate trade-offs between latency, energy consumption, and the judicious selection of the underlying platform. These insights are crucial for researchers and practitioners in the field, as they provide a clear direction for optimizing the performance of deep learning networks on resource-constrained platforms, ultimately contributing to the advancement of self-driving car technologies.

INDEX TERMS Hardware accelerator, FPGA, autonomous car, convolutional neural network.

I. INTRODUCTION

Self-driving vehicles have the potential to catalyze a paradigm shift in our transportation systems, particularly concerning to safety and efficiency. The escalating number of vehicles on the road has imposed mounting pressure to address critical issues such as traffic congestion, pollution, and road safety. In the research community, autonomous vehicles are recognized as a viable solution to these problems [1], [2], [3]. For instance, the World Health Organization has reported that a staggering 3.1 million people lose their lives in road accidents each year [4]. As human errors account for over 90 % of all driving accidents [5], autonomous vehicles can significantly enhance safety by eliminating human errors. Furthermore, autonomous vehicles offer a plethora of advantages such as superior fuel efficiency, reduced pollution, increased productivity, and optimized traffic flow [6], [7], [8].

Over the past decade, several research teams have been involved in the development of functioning autonomous cars. Based on a search within Elsevier's Scopus database, Fig. 1 shows the exponential growth of scientific and technical documents published on autonomous cars over the past decades. The contribution of pioneer countries in the field of

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Gao.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

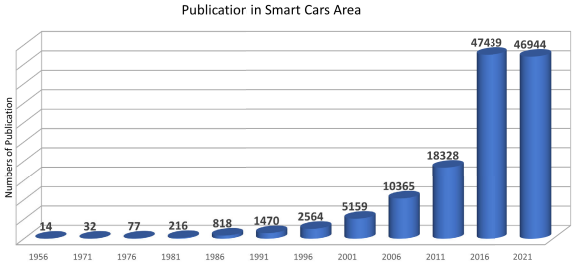


FIGURE 1. Number of scientific and technical documents published with the keywords "smart or autonomous" and "vehicles or cars" in their titles or abstracts.

visual prostheses, in terms of papers published during 2020–2024 is illustrated in Fig. 2. However, a critical aspect that requires further exploration is the implementation of CNNs in these vehicles, specifically in terms of energy efficiency, throughput, and performance [9], [10], [11], [12], [13], [14], [15], [16], [17]. Current commercial computing platforms, such as CPUs and GPUs, while offering high performance, exhibit considerable power consumption, making them less suitable for resource-limited applications [18]. In contrast, Field-Programmable Gate Arrays (FPGAs) have emerged as a viable low-power platform for autonomous vehicle task processing, offering in-field programmability and reconfiguration without system redesign [19].

While deep learning and CNNs have significantly advanced the field of autonomous vehicles, several key challenges still prevent their widespread deployment. CNNs, which play a critical role in tasks like object detection and scene understanding, require substantial computational resources, especially when handling the high volumes of real-time data generated by vehicle sensors. This makes real-time processing challenging, particularly in embedded systems with limited computational power. Additionally, CNNs struggle with generalizing to new environments, such as those with variable lighting, weather, or unexpected road scenarios, which reduces their reliability in unpredictable real-world conditions. Addressing these challenges is essential for the broader adoption of autonomous vehicles on public roads.

However, despite the significant advancements in this field, there exists a gap in the literature regarding a comprehensive comparison of FPGA and embedded system implementations of CNNs in autonomous vehicle applications. This paper aims to fill this gap by providing a detailed comparative survey of various hardware implementations of CNNs. We will analyze their strengths and weaknesses, focusing on energy efficiency, throughput, and performance, to identify the best model for each application.

This study is of considerable importance as it seeks to guide future research and development efforts in the field of autonomous vehicles, contributing to the development of more efficient and effective systems. The major contributions of this paper include:

- *A thorough analysis of the current state of the art* in hardware implementations of CNNs for autonomous vehicles.
- *Identification of key gaps* in the existing literature, particularly in the context of energy-efficient and high-performance solutions.
- *Recommendations for future research directions,* aimed at advancing the field and addressing the identified gaps.

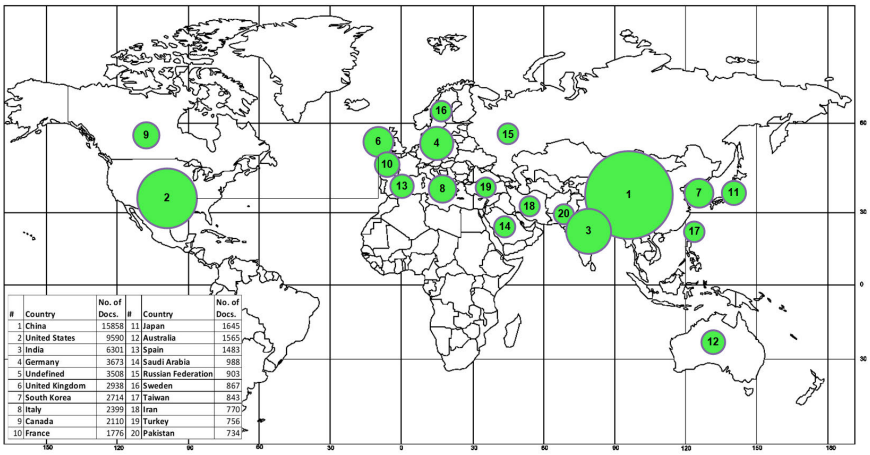
The structure of this paper is organized into twelve sections to provide a comprehensive analysis of the role of CNNs in autonomous vehicles. The Introduction outlines the motivations and objectives of the study, followed by a survey of Related Work that highlights the advancements in the field. In the Methodology, the approach used in examining CNN applications is detailed. The paper then delves into the Functional Components and Paradigms in Autonomous Vehicles, followed by a review of various Computing Platforms. Subsequent sections focus on specific Application Areas of CNNs, such as Vehicle and Pedestrian Detection, Status, Behavior, and Identification of the Driver, Recognition of Traffic Signs, Road Detection and Scene Perception, and Localization, which are all critical for achieving full vehicle autonomy. The final section, Discussion and Conclusion, presents a discussion of the findings, explores Future Directions for the field, and offers concluding remarks on the overall impact of CNNs on autonomous vehicle technologies. This structured approach ensures a logical progression of topics, aiding the reader in understanding the research comprehensively.

II. RELATED WORK

Historically, scholarly inquiries within the automotive sector have manifested in reviews and surveys that delve into specific facets and challenges. Extensive literature has explored subjects such as self-driving cars [20], crowd information, and traffic management [21], as well as intra-vehicular communications [22]. Additionally, the literature encompasses analyses specifically focused on Advanced Driver Assistance Systems (ADAS), elucidated by citations [22], [23], [24], within the framework of autonomous vehicle technology. However, these reviews typically concentrate on aspects like the sensing technology deployed for specific tasks or the evolving perspectives within the ADAS sector and do not delve into the specifics of hardware implementations of CNNs in autonomous vehicles.

In contrast, the work presented in the paper [25] shifts the focus towards a systematic review, systematically analyzing the landscape of published research that integrates Machine Learning (ML) models into resource-constrained implementations for ADAS applications. However, it does not provide a comparative analysis of FPGA and embedded system implementations of CNNs in autonomous vehicle applications.

While numerous reviews have explored methodologies in machine learning ([26], [27]), the convergence of machine



#	Country	No. of Docs.	#	Country	No. of Docs.
1	China	15858	11	Japan	1645
2	United States	9590	12	Australia	1565
3	India	6301	13	Spain	1483
4	Germany	3673	14	Saudi Arabia	988
5	Undefined	3508	15	Russian Federation	903
6	United Kingdom	2938	16	Sweden	867
7	South Korea	2714	17	Taiwan	843
8	Italy	2399	18	Iran	770
9	Canada	2110	19	Turkey	756
10	France	1776	20	Pakistan	734

FIGURE 2. Contribution of the countries actively involved in research on autonomous vehicles in terms of the number of papers they published during 2020–2024.

learning and autonomous vehicle systems, though explored in a few studies [28], [29], [30], remains a rapidly evolving field. There is a significant potential for further exploration, particularly through comprehensive reviews and research. In this particular review [28], the authors undertake a comparative analysis of various methodologies and shed light on the standard architecture of data pipelines in the context of applying machine learning models to ADAS tasks. Paper [29] has made significant strides in investigating the application of AI in the development of autonomous vehicles (AVs), with a particular focus on perception, localization, mapping, and decision-making. These studies have been instrumental in understanding how AI can be utilized and the challenges associated with its implementation in AV systems. The article [29] offers an extensive review of the advancements in intelligent vehicle systems, emphasizing the role of AI in addressing traffic issues and transforming the automotive industry. It covers various aspects including object detection, driver assistance, and autonomous driving, and their implications on traffic congestion, accidents, and environmental pollution. However, these works do not provide a comprehensive comparison of various CNN applications relevant to self-driving cars that can be effectively deployed on resource-constrained hardware, including FPGA and GPU-embedded systems. This paper aims to fill this gap.

Additionally, attention has been directed towards hardware implementations of CNN on resource-constrained platforms like FPGAs, embedded CPUs, and GPUs for diverse applications. Notably, the paper [19] offers a comprehensive overview of works related to FPGA-based robotic accelerators, spanning various stages of the robotic system pipeline. The historical development and application domains of representative neural networks are explored in this paper [31], emphasizing the importance of studying deep learning technology and the advantages of leveraging FPGAs for acceleration. Another survey, [32], scrutinizes computer vision FPGA-based endeavors in the automotive domain over the past decade. Furthermore, [33] contributes a survey elucidating the role and significance of FPGAs in automotive systems, with a specific emphasis on showcasing the opportunities and challenges that FPGAs present in the context of next-generation automotive systems.

In this study, performance refers to the metrics associated with machine learning tasks, such as accuracy and precision. The performance of hardware/software implementations is characterized in terms of throughput and energy efficiency. This distinction allows for a comprehensive evaluation of the different aspects of performance in machine learning applications. Additionally, algorithm modification encompasses the techniques used to implement CNNs on embedded systems, including various model compression techniques, architectures, frameworks, and optimizations for efficient deployment on resource-constrained platforms. Despite the significant advancements, a comprehensive comparison of these performance metrics across different hardware implementations, particularly in the context of autonomous vehicles, is lacking in the current literature.

Emerging machine learning solutions primarily focus on three strategies: (1) adapting models to suit resource-limited environments; (2) implementing or accelerating hardware processes and results; and (3) leveraging more robust hardware. In the context of model adaptation, three main trends are observed: modifying operations [34], adjusting numerical computations and data types [35], and tailoring models to fit environments with limited resources [36]. The strategy of hardware implementation involves mapping neural networks onto programmable processors or bespoke hardware such as FPGAs or neuromorphic processors. This is often coupled with explicit memory mappings for efficient resource management [37]. Lastly, the strategy of utilizing more powerful hardware is rooted in the recent advancements of embedded platforms with dedicated GPUs. These are particularly beneficial for automotive tasks as they accelerate the inference of deep learning models. A prime example

of this is the adoption of the NVIDIA Jetson and Tegra families in the embedded and automotive sectors. While these strategies have shown promise, a detailed comparative analysis of their effectiveness in the context of autonomous vehicles is still needed.

CNNs have emerged as indispensable elements driving the progress of autonomous vehicles, assuming a pivotal role across various domains of perception and decision-making. Their unparalleled capacity to discern intricate patterns and features directly from unprocessed sensor data has ushered in a new era in autonomous driving research. CNNs play a vital role in all five core functional components of self-driving vehicles, showcasing exceptional performance in tasks such as object detection, lane tracking, semantic segmentation, environmental perception, and decision-making. This capability not only enhances the safety and efficiency of autonomous vehicles but also underscores the indispensability of CNNs in their operation.

Furthermore, CNN-based methodologies have exhibited remarkable efficacy in sensor fusion tasks, seamlessly integrating information from diverse sensor modalities such as cameras, LiDAR, and radar to construct a holistic understanding of the driving environment. Additionally, CNNs have demonstrated prowess in end-to-end learning paradigms, wherein the entire perception-action pipeline is encapsulated within a unified neural network architecture, streamlining system design and mitigating computational complexity. Recognizing the transformative impact of CNNs on autonomous vehicle capabilities, this paper exclusively centers on CNN-based methodologies. However, a comprehensive survey that compares various CNN applications relevant to self-driving cars that can be effectively deployed on resource-constrained hardware, including FPGA and GPU embedded systems, is currently missing in the literature. This paper aims to fill this gap.

Despite the significant contributions in the field, there is a lack of comprehensive literature that explores the implementation of deep learning, particularly CNNs, on resource-constrained platforms tailored for autonomous car applications. This gap is what this survey paper aims to fill. We provide a comprehensive survey of various CNN applications relevant to autonomous cars that can be effectively deployed on resource-constrained hardware, including FPGA and GPU-embedded systems. The widespread adoption of FPGA and GPU embedded platforms within the autonomous car industry is driven by their energy efficiency and adept handling of advanced computational tasks.

However, these platforms have their strengths and weaknesses, and a detailed comparative analysis of their effectiveness in the context of autonomous vehicles is still needed. This paper contributes a focused survey aimed at consolidating and analyzing the existing knowledge in this domain, thereby contributing to the discourse on optimizing deep learning applications for resource-constrained environments in the context of autonomous car technology. Our work will highlight the unaddressed areas and potential avenues for future research, thereby providing a roadmap for researchers and practitioners in this rapidly evolving field.

III. METHODOLOGY

The implementation of Convolutional Neural Networks (CNNs) in autonomous cars necessitates a careful balance between energy efficiency, latency, and throughput, particularly on resource-limited platforms such as FPGAs and embedded systems. Current literature offers numerous studies on CNN implementations, but a comprehensive comparative survey focusing on these specific performance metrics within the context of autonomous vehicles remains scarce. This review aims to fill this gap by systematically evaluating and comparing the energy efficiency, latency, and throughput of CNN implementations on FPGAs and embedded systems, targeting applications in autonomous cars. The inclusion and exclusion criteria for this survey are meticulously designed to ensure that only relevant studies that contribute to this comparative analysis are considered, thereby providing valuable insights for researchers and practitioners aiming to optimize CNN deployments on resource-constrained hardware for autonomous driving.

A. ELIGIBILITY CRITERIA

1) INCLUSION CRITERIA

The inclusion criteria for this review are meticulously crafted to ensure that only the most relevant and high-quality studies are considered. The criteria are designed to address the specific gaps in the literature related to the implementation of Convolutional Neural Networks (CNNs) on embedded or resource-constrained platforms for autonomous cars. The primary prerequisites are as follows:

1) **CNN Model Inclusion**: The article must explicitly incorporate a CNN model. This criterion ensures that the review focuses on studies that leverage deep learning techniques, specifically CNNs, which are pivotal in processing and interpreting the vast amount of data in autonomous driving.
2) **Embedded or Resource-Constrained Platforms**: The study must involve the deployment of the CNN model on an embedded or resource-constrained platform, such as FPGAs or embedded systems. This requirement is crucial to address the unique challenges and opportunities of implementing deep learning algorithms in environments with limited computational resources, a common scenario in autonomous vehicles.
3) **Relevance to Self-Driving Cars**: The article must address tasks pertinent to autonomous vehicles, particularly in the domains of perception and localization. These tasks are critical for the safe and efficient operation of self-driving cars.

    - *Model Specifications*: The CNN model must be well-defined, detailing its architecture and the dataset used for training. Studies should provide

sufficient information on how the model is trained and evaluated.

- *Execution and Adaptation*: Flexibility in model execution is acknowledged, allowing for studies that modify or optimize CNN models to fit embedded platforms. Even if the deep learning model is not altered, its deployment on a resource-constrained platform is essential.
- *Performance Evaluation*: The article must include a comprehensive assessment of the model's performance using specific metrics. This ensures that the implementations are not only theoretical but have been tested and validated in practical scenarios.

Studies proposing modifications to CNN models or hardware, even if not directly applied to self-driving car tasks, are included if they demonstrate potential relevance and utility in the broader context of deep learning on resource-constrained platforms.

2) EXCLUSION CRITERIA

To maintain the focus and relevance of the review, certain criteria have been established to exclude studies that do not meet the requirements:

1) **Lack of Model Training Details**: Studies that do not provide details on the training or learning processes of the CNN model are excluded. This ensures that only robust and replicable studies are included.
2) **Non-Embedded Implementations**: Articles that conduct the overall implementation on personal computers or general-purpose machines are excluded. The focus is on resource-constrained platforms, and studies on general-purpose hardware do not align with this objective.
3) **Absence of Performance Testing**: Studies that do not include a performance test along with the corresponding metrics are excluded. Performance evaluation is critical to understanding the practical applicability and efficiency of the CNN models.
4) **Irrelevant Subject Matter**: Studies focused on domains outside the scope of autonomous driving, such as information management, vehicular networks, traffic surveillance, theft avoidance systems, Battery State-of-Charge, and State-of-Health, are excluded.
5) **General Tasks Without Specificity**: Studies addressing general tasks like perception or localization without explicit dedication to autonomous cars are excluded unless they introduce a significant improvement applicable to the field.
6) **Basic Machine Learning Tasks**: Papers focusing on rudimentary machine learning tasks for lightweight applications are excluded. The emphasis is on advanced CNN implementations for complex autonomous driving tasks.
7) **Access to Full Text**: Only studies with accessible full texts are included. This ensures a thorough review and understanding of the methodologies and findings presented.

There are no exclusion criteria concerning the format or date of publication, allowing the inclusion of a broad range of relevant studies.

B. SEARCH STRATEGY

The first step in the search process entails the careful selection of databases. Given the study's specific focus on computing and electronics, databases pertinent to these domains are discerningly selected. The designated databases employed for this investigation are Scopus, Springer, and IEEE Xplore. These databases were chosen for their extensive coverage of the field of computer science and engineering.

Scopus is known for its broad interdisciplinary coverage, including numerous high-impact journals and conference proceedings in the fields of computing and electronics. **IEEE Xplore**, on the other hand, is a premier database for electrical engineering and computer science, providing access to a vast collection of technical literature and cutting-edge research. **Springer** is included due to its comprehensive collection of scientific publications, particularly in the domain of engineering and applied sciences, ensuring that a diverse range of relevant studies is captured.

Including Scopus, Springer, and IEEE Xplore ensures a comprehensive search that captures a wide range of relevant studies. Scopus provides an extensive index of peer-reviewed literature, IEEE Xplore offers specialized access to key publications in engineering and technology, and Springer includes a broad array of research in engineering and applied sciences. This combination mitigates the risk of missing pertinent studies due to database-specific coverage limitations.

In our search endeavors, we utilized a carefully curated collection of keywords from the following categories across all search platforms: autonomous vehicle OR smart autonomous OR autonomous car OR smart car AND CNN OR convolutional neural network AND FPGA OR embedded platform OR SoC. The searches were systematically conducted encompassing title, abstract, and keywords collectively. In cases where this combination was not feasible, the search was primarily based on the title alone to mitigate noise in the search results. Boolean operators AND and OR were judiciously employed to refine search groupings, ensuring comprehensive yet focused search results.

The search activities transpired between March 2023 and January 2024. Consequently, articles published after February 1, 2024, fall beyond the defined scope of this investigation. The chosen time frame ensures that the review includes the most recent and relevant studies in the rapidly evolving field of CNN implementations for autonomous cars.

We also utilized the reference lists and the "Cited by" function from Google Scholar search results to find additional relevant works. This approach allowed us to capture a wider range of relevant articles, including those that may not

specifically mention CNNs but are still pertinent to our review. By examining cited references and works citing key studies, we ensured that our review encompasses a broad spectrum of influential and related research.

Our initial search yielded 780 titles. After removing duplicates and irrelevant studies, and applying the inclusion criteria, 60 eligible titles were identified for further review. Of these, 10 titles were extracted from review papers, providing a well-rounded collection of primary studies and comprehensive reviews relevant to our research focus.

This systematic approach to database selection and keyword usage, along with the inclusion of cited references, ensures a thorough literature review. The careful delineation of our search strategy and the rationale behind each decision contributes to the robustness and reliability of our findings.

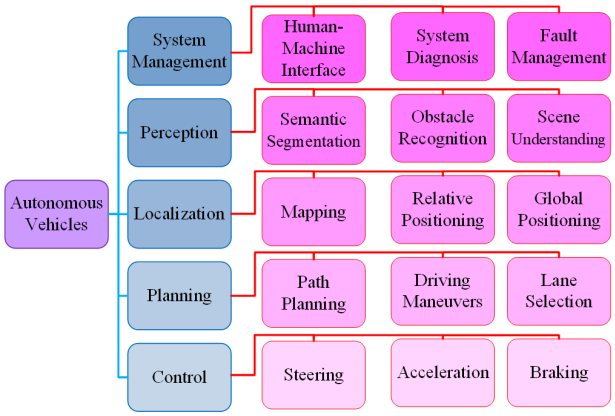


FIGURE 3. Autonomous vehicle functional units.

IV. FUNCTIONAL COMPONENTS AND PARADIGMS IN AUTONOMOUS VEHICLES

Autonomous vehicles consist of five fundamental functional components shown in Fig. 3: perception, localization, planning, control, and system management [18]. The perception component observes the vehicle's surroundings, detecting crucial entities such as traffic signs, other vehicles, and obstacles. On the other hand, the localization module maps the vehicle's environment and determines its absolute position. The planning element employs inputs from the perception and localization modules to execute high-level actions that determine the vehicle's intended path, lane changes, and speed. The control module, meanwhile, oversees the execution of low-level actions commanded by the planning module, such as steering, acceleration, and braking of the vehicle. Finally, the system management module supervises the performance of all other components and facilitates the interface between human operators and the automated system [9].

Autonomous vehicles are increasingly using a perception-planning-control paradigm, where distinct functions are performed independently. The perception function relies on obtaining and processing sensor data, achieved through multi-sensor systems and sensor fusion to achieve precise environmental registration. Planning and control are provided by rule-based systems, often employing classic controllers based on linear vehicle models. However, this approach requires manual calibration of control parameters based on simulation and field testing, which is a time-consuming process [9]. As a result, recent limitations observed in autonomous vehicle systems—particularly those related to the computational overhead and complexity of integrating multi-sensor data—have inspired researchers to investigate alternative solutions for more efficient, reliable, and scalable autonomous systems.

The rapid development of deep learning technologies has revolutionized various fields of artificial intelligence (AI), significantly enhancing performance in areas such as image classification and speech recognition, and sensor data processing [10], [11], [38], [39]. The high approximation ability of deep neural networks (DNNs) has led to the adoption of deep learning technology in a wide range of self-driving applications, such as decision-making and planning [12], [13], vehicle-to-vehicle communication [14], perception [15], and mapping and localization [16], [17]. This shift towards deep learning models has the potential to overcome some of the limitations associated with traditional rule-based systems, especially in terms of dynamic environmental changes and complex decision-making.

Sensor-based systems in AVs face significant limitations. Cameras struggle in low light and adverse weather, while LiDAR is expensive and limited in range under poor conditions. Radar, though weather-resistant, offers low resolution. While sensor fusion addresses some challenges, overall performance still relies on data quality. Research into advanced path-planning algorithms, such as BRRT*-DWA combined with AMCL, has improved AV navigation in dynamic environments by enhancing localization and decision-making [39], [40]. Future solutions may include integrating neural network-based adaptive control to further enhance sensor data interpretation and decision precision, promoting wider AV deployment [41]. Therefore, careful consideration is needed when selecting computing platforms for autonomous vehicles.

V. COMPUTING PLATFORMS FOR AUTONOMOUS VEHICLES

The major commercial computing platforms in use today for autonomous vehicles consist of central processing units (CPUs), graphics processing units (GPUs), FPGAs, and application-specific integrated circuits (ASICs). The features of several common platforms used for performance computations in self-driving cars are presented in Table 1.

In autonomous vehicle applications, CPUs play a crucial role in performing a variety of tasks, including control logic and algorithm development, due to their ability to handle general-purpose processing. A typical CPU can deliver between 10-100 GFLOPS, although with lower power efficiency than other platforms, often less than 1 GOP/J

[42]. While CPUs are known for their flexibility, they face structural limitations when it comes to deep learning and AI tasks, particularly because of their Von Neumann architecture [31]. In AI applications, CPUs spend a significant amount of time reading and processing data or instructions, and they struggle with the high computational demands these tasks impose. Despite having larger cache sizes and higher on-chip bandwidth than GPUs and reconfigurable architectures, CPUs are limited in their ability to handle parallel processing, which makes them inefficient for the large-scale data processing required in autonomous systems [43].

GPUs play a critical role in autonomous vehicle development due to their exceptional parallel computing capabilities. With thousands of processing cores, modern GPUs, such as the Nvidia Ampere A100 [44] and Tesla V100 [45], can deliver up to 20 TOPS, making them highly efficient for tasks like image processing and object detection in real time. Their ability to handle deep learning workloads efficiently has made them indispensable for training and running complex neural networks. Despite their high performance, GPUs come with a significant power cost, often consuming between 10W to over 300W, which can exceed practical limits for resource-constrained applications like edge devices [19]. However, embedded GPU platforms, such as the NVIDIA Jetson and Tegra families [46], [47], [48], [49], have been specifically designed for autonomous vehicles, offering a balance between computational power and energy efficiency for real-time edge processing. The extensive software support for GPUs, including frameworks like CUDA and OpenCL, has further solidified their role as a dominant platform for deep learning acceleration in autonomous driving systems [43].

FPGAs have gained significant attention as a platform for achieving low-power, high-performance processing in autonomous vehicles. FPGAs consume less power than GPUs and are often integrated into smaller systems with constrained memory resources. They can perform massively parallel computations and are particularly effective in applications requiring real-time processing, such as perception (e.g., stereo matching), localization (e.g., SLAM), and planning (e.g., graph search). Unlike GPUs, FPGAs offer reconfigurability, allowing designers to tailor hardware architectures to specific tasks, leading to optimized performance. Recent advancements, such as the Xilinx Virtex UltraScale+ [50], AMD Versal VC1902 [51], and Intel Stratix 10 [52], have pushed the boundaries of FPGA capabilities, offering up to 200 MB of on-chip memory (including BRAM, Ultra RAM, and distributed RAM) and delivering performance metrics competitive with high-end GPUs while maintaining superior energy efficiency [19], [50], [52].

FPGAs are particularly well-suited for applications where energy efficiency is paramount and where workloads benefit from customizable hardware. For instance, they are frequently employed in applications where tasks can be offloaded from the CPU to an FPGA to achieve lower latency and power consumption, as seen in many current autonomous vehicle platforms. However, the storage capacity of FPGA on-chip memory is still limited compared to the memory requirements of modern deep learning models, which often range from 100 MB to several GB. This necessitates the use of external memory, such as DDR SDRAM or HBM (High Bandwidth Memory), which can introduce performance bottlenecks due to limited memory bandwidth and increased power consumption.

ASICs represent another computing platform, offering unparalleled performance and energy efficiency for specific tasks due to their highly specialized nature. ASICs are custom-designed chips optimized for particular operations, such as those required in neural network inference engines. While they lack the flexibility of FPGAs or the general-purpose capabilities of CPUs and GPUs, ASICs can provide superior performance for dedicated tasks. Examples include Google's TPU (Tensor Processing Unit), designed specifically for accelerating machine learning workloads, and Tesla's custom-designed FSD (Full Self-Driving) chip, which is optimized for the needs of autonomous driving. However, ASICs require significant upfront design costs and time, making them less flexible for applications where algorithms are still evolving. Table 2 presents a detailed comparison of the computational performance, power consumption, and other relevant attributes of FPGAs, GPUs, CPUs, and ASICs, highlighting their strengths and limitations in autonomous vehicle applications.

The typical architecture for FPGA-based neural network accelerators consists of a CPU host and an FPGA component, as shown in Fig. 4(a). This configuration may involve dedicated FPGA chips that interact with a host PC or server via PCIe connections, or may be integrated within the same chip or package in SoC platforms such as the Xilinx Zynq series and Intel HARPv2 [42]. In the first case, the host and the FPGA have separate off-chip memory spaces, and memory access between the two is facilitated by various techniques. Direct memory access (DMA) allows the FPGA to read from or write to the CPU's memory directly without CPU intervention, greatly improving data transfer speeds. Shared virtual memory (SVM) provides a unified virtual address space accessible to both the CPU and the FPGA, simplifying memory management and enabling seamless data sharing. Cache coherency protocols ensure both components maintain a consistent view of shared memory, reducing the need for explicit synchronization. Dual-port RAM allows simultaneous access from both the CPU and FPGA, facilitating efficient, contention-free data exchange. Memory-mapped interfaces allow the CPU to access FPGA resources as if they were memory locations, enabling easy communication and control. In such systems, the CPU typically handles control tasks and nonlinear computations, while the FPGA is used for parallelized, intensive processing tasks. On the other hand, SoC-FPGA systems provide a single off-chip memory space for both CPU and FPGA logic, which potentially eases communication and may reduce communication bottlenecks between CPU and

TABLE 1. Features of some of the platforms.

Platform	Xavier [46]	TX2 [47]	TX1 [49]	TK1 [53]	ZYNQ US+ [54]	Altra Stratix V [55]
Top Computational Performance (GFLOPS)	1400	1330	1024	325	800	1000
Top Off-Chip Memory Bandwidth (GB/S)	59.7	51.2	25.6	17	19.2	76.8
Top On-Chip Memory Bandwidth (GB/S)	59.7	51.2	25.6	17	80	224
Energy per Operation (pJ/Op)	14.29	11.28	14.65	15.38	12.5	1

TABLE 2. Comparison of hardware accelerators for deep learning, the data presented in this table has been interpreted and synthesized from the following references: [19], [32], [42], [43].

Platforms	Vector Processors (VPs)	Graphics Processing Units (GPUs)	Field-Programmable Gate Arrays (FPGAs)	Application-Specific Integrated Circuits (ASICs)
Energy Efficiency (GFLOPS/W)	10-40	40-80	15-80	50-200
Raw Computation Power (TFLOPS)	1-2	5-20	1-10	4-30
Memory Bandwidth (GB/s)	50-350	127-900	15-400	600-1200
Flexibility	Highly flexible	Versatile	Highly flexible	Specialized for specific tasks
Adaptability	Good	Good	Highly adaptable	Fixed functionality
Parallelism	SIMD	SIMD and SIMT	Customizable parallelism	Fixed parallelism (customized)
Programming Model	SIMD or vectorized	CUDA, OpenCL	HDL, HLS, DSL, MBD, graphical, entry, IP cores	HDL, HLS, DSL, MBD, UVM,scripting
Application Range	General-purpose	General-purpose	Versatile, customizable	Specific, optimized for workloads
Programming Ease	Relatively easy	Relatively easy	Moderate (HLS tools)	Complex (hardware synthesis)

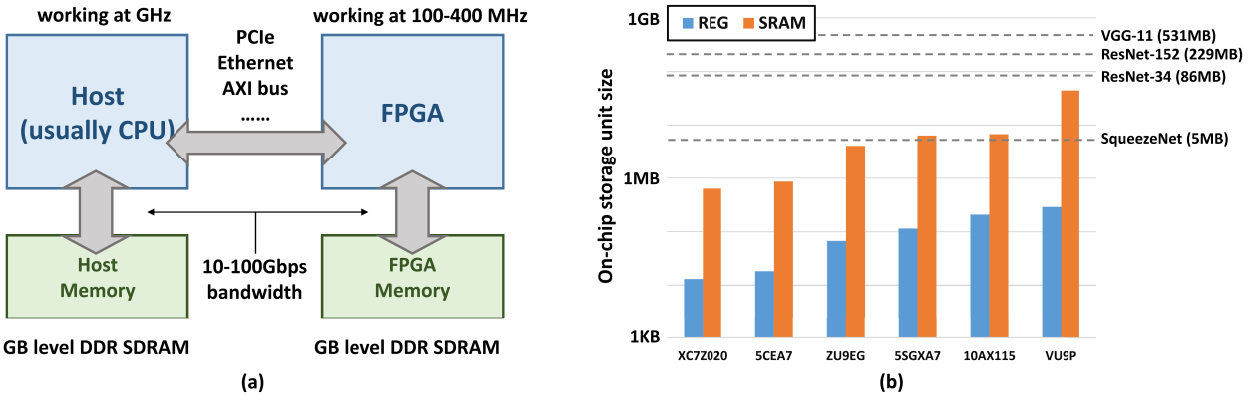


FIGURE 4. (a) A typical structure of an FPGA-based NN accelerator. (b) The gap between NN model size and the storage unit size on FPGAs [42].

FPGA. Therefore, the choice between an SoC and a standard FPGA depends on the specific application requirements, with factors such as data transfer rates, latency, and integration complexity influencing the decision.

Fig. 4(b) illustrates the significant gap between the on-chip memory capacity of FPGA devices and the memory requirements of typical neural network models. While modern FPGA chips can provide up to 200 MB of combined on-chip memory (including BRAM, Ultra RAM, and distributed RAM), this falls short of the memory requirements of most neural networks, which often range from hundreds of MB to several GB. As a result, external memory solutions such as DDR SDRAM or high bandwidth memory (HBM) are essential to store all the required parameters of large neural networks. However, the use of external memory often becomes a bottleneck, limiting system performance due to bandwidth constraints and higher energy consumption.

In conclusion, while GPUs remain a popular choice for high-performance computing in autonomous vehicles due to their raw computational power, FPGAs offer a compelling alternative for applications where power efficiency and customizable processing are critical. The choice of computing platform ultimately depends on the specific needs of the application, whether it be flexibility, performance, power efficiency, or ease of deployment.

VI. APPLICATION AREAS OF CNNs IN AUTONOMOUS VEHICLES

In recent times, the profound impact of DNNs also called deep learning, has been extensively felt in computer vision, leading to remarkable advancements in robot perception. Today, many advanced algorithms rely on a type of neural network based on convolution operations to effectively deliver their objectives. Among the algorithms that have emerged to achieve precise object detection with improved

speed and accuracy are Fast R-CNN [56], Faster R-CNN [57], SSD [58], YOLO [59], and YOLO9000 [60].

The application areas of CNNs in autonomous vehicles are diverse and can be categorized to address specific requirements and operational contexts:

- **Vehicle and Pedestrian Detection**: This application area is critical for identifying and reacting to dynamic entities such as other vehicles and pedestrians in the vehicle's environment. It requires high accuracy and real-time processing capabilities to ensure timely and correct responses to avoid accidents and enhance safety.
- **Driver Status, Behavior, and Identification**: Monitoring the driver's state is essential for enhancing overall safety. This involves precise behavior analysis and continuous monitoring to detect various driver states such as drowsiness, distraction, or incapacitation, allowing for prompt intervention when necessary.
- **Recognition of Traffic Signs**: Autonomous vehicles must comply with road regulations, which involve the quick and accurate interpretation of various traffic signs. This capability is crucial for making informed driving decisions and ensuring the vehicle adheres to traffic laws.
- **Road Detection and Scene Perception**: Understanding the driving environment is vital for effective path planning and obstacle avoidance. This requires robust segmentation techniques that can handle diverse and varying road conditions, such as different types of terrain, weather conditions, and the presence of static and dynamic obstacles.
- **Localization**: Accurate navigation and map updates are fundamental to the operation of autonomous vehicles. Localization integrates data from multiple sensors to determine the vehicle's precise position, which is crucial for effective navigation, route planning, and ensuring the vehicle remains on the correct path.

This categorization highlights the distinct sensor types, hardware implementations, and unique requirements of each application area. By clearly defining these categories, we ensure a comprehensive and targeted approach to leveraging CNNs in autonomous vehicles. This structure provides a clear framework for understanding the diverse applications and emphasizes the specific needs and challenges inherent to each area.

The majority of current semantic segmentation work, which is based on CNN, is focused on fully convolutional networks (FCN) [61], while some recent strides have also been made through spatial pyramid fusion networks (SPNets) [62] and pyramid scene parsing networks (PSPNet) [63]. These novel CNN architectures adopt a method that couples global image information with locally extracted features for superior results. Furthermore, by using auxiliary natural images, it is possible to train an automatic stackable encoder model offline to learn overall image features before its eventual application in online object tracking [64]. In Table 3, some of the common networks used in self-driving cars and their computational demands are showcased. The table considers an input image size of 224 × 224 pixels for classification, as computational complexity correlates with the size of the input image.

TABLE 3. Resource utilization of common networks in autonomous cars.

Network	Year	# Param M	# Opr GFLOPs	HW
AlexNet [11]	2012	60.0	1.4	GTX 580
VGG19 [65]	2014	144.0	39.0	Titan Black
Faster R-CNN [56]	2015	134.5	0.24	K40
ResNet152 [66]	2016	57.0	22.6	GPU
MobileNet [34]	2017	4.2	1.1	
ShuffleNet [67]	2017	2.36	0.27	ARM
PSMNET [68]	2018	14.76	1.46	Titan Xp
VLocNet++ [69]	2018	102.4		Titan X
YOLO(v5s) [70]	2022	12.6	16.8	GForce RTX

CNN-based stereo vision systems have demonstrated remarkable effectiveness in different computer vision tasks such as image classification, object detection, and semantic segmentation. Additionally, CNN has been utilized recently in stereo estimation [71] and stereo matching [72]. In this regard, several works have deployed CNNs on various FPGA platforms. For instance, YOLOv2 for object detection was implemented using a binary CNN on the Xilinx ZCU102 FPGA platform [36]. This design achieved a detection rate of 40.81 frames per second (FPS), which is 4.177 times faster than the ARM Cortex-A57 and 5.27 times quicker than embedded NVIDIA Pascal GPUs. Another noteworthy example is RealtimeSeg, a novel semantic segmentation model introduced through neural architecture search (NAS) to achieve real-time execution on edge devices [73]. RealtimeSeg employs a multi-objective NAS approach, considering inference time, FLOPs, and accuracy. Accelerated with NVIDIA TensorRT, it achieves a high accuracy of 71.7 mIoU(%) at 195.7 FPS, with 1.5 GFLOPs and a model size of 2.35 million parameters, making it highly efficient. Compared to MoSegNAS [74], which attains 75.9 mIoU(%) at 73.2 FPS with 35 GFLOPs, RealtimeSeg offers lower computational demands and higher accuracy. Similarly, M-FasterSeg [75] achieves 71.5 mIoU(%) at 30 164.3, but requires 26.38 GFLOPs and 3.1 million parameters, resulting in a larger model size. These comparisons illustrate the significant performance improvements achievable with embedded systems implementations, highlighting the potential of edge devices for efficient CNN deployment in autonomous vehicles.

When it comes to autonomous vehicles, one of the most crucial matters is the task of localization and mapping. Simultaneous Localization and Mapping (SLAM) is an advanced algorithm for robot navigation that allows for the creation or updating of a precise map of an unknown environment while concurrently tracking the robot's precise position. Localizing self-driving cars necessitates an accurate map of the area, and constructing or updating a map requires

a precise location indicator or an estimated position from known coordinates.

To summarize, numerous FPGA-based CNN accelerators have been developed, as reported in [42]. In the following sections, different applications of perception in self-driving cars will be discussed, drawing from the aforementioned developments in computer vision for improved functionality.

VII. VEHICLE AND PEDESTRIAN DETECTION

A. STRUCTURE OF VEHICLE AND PEDESTRIAN DETECTION ALGORITHMS

The use of CNNs is a prevalent technique for both feature extraction and classification tasks. However, the higher computational requirements of these networks make them less suitable for resource-limited applications. As a consequence, research efforts have focused on reducing the memory and processing requirements of CNNs, primarily by enhancing their computational efficiency. For instance, a modified iteration of SqueezeNet [76] has been employed in [77], in conjunction with a coarser network layer, to detect pedestrians while mitigating the computational demands of the model. By correlating the size of the network with the size of the feature map, the computational load can be significantly reduced.

An excellent instance of reducing memory size can be seen in [78]. The authors employ the DeepPed network [79], which is built upon the foundation of AlexNet. To decrease memory usage and enhance latency, they implement two optimization techniques for the network: pruning and scalar quantization (which is based on k-means clustering).

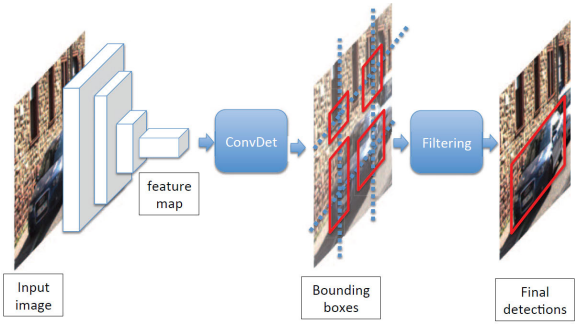


FIGURE 5. SqueezeDet detection pipeline [82].

The quantization is pivotal for obtaining smaller networks and reducing delay. Two distinct methods are outlined for quantization: the first involves the precise training of the network in a cognizant state for quantization, while the second entails transforming operations and values into fixed-point post-training. In the latter approach, certain operations, such as activation functions, are retained in floating-point.

Their investigation into the impact of these strategies extends to both convolutional and fully connected (FC) layers. Remarkably, they note that while pruning has a considerable impact on convolutional layers, the application of quantization does not place a significant strain on their performance. In contrast, concerning fully connected layers, both methods result in comparable performance decline. Nevertheless, the concurrent implementation of these two approaches yields superior results. To gauge the influence of these enhancement strategies on memory consumption, they contrast the size of the original network, which is 216 MB, with the size of the final compacted network, which is 3.5 MB. This results in a compression factor of 61.35x, while only causing a minor increase of 0.4 % in the Log Average Miss Rate.

Presently, research has explored the use of lightweight CNNs for pedestrian and vehicle detection [80], leading to the development of a lightweight Single Shot Detector (SSD) network with a MobilenetV2-based architecture that, while slightly less accurate than other CNNs, is faster. Additionally, this network employs several strategies to preserve accuracy while reducing computational requirements. For instance, the resolution has increased to enhance the detection of pedestrians in distant frames. Furthermore, depth blocks were added at the beginning of SSD to reduce network complexity, while the prediction branch was expanded with two additional heads to consider small and medium-sized objects. Moreover, the network was improved by implementing random sampling of filter layers through network pruning, and two-stage training from scratch was executed without pre-trained architecture to demonstrate feasibility [81]. Finally, tracking through frames was utilized to further improve detection.

Overall, these approaches and techniques aim to optimize the performance of CNNs while reducing their computational requirements and facilitating their use in resource-limited applications.

B. THE METHOD OF EMBEDDING AND PERFORMANCE OF VEHICLE AND PEDESTRIAN DETECTION

As previously stated, implementing neural networks on embedded systems proves challenging due to the real-time constraints, which often require a balance between latency and performance. In the pursuit of addressing this challenge, researchers have developed and optimized different versions of neural networks.

An illustrative instance revolves around a customized iteration of SqueezeNet [77], integrating 1×1 convolutions and deferred down-sampling to mitigate computational demands. Leveraging the automotive processor NXP S32V234 [83], significant strides were made in diminishing memory requirements and enhancing latency by employing an 8-bit integer representation for computations—excluding tanh activations, which operate on floating-point precision. The resultant optimized network attains a peak speed of 30 FPS, all while consuming a mere 2 W.

Various methodologies have been employed to tackle the latency challenges inherent in deep neural networks. Notably, in a study by [78], a comprehensive system consisting of an Aggregate Channel Feature (ACF) detector, an optimization

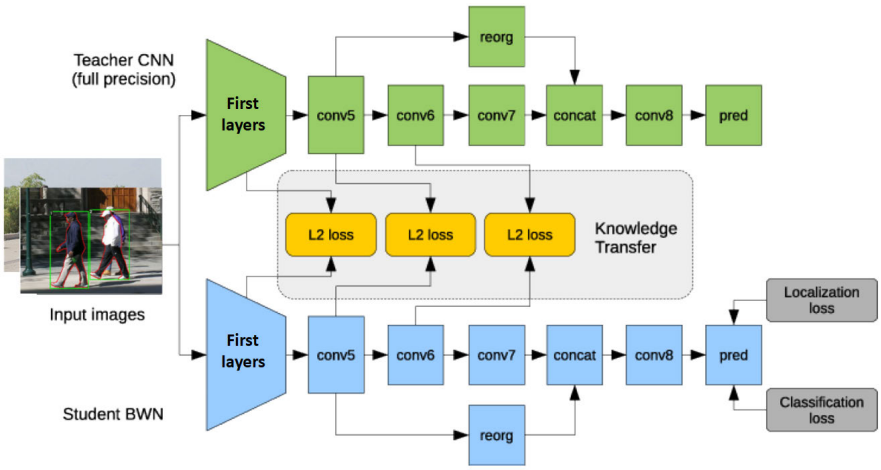


FIGURE 6. Knowledge transfer method for training binary weight YOLO object detector.

network based on AlexNet, and a Support Vector Machine (SVM) was devised. The fully optimized system attains a compression ratio of 61, resulting in a final size of 3.5 MB. Nonetheless, when implemented on an NVIDIA Jetson TX1, the system only manages to reach a frame rate of 2.4 FPS. The authors implemented a strategy of reusing detection results for successive frames, resulting in a peak performance of 10 FPS.

In another endeavor [84], the authors innovatively transformed the detection and classification modules of the primary network into a unified module named ConvDet, as shown in Fig. 5. This fully convolutional detection model significantly reduced the number of parameters—decreasing the network's size to as little as 7.9 MB in its minimal version—while simultaneously enhancing the region proposal accuracy for detected objects. Consequently, this optimized network exhibited performance comparable to or better than that of more resource-intensive counterparts, such as Faster R-CNN [57], which typically requires over 500 MB of storage, and YOLO, which occupies 735 MB [59].

An exemplar of optimized neural networks is delineated in [84], wherein a tailored iteration of a Faster R-CNN was deployed on the NVIDIA Tegra TX1. The researchers strategically applied pruning and quantization with dual objectives: dimensionality reduction and augmentation of computational power. This meticulous approach resulted in a substantial decrease in size, from 260 MB to 42 MB, and a noteworthy latency reduction from 508 ms to 327 ms per frame. This size reduction was achieved through targeted pruning of fully connected layers based on a predetermined threshold and the utilization of Single Instruction Multiple Data (SIMD) architecture commands, which transformed network weights from 32-bit real values to 8-bit integers.

In the pursuit of enhancing operational efficiency, a promising avenue involves scrutinizing and fine-tuning individual components. As expounded in [85], the authors accelerated the speed of an R-CNN by pinpointing layers that exhibited the most time-consuming computations. Their discernment highlighted fully connected layers and batch normalization layers as the most inefficient components. Consequently, they strategically removed batch normalization layers while concurrently reducing the neuron count in fully connected layers unrelated to direct classification. This meticulous intervention resulted in a notable speed increase, elevating the performance from 15 FPS to 25 FPS on a personal computer. However, on the NVIDIA Jetson TX1, the attained speed plateaued at 5 FPS. These results reinforce the observation that CNNs remain computationally demanding, hindering their real-time performance.

Quantizing the network weights is a potential strategy for overcoming the restrictions for embedding CNNs. Binary weight quantization is the most severe form of quantization, transforming floating-point numbers into binary values of either 0 and 1 or -1 and 1. In [86], the authors succeeded in converting two YOLO CNNs based on Darknet and MobileNet architectures into networks with binary weights. To mitigate negative consequences such as accuracy loss, they employed two strategies: binary axis and learning by imitation, shown in Fig. 6. In the binary axis phase, the binary layers of the network were trained group-wise, starting first with the upper layers and then moving to the lower layers on the network chart.

To reduce accuracy loss, learning by imitation mimicked the binary-weighted network's features in specific layers from a teacher network with floating-point numbers. During the network's training period, a new loss expression, based on the difference between the output of a layer in the teacher and the student networks, facilitated the calculation of this imitation. This strategy can be extended to a more general case where the teacher and the student have different architectures but with only some layers in common. In such a case, the student can be partially initialized from the corresponding layers of the teacher. This suggests the potential for broader applicability and flexibility of

their proposed method in various network architectures. The authors were successful in converting Darknet-YOLO from 257 MB to 8.8 MB, with only a 2 % accuracy loss.

While FPGAs generally exhibit greater energy efficiency compared to general-purpose GPU systems, the level of efficiency is contingent upon the specific operation and inference engine employed. A study in [87] juxtaposed an NVIDIA Tegra X2 board against a Xilinx Ultrascale, focusing on energy consumption and operational power. In the context of a complex operation like detection, both platforms demonstrated analogous results for these parameters. However, it is noteworthy that the FPGA platform was not fully optimized, and its performance exhibited considerable variability across various engines like PipeCNN [88] or xfDNN. The underlying challenge stemmed from the fact that the high-level synthesis tool (HLS) used to transfer CNNs to FPGA platforms proved to be less efficient than traditional design approaches.

Conversely, for more straightforward operations like classification, a notable disparity in operational power was evident. General-purpose GPU platforms exhibited an operational speed of approximately 600 FPS, whereas the FPGA achieved only 60 FPS. It is imperative to emphasize that the FPGA system in this context had not undergone comprehensive optimization for the algorithm. The paper also reveals a significant finding regarding performance scaling on embedded platforms. It was observed that throughput is linked to CPU clock speed rather than GPU clock speed due to the memory-bound nature of the application and the CPU's role in managing data transfers. Specifically, for simpler tasks like image classification, the GPU's compute operations are not influenced by GPU clock frequency changes because these tasks are less demanding on the data transfer process. Instead, they rely more on the computational power of the GPU, which remains sufficient even at lower clock speeds. In contrast, for more complex tasks such as object detection, the GPU handles more substantial workloads that require efficient data transfer and processing. In these cases, both the CPU and GPU clocks significantly impact throughput, as the CPU must efficiently manage data transfers to and from the GPU, while the GPU must process the data at a higher rate. Therefore, optimizing both CPU and GPU clock speeds is crucial for achieving better performance in complex, data-intensive tasks.

However, existing deep-network-based pedestrian and vehicle detection (PVD) methods suffer from low inference speed due to large redundancy computations. To address this challenge, the work presented in [89] proposes an innovative approach called Orthogonal Re-parameterized Networks (ORNet). They designed a Ghost Structural Re-parameterized (GS-ReP) module, shown in Fig. 7, which efficiently combines the low computational ghost method with structural re-parameterization.

This results in a multi-branch structure that generates rich features while simplifying the multi-branch structure into a single-branch structure. They introduce a Feature Orthogonal Loss (FOL) function to further reduce redundancies between features from different branches of GS-ReP. This produces low-redundancy features that enhance detection accuracy. By integrating GS-ReP into the small version of the YOLOv5-s model, they create ORNet. Extensive experiments demonstrate that their ORNet achieves state-of-the-art results. On the public KITTI dataset, ORNet's mean average precision reaches 91.01 %, with only 5.5M parameters. On the NVIDIA Jetson Xavier NX embedded device, ORNet achieves FPS ranging from 25.9 to 31.7 for 640 × 640 pixels sized images, depending on the batch size setting.

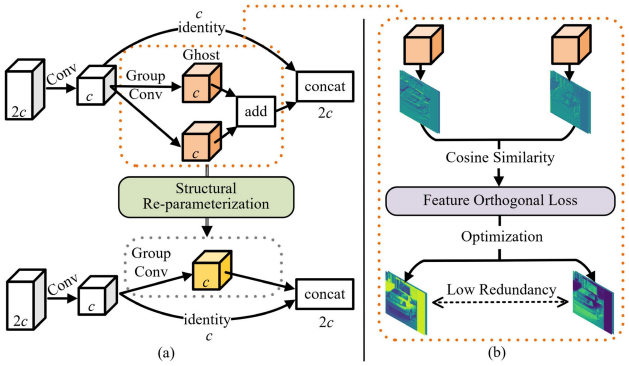


FIGURE 7. (a) The GS-ReP module, where c represents the number of feature map channels. (b) The FOL function.

In conclusion, the development of efficient and optimized neural networks is crucial in meeting real-time constraints on embedded systems. Utilizing layered techniques such as pruning, quantization, and network optimization, researchers have enhanced the performance and reliability of neural networks, making them more practical in embedded systems. The summaries of the collected articles are presented in Table 4 and Table 5, highlighting the key factors influencing performance in vehicle and pedestrian detection applications. Table 4 details the algorithms used, their modifications, the datasets employed, and the resulting performance metrics, providing insight into how these elements contribute to detection accuracy and efficiency. Table 5 complements this by focusing on the implementation details, including the software libraries, hardware platforms, image resolutions, and throughput, thereby illustrating the practical impact of these algorithms when deployed on specific hardware.

VIII. STATUS, BEHAVIOR, AND IDENTIFICATION OF THE DRIVER

A. STRUCTURE OF STATE, BEHAVIOR, AND DRIVER IDENTIFICATION ALGORITHMS

The principal aim of this class of applications is to extract significant signals pertinent to the driver's attention while ensuring minimal disruption to their focus within the driving environment. In addition to recording electrical signals from

TABLE 4. Data information and algorithm for car and pedestrian detection.

Ref.	Task	Model	Dataset	Algorithm Modification	Performance
[77]	Pedestrian Detection	Squeezemap	Caltech Pedestrian	Quantization	-
[78]	Pedestrian detection	DeepPed	Caltech Pedestrian	Quantization & pruning	31.38x Compressing, 0.4 % mean logarithm
[80]	PVD	MobileNet+SSD	COCO & ImageNet	Pruning	90.6 % AP
[84]	Vehicle Detection	Faster-RCNN	Own	Quantization & Pruning	83.66 % Accuracy
[82]	Object Detection	SqueezeDet	KITTI	ConvDet	76.7 MAP
[85]	Object Detection	Faster R-CNN	ETH	Pruning	90 % Accuracy
[86]	Object Detection	YOLO-v2	KITTI	Binarization	76 % mAP
[87]	Object Detection	YOLO	-	Hardware Mapping	-
[89]	PVD	ORNet	KITTI	GS-ReP & FOL	91.01 % mAP

TABLE 5. Implementation information for car and pedestrian detection.

Ref.	Library	Platform	Hardware	Picture Size (pixels)	Class	Throughput (FPS)	Efficiency (FPS/W)
[77]	-	Embedded CPU	NXP S32V234	640x480	1	30	15
[78]	Caffe	Embedded board	NVIDIA Jetson TK1	640x480	1	2.5 to 10	-
[80]	OpenVINO	CPU	Core i5-6500	672x384	1	40.2 to 63.5	-
[84]	Caffe	Embedded Board	NVIDIA Tegra X1	-	1	33	-
[82]	TensorFlow	GPU	TITAN X	1242x375	3	57	1.4
[85]	Caffe	Embedded Board	NVIDIA Jetson TX1	720x480	1	5	-
[86]	MXNET	-	-	720p	3	-	-
[87]	Xfcdnn & pipeCNN	Embedded Board and FPGA	NVIDIA Tegra X2 & Xilinx UltraScale+	408x408	-	600 60	75 3
[89]	-	Embedded Board	NVIDIA Jetson Xavier NX	640x640	3	25.9, 31.7	1.7, 2.1

the body, head movements and eye movements have also been considered potential sources of information regarding the driver's state. In [90], the head position is predicted by utilizing a combination of CNNs and Recurrent Neural Networks (RNNs). The input is composed of depth maps, and the output comprises three values that represent the three-dimensional angles: vertical, horizontal, and altitude. CNNs are utilized for feature dimensionality reduction and extracting meaningful information for the RNN, which is implemented through a two-layer Long Short-Term Memory (LSTM) network that produces the three-dimensional angle value directly. The datasets used in this study include the Biwi Kinect Head Pose dataset [91] and the ICT-3DHP database [92] with data augmentation. This approach yields superior outcomes compared to other research like [93], which relied solely on depth instead of a combination of RGB and depth. Specifically, the Mean Absolute Error (MAE) of the estimated head position was reduced from 4.42 degrees in [93] to 3.7 degrees in [90] on the Biwi Kinect Head Pose dataset, demonstrating a significant improvement in accuracy.

Additionally, an alternative approach has been employed for extracting eye position [94]. This method involves face detection using the Viola-Jones algorithm with Haar-like features, light normalization, the generation of selected eyes, and eye classification through SVMs. While this technique may not yield a highly precise evaluation of the driver's state, it holds the potential to contribute to the development of a comprehensive system centered on gaze detection.

Furthermore, deep learning emerges as a promising avenue when faced with highly intricate inputs or extensive information beyond the capacity of traditional classifiers. In a study outlined in [95], driver behavior underwent classification into ten distinct activities leveraging various CNNs. The model processes RGB images measuring $64 \times 64 \times 3$ pixels, employing well-established CNN architectures such as VGG-16, AlexNet, GoogleNet, and ResNet. Training datasets were derived from the Carnetsoft driving simulator. However, a notable drawback associated with employing intricate structures like these networks is the increased demand for training samples and relatively extended generalization times compared to conventional models (e.g., linear SVM).

Moreover, researchers in [96] devised an alternative detection and classification model based on a CNN and feature extraction. The initial network processes an RGB image as input and outputs the detection of eyes, mouth, and face. Subsequently, the second network employs these extracted features as input to classify individuals into states such as slouching, normal, or drowsy. For detection purposes, the Multitask Complex Network (MTCNN) [97] was employed, while the classification network consists of four streams, each structured similarly to AlexNet, accepting any feature from the preceding network as input.

Moreover, [98] introduces a hybrid method combining Generative Adversarial Networks (GAN) and Stacked Generalization Model (SGM) to overcome challenges in driver identification. The proposed method addresses issues such as sensitivity to different scenarios, overfitting, and reduced accuracy with an increasing number of drivers.

TABLE 6. Data information and algorithm for driver's status, behavior, and identification.

Ref.	Task	Model	Dataset	Data	Processing	Performance
[90]	Estimate Head position	CNN & RNN	Biwi & ICT-3DHP	RGB Image	-	2.0 MAE
[93]	Eye Position	SVM	Yale B, AR, ORL	RGB Image	Normalization, generating selected eyes	93.8 % Accuracy
[95]	Driver Behavior	CNN	Own	RGB Image	-	92 % Accuracy
[96]	Driver Status	CNN	Own	RGB Image	-	91.2 % Accuracy
[101]	Driver identification	MLP	Own	Gas and Brake pedal signals	Cepstral Analysis & FFT & BP	92 % Accuracy
[102]	Driver Behavior	RNN	Own	Accelerometers, gyroscopes, GPS	-	88.7 % Accuracy
[103]	Driver Status	CNN	Own	-	-	89.0 % Accuracy

By integrating different machine learning methods within the SGM ensemble and augmenting driving data, the system achieves improved accuracy and generalization performance. Notably, the novel use of GAN for augmenting driving signal data, the application of histogram features, and the utilization of smartphone data in SGM contribute to the advancements in driver identification research.

B. THE METHOD OF EMBEDDING AND PERFORMANCE OF STATE, BEHAVIOR, AND DRIVER IDENTIFICATION ALGORITHMS

In this category of applications, SVMs are often chosen due to their straightforward integration process and the advancement of tools designed for model implementation on both hardware and software platforms. Here, model implementation refers to the process of generating and optimizing code for embedded platforms. Logistic regression embedding is also deemed easy to employ while still providing favorable results [99]. Similarly, neural networks can be more readily implemented through frameworks like Tensorflow and Theano, as well as specialized hardware like NVIDIA Jetson slaves and software libraries like TFLite or TFMicro. An illustrative instance involves a system consisting of a CNN and RNN, which was implemented in NVIDIA Jetson TX1 and TK1 platforms to estimate head position [90]. The input comprised images with a resolution of 64 $\times$ 64 pixels, which were captured using a Kinect camera [100] and the Libfreekinect library. The network consisted of three convolutions complemented by a maximum pooling stack as well as two additional convolution layers and two LSTM layers. This system could attain 19 FPS on TK1 using solely a GPU and 30 FPS on TX1 when employing the cuDNN library and GPU.

The study [95] successfully implemented four distinct CNNs, including VGG-16, into a Jetson TX1 platform. The research addressed the challenge of high latency during inference, a common issue with CNNs, by achieving an impressive 92 % accuracy in tasks where the volume of input data exceeded the processing capacity of traditional algorithms. Despite the latency issue, which resulted in an 8 FPS rate in ResNet when processing images with 64 $\times$ 64 $\times$ 3 pixels, the study demonstrated the potential of deep learning in real-time detection of distracted driving. Although 8 FPS falls short of the typical 30 FPS threshold for real-time processing, the study highlights the promise of deep learning techniques in this domain, suggesting that with further optimization, real-time performance could be achievable.

As previously alluded to, the implementation of embedded deep learning aims to devise ways to streamline the networks. In [96], the authors propose removing two branches in multi-branch CNNs designed to detect sleepiness while also training a more extensive network to train a smaller one, a process known as knowledge extraction. The entire system encompasses multi-task cascading convolution networks (MTCNN) for detecting eyes and mouths, as well as another network, namely the driver's drowsiness detection network (DDDN), which has multiple branches assigned to computing the level of drowsiness.

Removing branches reduces excessive information, increasing the speed from 6.1 FPS to 13.5 FPS on NVIDIA Jetson TK1. Compressed models (knowledge distillation) only suffered a 3.6 % loss in accuracy (from 94 % to 91 %) but reached 14.9 FPS. Nonetheless, two main challenges of the study exist. First, the dataset is not universal and therefore lacks verifiability, as is the case with many other studies. Second, optimizations notwithstanding, the delay does not reach satisfactory operational performance. This problem hampers the use of CNNs for embedded applications that necessitate real-time performance. Finally, [101] depicts ANNs implemented in the KINTEX-7 Xilinx FPGA family for identifying drivers based on the signals from the gas and brake pedals. The objective is to compress the multilayer perceptron matrix operation to hardware and implement fewer software parts, such as bandpass filtering preprocessing.

In the realm of embedded systems, a strategy for facilitating the integration of NNs into Microcontroller Units (MCUs) is elucidated in [102]. The authors employ an RNN within a smartphone to detect driving activities. They adopt a strategy for quantizing and compressing weights, as outlined in [104], where weights are quantized to 8-bit unsigned integers, and inputs are quantized dynamically. The multiplication operation is accumulated in 32-bit integers and subsequently converted to float when biases are added and used for the

TABLE 7. Implementation information for driver's status, behavior, and identification.

Ref.	Library	Platform	Hardware	Picture Size (pixels)	Class	Throughput (FPS)	Efficiency (FPS/w)
[90]	Keras	Embedded	NVIDIA Jetson TK1 & TX1	640x480	3d Angles	19.3	-
[93]	-	Embedded	INTEL Bay-Trail	-	Eye/Not eye	16	-
[95]	TensorFlow	Embedded	NVIDIA Jetson TK1	64x64	10 Driving Behaviors	14	-
[96]	TensorFlow	Embedded	NVIDIA Jetson TK1	640x480	Normal, Yawning, drowsy	14.9	5.5
[101]	-	FPGA	XC7k32T Xilinx	-	N Drivers	-	-
[102]	-	Smartphone	-	-	18 Driving Patterns	-	-
[103]	OpenCV & TFLite	Smartphone	Qualcomm Snapdragon 430	224x224	Rested and Sleep	5 to 10	-

computation of the activation function. This approach enables a reduction in the model's weight from 412 KB to 77KB without a noticeable loss in accuracy and a decrease in latency. This strategy is further expanded in [35] to facilitate full integer inference.

The same strategy is incorporated in the TensorFlow Lite [105] framework, which aids in mitigating the memory and latency constraints imposed by CNNs. In [103], the authors implement a drowsy driver classifier using a quantized CNN (MobileNet) applied to video recordings on a smartphone. TensorFlow Lite provides functions to quantize the networks and an Application Programming Interface (API) for invoking the networks and implementing inference in resource-constrained devices, thereby simplifying the task of embedding NNs. However, latency remains a challenge that limits the achieved frame rate to 5 FPS - 10 FPS, which is insufficient for many real-time contexts. This rate is inadequate for many real-time applications. In contrast, CNN models that can process more than 30 FPS are classified as operating in real-time [74]. Relevant articles, as illustrated in Table 6 and Table 7, detail the critical parameters that influence the system's performance, including tasks, neural networks, libraries, datasets, algorithm modification, hardware platforms, and implementation results.

IX. RECOGNITION OF TRAFFIC SIGNS

A. STRUCTURE OF THE TRAFFIC SIGN RECOGNITION ALGORITHM

Many algorithms for the detection of traffic signs and signals follow a similar structural outline: two distinct stages, namely detection and identification. The first stage involves the detection or generation of a region of interest (ROI), which is used to define the area in the image where the sign is located. The second stage involves the identification, in which the detected sign is categorized as one of the predefined cases.

While most literature adopts the detection and identification approach, some studies focus exclusively on detection. In the work of [106], an encoder-decoder CNN based on U-Net is utilized to generate grayscale images. The images undergo a process of binarization, after which they are clustered using the DBScan method [107]. Following this, a filtering process is applied to determine the final regions where the traffic lights are located.

In the significant study referenced as [108], traffic sign detection (TSD) is tackled using a knowledge-based attention mechanism implemented through an RNN. The authors employ SqueezeNet in conjunction with a fire module [76] to generate feature maps for LSTM cells. A unique adaptation is the integration of an attention mechanism within the LSTM cells, enabling selective focus on specific image segments during the decision-making process. This mechanism allows the model to selectively focus on specific segments of the image during the decision-making process, thereby enhancing the model's ability to detect traffic signs. Furthermore, the use of an inverse Gaussian method infuses domain knowledge into the detection procedure. Following the ROI, the subsequent step entails the extraction of relevant features for classification, which are then directly input into the final classifier. This approach represents a comprehensive and innovative solution to the challenges of traffic sign detection.

CNNs are also widely employed as classifiers for traffic sign detection. In a comprehensive investigation [109], a CNN is employed for the direct classification of diverse German traffic and guidance signs derived from the ROI obtained during detection. The network architecture comprises two convolutional stacks with maximum pooling, and an extra convolutional layer, and culminates in two fully connected final layers.

Another study using CNNs uses as input the ROI derived during the detection phase [110]. The network structure is similar to the one in [109], which includes convolutions and maximum pooling, followed by fully connected layers for classification. A distinct utilization of CNNs is illustrated in [111], where a dual-task network is designed to concurrently handle classification and regression tasks, specifically for classifying label shapes and determining two-dimensional signal direction. Building upon either VGG16 or Inception-V2 as the base network, features are generated for both branches. The detection component incorporates an adapted version of the SSD and utilizes a technique known as Non-Maximum Suppression (NMS) [58]. The network's outputs transform to delineate the final sign.

B. THE METHOD OF EMBEDDING AND PERFORMANCE OF TRAFFIC SIGNS

The application of FPGAs in road sign detection has been explored in various studies, one of which is outlined in [112].

This study presents a technique for detecting two distinct traffic lights, green and red, by dividing bubble detection into two branches. Each branch involves a single pass with four label connections, registering the position of bubbles in one table and storing pixel connections in another. These tables are subsequently merged to determine the positions of the bubbles. The system comprises three main components: preprocessing, bubble detection using Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM), and identification of green and red traffic lights.

The solution was implemented on a Xilinx Zynq ZC-702 board, balancing the workload between the FPGA and the on-chip ARM processor. The AXI4 streaming Direct Memory Access (DMA) video pathway was utilized to facilitate rapid memory access, resulting in high-speed data transfer from the FPGA to the frame buffers in Double Data Rate (DDR) memory. The detection system reportedly identifies green and red traffic signals with recall rates of 92.11 % and 94.44 %, respectively, although the performance varies based on the count of red lights present in the image. This underscores the fact that, at the time, the state-of-the-art machine learning techniques were not yet ready for practical deployment in real-world scenarios. The system achieves a detection rate of 60 FPS, which can potentially be increased to 100 FPS.

However, the study lacks detailed information regarding data formats and calculations, providing only a block-level illustration of the system's chip-level architecture. This omission leaves room for further exploration and clarification in future research. An alternative implementation was also explored in [113], which utilized a platform combining a Xilinx Virtex II XC2V1000 FPGA and Axeon VindAX processor on a PCI development board for System On a Module (SOM) in image preprocessing. The detection technique used allowed for the full identification of signs despite changes in scale (down to 25 pixels), rotation (up to 20 %), and occlusions (up to 15 % - 20 % for vertical occlusions and 5-10 % for horizontal occlusions, contingent on the occluded region) at speeds of 19 FPS to 21 FPS in Video Graphics Array (VGA) images. The authors reported that the SOM maps used for training were based on images from Spain and the Czech Republic, whereas the system was evaluated using image data from Britain. It used its own test set and achieved an estimated success rate of nearly 100 %.

In the investigation detailed in [114], the focus is directed toward two pivotal aspects within the realm of autonomous driving tasks: traffic sign classification and detection. The structure of the paper is organized into two coherent sections, systematically examining tradeoffs among five distinct deep learning frameworks during the classification phase and subsequently undertaking the training of six models dedicated to traffic sign detection in the second phase. The inquiry entails a meticulous benchmarking process, evaluating training speed and inference accuracy on GPUs, followed by the deployment of selected models onto FPGAs to assess performance under diverse parameters. The study strategically employs the German Traffic Sign Recognition Benchmark and Detection Benchmark datasets, incorporating CNN models such as ResNet-20, ResNet-32, ResNet-18, ResNet-50, MobileNet-v1, MobileNet-v2, SqueezeNet-v1.1, VGG, and SSD models. Particularly noteworthy is the utilization of OpenVINO for FPGA deployment, effectively addressing challenges associated with hardware variability. The paper makes substantial contributions in the form of a comprehensive benchmark, a nuanced analysis of FPGA deployment performance, optimization methodologies for both classification and detection tasks, and a comparative evaluation of power efficiency between FPGA and GPU in the context of traffic sign detection.

In the work documented in [115], an enhanced CNN architecture is introduced, amalgamating principles from VGGNet with advanced image-preprocessing techniques. The proposed methodology leverages a combination of pruning and post-training quantization-based optimization, yielding an accuracy loss of less than 1 %. The architectural efficacy is meticulously assessed through comprehensive training, testing, and validation processes utilizing the German Traffic Sign Detection Benchmark (GTSDB) within Google's TensorFlow framework. The outcomes reveal a validation accuracy of 99.2 %, while the test accuracy attains a flawless 100 % for novel input inference. Notably, the experimental findings showcase a marked reduction in the memory footprint of the CNN model, showcasing its adaptability for seamless implementation on FPGAs.

In [116], the authors present a cost-effective CNN-based hardware accelerator for real-time Traffic Sign Recognition (TSR). Initially, they introduce an innovative hardware-friendly quantization technique aimed at diminishing computational complexity. This method enables the reconstruction of the CNN, ensuring that all operations, including the skip connection path within residual blocks, exclusively employ integer arithmetic. Consequently, the computational overhead is mitigated by substituting the quantization affine mapping process with a more efficient shift operation. Subsequently, the proposed hardware accelerator incorporates two parallelization strategies, strategically balancing the imperative of real-time inference against resource consumption.

Furthermore, the authors introduce a streamlined and efficacious hardware design approach tailored to handle the skip connection path within residual blocks. This design optimizes the data flow of the skip connection path, concurrently reducing additional internal memory usage. Experimental findings demonstrate that the fully integer-based CNN reconstruction demands merely 24 million integer operations (IOPs) and possesses a compact model size of 0.17 MB. Impressively, the proposed CNN attains a remarkable TSR accuracy of 99.07 %. When implemented on a Xilinx ZC706 System-on-Chip (SoC), the envisioned hardware accelerator achieves commendable computation

performance, registering 960 million Operations Per Second (MOPS) and an impressive frame rate of 40 FPS.

Table 8 and Table 9 provide an overview of the differences between the articles and a comparative analysis.

X. ROAD DETECTION AND SCENE PERCEPTION

A. STRUCTURE OF ROAD RECOGNITION AND SCENE UNDERSTANDING

Scene understanding, in the context of autonomous vehicles, pertains to the process of segmenting and detecting elements within the driving environment to generate an accurate depiction of the surroundings. Although not a standalone function, scene understanding is integral to the operation of self-driving cars, as it provides critical information for various autonomous driving tasks, including collision avoidance and lane changing. It can also be directly incorporated into these tasks to enhance their efficacy, underscoring its importance in the overall functionality of autonomous vehicles. In certain scenarios, the focus may be on detecting or segmenting only the road, rather than segmenting the entire scene. Traditional methods for road detection relied on manually extracted features, such as modeling the road through polygons, the color of lane markers, or template matching [117]. However, recent advancements have seen the adoption of deep learning techniques as the primary approach to road detection.

Comprehensive scene understanding necessitates semantic segmentation, which can be directly executed using CNNs. This process, however, imposes substantial computational demands, often requiring hundreds of GOPS or TOPS. Given the limited resources of the platform, CNNs frequently require adjustments or modifications to decrease their memory usage and computational demands. An alternative approach involves modifying the network for direct hardware mapping, which is advantageous when low energy consumption and/or high throughput are desired. This highlights the ongoing challenges and potential solutions in the field of scene understanding for autonomous vehicles.

B. THE METHOD OF EMBEDDING AND PERFORMANCE OF ROAD DETECTION AND SCENE UNDERSTANDING

In their work [118], the authors present three methodologies aimed at facilitating the integration of CNNs into devices constrained by limited computational resources. The outlined techniques encompass quantization, layer fusion, and sparse multiplication. The primary objective is to assess the available free space for execution, demonstrated through the transformation of a pre-trained ResNet10 network using the Cityscapes dataset for implementation on a Texas Instruments TDA2x SoC.

The initial technique applied is quantization, which entails performing quantized CNN inference on embedded devices characterized by limited resources while minimizing accuracy degradation. Following quantization, layer fusion is implemented, consolidating diverse operations into a unified process. For instance, 2D convolution and maximum pooling can be amalgamated to minimize the need for intermediate result storage. Furthermore, specific inputs are acquired, and loaded into internal memory once, thereby reducing memory demands when multiple convolutions utilize them, as observed in Inception modules. In a general context, the utilization of layer fusion contributes to the reduction of memory bandwidth requirements and enhances computational speed.

The final executed method is sparse multiplication, strategically avoiding computations for values in a matrix that are zero or close to it. Lightweight matrices are obtained by applying regularization during training, such as L2 or L1 norms. This approach aids in minimizing delay by reducing the total multiply-accumulate operations (MAC). Through the application of sparse convolution alone, the authors achieve a noteworthy speed increase of 93.3 times, with only a marginal 1.79 % reduction in mean intersection over union (mIoU) [118].

An additional significant contribution of this article lies in its elucidation of the process of implementation of a deep learning network within an embedded system. Typically, Resource-Constrained devices primarily engage in inference tasks. Typically, the network is trained in an appropriate setting, for instance, a PC that is outfitted with a GPU. Subsequently, the model is converted to a format suitable for embedding, employing tools like Tensorflow Lite or Glow. Finally, a library is employed to invoke the adapted graph, generalize the results, and execute it using the Texas Instruments Deep Learning Library (TIDL).

An alternative approach to incorporating CNNs involves optimizing the computational overhead associated with operations like convolutional layers. By employing depth-wise convolutions, as initially proposed by [34], researchers have successfully crafted lightweight CNN architectures. Notably, in [119], the authors demonstrate the integration of a modified version of the SDD [58], specifically tailored for multi-scale object identification.

The underlying principle of depth-wise separable convolutions is the separation of standard convolutions into independent operations concerning depth. Specifically, the initial operation applies a filter to each channel as is, without any alterations. The subsequent operation employs a pointwise convolution to concatenate the output resulting from the first operation. This sequence allows for efficient processing and transformation of the input data. The calculation of computational complexity for conventional convolutions in comparison to depth-wise separable convolutions can be performed using (1), where N represents the number of output channels and D_k denotes the size of filter kernels. The use of such separable convolutions can lead to computational savings.

$$1/N + 1/(D_k^2) \tag{1}$$

In [120], the authors present a dedicated network for line position and direction, which has been designed for embedding in an NVIDIA Drive PX platform. This network proposition

TABLE 8. Data information and algorithm for traffic sign recognition.

Ref.	Task	Pre-detection	Detection	Pre-classification	classification	Dataset	Performance
[106]	Traffic Light Detection	CNN & Binarization	DBScab & Cluster Filtering	-	-	Own	92.22 % DICE
[108]	13 Class Traffic signs	KB-RANN	SqueezeNet	-	-	BTSD	81.0 % mAP
[109]	24 Traffic signs pyramid	Image	Adaboost	HOG	CNN	TI Automotive	-
[110]	43 traffic Signs	-	MB-LBP	Histogram Equalization	CNN	GTSBR & BTSD& Own	94.0, 78.34 % Accuracy
[111]	Prohibitive, Mandatory & Danger Signs	Cropping & Resizing	CNN & Template Transform	-	CNN & Template Transform	STDS & GTSBR	82.8 to 88.4 % mAPS
[112]	Green and red light	RGB to HSV & Binarization	4-Connectivity labeling	Shape Filter & resizing & HOG	SVM	Own	93.27 % Recall
[114]	traffic sign classification and detection	FP11	SSDs	FP11	CNN	GTSRB & GTSDB	97.3 %
[115]	43 traffic Signs	local histogram equalization	-	pruning and post-training quantization	CNN	GTSDB	96.5 %
[116]	traffic sign recognition	-	-	hardware-friendly quantization	CNN	GTSRB	99.07 %

TABLE 9. Implementation information for traffic sign recognition.

Ref.	Library	Platform	Hardware	Picture Size (pixels)	Throughput (FPS)	Efficiency (FPS/W)
[106]	Keras	Embedded	NVIDIA Jetson TX2	256x455	15	1.5
[108]	C	Embedded	NVIDIA Jetson TX1	542x412	10	1.0
[109]	TI's Vision SDK	Embedded	TI's TDA3x SOC	-	15	-
[110]	TensorFlow	Smartphone	Huawei P9 Lite	640x480	10	-
[111]	Caffe /SNPEI SDK	Embedded	Qualcomm Snapdragon 820A	1280x720	0.2-7	-
[112]	-	FPGA	Xilinx Zynq ZC-702	1024x768	60	-
[114]	TensorFlow	FPGA	Arria 10	32x32	24	3.5
[115]	TensorFlow	FPGA	Xilinx FPGA	32x32	-	-
[116]	Pytorch	FPGA	Xilinx ZC706	32x32	40	5.8

achieves 100 FPS with images of 240 × 260 resolution. The proposed network architecture is relatively simple for visual processing, consisting of two stacks of convolutional layers, which are followed by pooling and two fully connected layers, before being dropped. Both stereo camera images and synthetic-generated images are utilized.

Another study [121] details a multi-task segmentation and detection system, which addresses the high latency issues associated with competing systems. By utilizing a template that provides common features at different scales, the authors achieve favorable results. The structure of the network comprises a shared encoder and decoders that are specific to each task. The encoder is derived from Inception V-2, while the decoders are built on an FCN for segmentation tasks and an SSD-based network for detection tasks. The authors demonstrate a 32.5 FPS delay at a standard resolution of the KITTI dataset running on an NVIDIA Jetson TX2. It should be noted that the results reported are far from real-time processing.

Another study [122] explores a CNN running on a smaller-scale chip (45.0 square millimeters). The chip was able to achieve 241 FPS at p1080 with a power consumption of only 80 mW. The proposed CNN architecture was specifically designed for road segmentation and includes two novel features: two additional feature channels that are associated with the coordinates of rows and columns, as well as an identical number of neurons in each layer to allow for the reuse of hardware between layers.

Diverging from prevalent methodologies, the investigation detailed in [123] directs its focus toward heterogeneous systems integrating both FPGA and GPU accelerators. In a concerted effort to address the challenge of intra-accelerator workload consumption, the study introduces innovative optimization techniques, namely accelerator execution overlapping and dynamic workload tuning. Emphasizing equity in computational resource utilization, the researchers tailor applications with data-level parallelism, presenting a comprehensive procedural framework applicable to early-stage development of FPGA-GPU heterogeneous systems using OpenCL. Leveraging foundational research, the study extends and customizes a heterogeneous design, implementing the proposed optimization procedure in the context of two Lane Detection Algorithms (LDA). The experimental findings underscore substantial speedup gains, registering an average improvement of 2.09× and 1.83× for particle-filter-based and RANSAC-based LDAs, respectively, over native parallel implementations.

In [124], the authors proposed compressed versions of two well-known 3D object detectors, PointPillars and PV-RCNN, utilizing dictionary learning-based weight-sharing techniques for running on NVIDIA's Jetson TX2. They applied the VQ and DL weight-sharing techniques to the

PointPillars and PV-RCNN models, targeting their convolutional layers, and measuring the performance drop induced by the acceleration, compared to the original networks.

In [125], the authors investigated the application of weight-sharing methods in deep learning-based scene analysis for automotive scenarios. The impact of transforming two well-known DNN models was evaluated on 2D image-based, 3D LiDAR-based, and fusion-based detection approaches. The KITTI dataset was used for the evaluation of the presented approaches. Comparing the uni-modal versus multi-modal detection approaches, it was demonstrated that the multi-modal fusion not only improves the performance of the individual detectors but also considerably improves the performance of the networks when they are accelerated/compressed by the considered weight-sharing techniques.

In the study [126], the researchers introduce a novel approach for the hardware acceleration of the YOLOv7-tiny object detection algorithm on FPGAs using HLS tools. The proposed method achieves a latency of 15 ms, making it suitable for real-time applications. It also reduces the usage of DSP units by 90 % and flip-flops by 60 %, compared to contemporary designs [37]. The method is predicated on offloading memories from BRAM to DRAM and configuring them via a Direct Memory Interface (DMI) using the AXI interface. This strategy can decrease BRAM usage and facilitate the design of low-budget SoCs such as XC7Z7020 or XC7Z7045.

However, due to the latencies associated with data transfer from DMI, this approach may result in increased latencies compared to those obtained in experiments with the XC7Z7100 board. Nonetheless, even with this new strategy on XC7Z7045, the design can meet real-time requirements with a latency of approximately 32 ms in the synthesized design. The work employs two tensors, represented as separate C++ structs, to capture network parameters and output feature maps. To enable the accelerator to operate on the XC7Z7045, certain arrays in these tensors must be deallocated from BRAM and set via the AXI4 interface for optimal model performance. This implementation on XC7Z7045 slightly alters both Flip-Flop (FF) and Look-Up Table (LUT) usage due to control signals and the rearrangement of new tensor arrays in BRAM. The authors also evaluate the accuracy of the final model on the MS-COCO test dataset and find that it is only 1.1 % lower than the original model. The authors conclude that their method holds promise for the hardware acceleration of YOLOv7-tiny on FPGAs.

This article [127] introduces StereoEngine, an innovative streaming hardware architecture designed as a comprehensive stereo vision accelerator. Engineered for real-time efficiency and energy optimization, StereoEngine calculates dense depth seamlessly. Its distinctive approach incorporates a binary neural network (BNN) to enhance disparity by acquiring discriminative binary descriptors. The hardware-friendly implementation operates exclusively on an FPGA, negating the necessity for external CPUs or GPUs. Through meticulous algorithmic refinements, encompassing layer integration and convolution reuse, StereoEngine attains significant speed enhancements and improvements in energy efficiency. Experimental assessments underscore its superiority over software-based counterparts on Nvidia GPUs and other FPGA-based stereo vision accelerators [71], yielding impressive speedups of up to 50 times and an energy efficiency improvement of 211 times. Noteworthy is StereoEngine's outperformance of existing FPGA implementations in accuracy on the challenging KITTI 2015 dataset, achieving a remarkable speed of 114.08 FPS for processing 1024×768 pixels images, all within a power consumption of merely 3 W.

Presenting Lite-Stereo [128], an efficient stereo vision accelerator rooted in BNN principles and purposefully tailored for FPGAs. Diverging from StereoEngine [127], Lite-Stereo adeptly addresses resource constraints through the introduction of the pioneering RES-BNN architecture, resulting in a remarkable 71 % reduction in resources and a notable $2.71\times$ increase in processing speed.

An enhanced module, Semi-Global Matching (SGM) [129], is a computer vision algorithm used for estimating a dense disparity map from a rectified stereo image pair. It performs line optimization along multiple directions and computes an aggregated cost by summing the costs to reach each pixel with disparity from each direction. This method has found wide adoption in real-time stereo vision applications such as robotics and advanced driver assistance systems due to its predictable run time, favorable trade-off between quality of results and computing time, and suitability for fast parallel implementation in ASIC or FPGA. Drawing inspiration from Real-time Raster-Respecting SGM (R^3SGM) [130], a resource-efficient method designed to run in real-time on a low-power FPGA, contributes to a 56 % reduction in ALUTs consumption and a substantial $2.7\times$ acceleration in processing speed.

CNNs have been successfully applied to stereo matching. They have greatly improved the speed and accuracy of stereo matching compared to traditional methods. Due to its wide adoption of SGM in real-time applications and its suitability for implementation on resource-constrained hardware like FPGAs, we focus on it. Our work specifically addresses the optimization of hardware architectures for resource-demanding bottleneck modules in SGM, hence the emphasis on SGM. Comparatively, Lite-Stereo surpasses StereoEngine with impressive reductions of 60 % and 29 % in ALUTs and RAM bits, respectively, while upholding accuracy and energy efficiency, particularly evident in the challenging KITTI2015 dataset. Operating at 53 FPS for 1280×960 images with a 128-disparity range, Lite-Stereo fulfills real-time requirements, offering a minimal hardware footprint suitable for integration into robotics systems.

Through this substitution, R^{3}SGM can opt for unidirectional minimization, facilitating the sequential processing of the image in raster order. The aggregation cost for the current pixel, denoted as P, is computed by the R^{3}SGM algorithm through the accumulation of contributions from its four neighboring pixels, namely: 1) left (r0); 2) top left (r1); 3) top (r2); and 4) top right (r3). The conclusive aggregated cost, represented as C_{agg}(p, d), for the pixel p at a specific disparity, d, is iteratively determined through a recursive computational procedure (2).

$$\begin{aligned} C_{agg}(p,d) &= C(p,d) \\ &+ \frac{1}{|X|}\sum_{x\epsilon X}(min\{C_{agg}(p-x,d), C_{agg}(p \\ &- x, d+1) + P_1 \min_k C_{agg}(p-x,k) + P_2\} \\ &- \min_k C_agg(p-x,k)) \end{aligned} \tag{2}$$

In the context of this formulation, denoted as C(p, d), the initial cost associated with the current pixel, p, for a given disparity, d, is considered. Here, the variable X signifies the aggregate count of neighboring pixels utilized in the aggregation process. The aggregated cost of a neighboring pixel, denoted as C_{agg}(p - x, d), is representative of the combined cost involving the adjacent pixel, x. The factors P_1 and P_2 are introduced as penalty factors within this computational framework, contributing to the overall optimization of the disparity mapping process.

Overall, there is a notable degree of uniformity in the use of datasets and performance metrics throughout the reviewed studies. Such consistency facilitates direct comparison to network architecture, hardware requirements, and algorithm modification. Table 10 summarizes the selected papers with a focus on performance and implementation specifications.

XI. LOCALIZATION

A. STRUCTURE OF LOCALIZATION ALGORITHM

SLAM is an advanced algorithm used in car navigation. As previously mentioned, it enables the creation or updating of a precise map of an unknown environment while simultaneously tracking the vehicle's location. Dense SLAM algorithms [131] generate detailed environmental maps but require high computational resources, making them unsuitable for edge devices. Sparse SLAM [132], [133], [134], [135], on the other hand, is more efficient but only focuses on limited landmarks or features. Semi-dense SLAM [136] has emerged as a compromise, offering better computational efficiency than dense methods and a denser map than sparse methods. A SLAM system consists of a front-end that links sensory measurements to landmarks and infers robot motion, and a back-end that minimizes measurement noise errors. Common back-end methods include filter-based and numerical optimization-based algorithms.

Mobile robot localization faces the challenge of maintaining accuracy and efficiency under resource constraints. High frame rates are required in SLAM systems to prevent feature loss between frames. While most SLAM algorithms operate on power-intensive platforms, the need for efficient, real-time solutions has led to the development of specialized chips and accelerators. FPGA SoCs, offering rich sensor interfaces and programmability, have been increasingly utilized.

EKF-SLAM [132] algorithms, utilizing the Extended Kalman Filter for SLAM, are typically feature-based and employ maximum likelihood for data association. Bonato et al. [137] pioneered the first FPGA-based architecture for EKF-SLAM, achieving notable power efficiency. Oriented FAST and Rotated BRIEF SLAM (ORB-SLAM) [133], a widely-used sparse SLAM algorithm, identifies feature extraction as its most computation-intensive stage. To address this, Fang et al. [138] designed a hardware ORB feature extractor, significantly outperforming ARM Krait and Intel Core i5 in both computation latency and energy consumption. Liu et al. [139] proposed eSLAM, an energy-efficient FPGA implementation, to expedite both feature extraction and feature matching stages. Despite the computational complexity of EKF-SLAM, Montemerlo et al. [134] introduced Fast-SLAM, an efficient SLAM algorithm that decomposes the SLAM problem into a robot localization problem and a landmark estimation problem. Gu et al. [135] implemented the VO-SLAM algorithm, characterized by low computational complexity, on a DE3 board, achieving a processing speed of 31 FPS with 30000 global map features and significant energy savings.

Semi-dense SLAM algorithms have emerged as a compromise between sparse and dense SLAM algorithms, aiming for improved efficiency and denser point clouds. However, they often require powerful multicore CPUs for real-time processing. A notable example is the Large-Scale Direct Monocular SLAM (LSD-SLAM), a widely-used semi-dense SLAM algorithm that operates directly on image intensities for both tracking and mapping, tracking the camera through direct image alignment, and estimating geometry from semi-dense depth maps obtained by filtering multiple stereo pixel-wise comparisons.

In recent years, CNNs have made significant advancements in the localization and understanding of robots and autonomous vehicles, offering a contrast to traditional manual methods. Consider feature extraction, a fundamental component of SLAM. The SuperPoint method, which is based on CNNs, [140] has been shown to achieve a matching accuracy that is up to 30 % higher than the handcrafted ORB. Other CNN-based approaches, such as DeepDesc [141] and GeM [142], have also made substantial strides in both feature extraction and descriptor generation. However, it is important to note that CNNs come with a higher computational complexity and increased memory requirements.

B. THE METHOD OF EMBEDDING AND PERFORMANCE LOCALIZATION

Several studies have investigated the implementation of CNNs on FPGAs. Specifically, Xilinx DPU represents one of

TABLE 10. Data information and algorithm for Road detection and scene perception.

Ref.	Task	Dataset	Algorithm Modification	Hardware	Performance	Picture Size (pixels)	Throughput (FPS)	Efficiency (FPS/W)
[118]	Finding drivable Space	Cityscapes	Quantization inference & layer Fusion & Spars CNN	TI TDA3x SOC	95.90 % MIOU	1024x512	20	-
[119]	Multi-scale Object Identification	Own	Depth-Wise conv	Google Pixel	60 % Precision Detection	-	2	-
[120]	Lane Position and Orientation	Own	Caffe	NVIDIA Drive PX	89.32 % accuracy	240x360	100	-
[121]	Object & Drivable Terrain detection	KITTI	-	NVIDIA Jetson TX2	82.1 % mAP	1392x512	5.32	-
[122]	Road Detection	Cityscapes & KITT	Hardware Mapping ASIC 32nm	0.45mm2	91.60 mAP	1920 × 1080	241	3000
[123]	Lane Detection	Caltech lanes	Loop Unrolling and & Memory Access Coalescing	Stratix V G5	Speedup of 2.09×	-	-	-
[124]	3D Scene Analysis	KITTI	Weight Sharing & Vector Quantization	NVIDIA Jetson TX2	3 % Accuracy loss	1242×375	-	-
[128]	Stereo Estimation	KITTI	enhanced R^3SGM + RES-BNN	Altera Stratix V	6.41 % error rate	1280×960	53	11.8
[127]	Stereo Estimation	KITTI	SGM + BNN	Altera Stratix V	6.36 % error rate	1280×960	64	11.2

TABLE 11. Data information and algorithm for localization.

Ref.	Task	Dataset	Firmware Design	Hardware	Performance	Picture Size (pixels)	Throughput (FPS)	Efficiency (FPS/W)
[143]	Feature Extraction	HPatches	Quantization and Weight Reduction	ZCU102	59 ms Speed	640x480	20	-
[144]	Real-Time Localization & sensor Fusion	CIFAR-10	Network Compression & Hardware Architecture Search	Xilinx ZU11EG + NVIDIA Jetson Xavier SoC	30 km/h Vehicle Speed	1216x608	10	0.1
[145]	Decentralized Place detection	KITTI	Pipeline Scheduling Fixed-Point	Xilinx ZCU102	13 ms/Frame	608x160	77	-
[146]	DSLAM, Feature Extraction	-	Quantization	ZU9 MPSoC	300MHz Frq	640x480	20	-

the most advanced programmable applications dedicated to CNNs, featuring specialized instructions and efficient performance across varying CNN topologies. In [143], a hardware architecture was proposed to expedite feature extraction for SuperPoint based on CNNs on the Xilinx ZCU102 platform, achieving 20 FPS in an SLAM system. The proposed design optimizes a software data stream to handle any additional post-processing operations in CNN feature extraction networks, leveraging 8-bit fixed point numbers for post-processing operations and the CNN infrastructure. Compression techniques such as data quantization and weight reduction have been commonly used for robotics and similar CNN-inspired designs [19].

Another study [144] introduced an efficient computational platform design for autonomous vehicles. The authors proposed neural network compression and a hardware architecture search to reduce workload volume at the software level. Furthermore, custom accelerators for pre- and post-deep learning algorithm processes were also introduced at the hardware level. Finally, the authors presented a hardware platform design, named NOVA-30, and a self-driving car evaluation project.

In [145], the authors leverage the power of CNN in an embedded Decentralized SLAM (DSLAM) system, employing Depth-VO-Feat [147] for monocular Visual Odometry (VO) and NetVLAD [148] for Decentralized Place Recognition (DPR). They construct a real-time, CNN-based monocular DSLAM system on an embedded FPGA platform. Recognizing that VO tasks require higher numerical precision than other applications like image classification [11] and object detection [149], they propose a fixed-point fine-tuning method for CNN-based VO. The study also explores the impact of DPR frequency on DSLAM performance, proposing a cross-component pipeline scheduling method to enhance DPR frequency. This work, to the best of their knowledge, is the first to implement all components of monocular DSLAM with CNN. The system is deployed on the Xilinx ZCU102 Multi-Processor Soc (MPSoC) hardware platform, equipped with a DPU. The proposed DSLAM system is illustrated in Fig. 8.

In [146], an innovative solution is introduced to facilitate multitasking in embedded robots utilizing CNN accelerators. The proposal centers around the development of an INterruptible CNN Accelerator (INCA), which employs a novel virtual-instruction-based interrupt mechanism. Specifically, the authors accelerate feature extraction and place recognition tasks of the DSLAM system on the same CNN accelerator within an embedded FPGA framework. Notably,

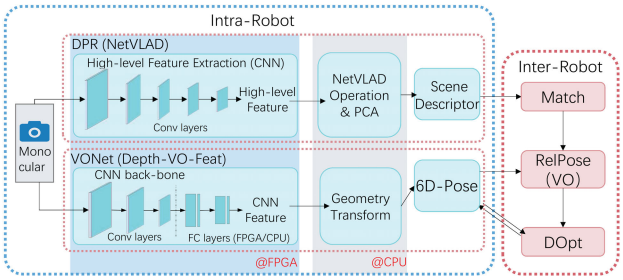


FIGURE 8. Depth-VO-Feat in DSLAM system as the monocular VO, and NetVLAD to do DPR [145].

experimental results demonstrate that compared to the conventional layer-by-layer interrupt method [37], their VI method significantly reduces the interrupt response latency to 1 %. This advancement holds promise for enhancing real-time performance in robotic applications. The details of selected articles and relevant implementation specifications and performance are presented in Table 11.

XII. DISCUSSION AND CONCLUSION

A. DISCUSSION

The performance and energy efficiency of CNN algorithms on FPGAs are impacted by several factors, including algorithmic modifications, hardware platforms, and optimization techniques. In this section, we analyze how different CNN architectures and algorithmic choices affect performance metrics like throughput, accuracy, and efficiency, as well as energy consumption, by comparing state-of-the-art solutions across various tasks, including object detection, pedestrian detection, driver status monitoring, traffic sign recognition, and road scene perception.

1) OBJECT AND PEDESTRIAN DETECTION

Tables 4 and 5 summarize implementations of CNNs for object and pedestrian detection on various hardware platforms, including FPGAs and GPUs. While traditional models such as Faster R-CNN and YOLO variants deliver respectable accuracy (e.g., YOLOv2 achieving 76% mAP on the KITTI dataset), significant gains in efficiency are achieved through algorithmic modifications like quantization and pruning. For example, the combination of pruning and quantization in Faster R-CNN on NVIDIA Tegra X1 yields an efficiency improvement without sacrificing much accuracy (83.66% accuracy), showing the trade-off between model complexity and energy consumption.

Furthermore, algorithmic modifications like binarization, as used in YOLO-v2 (76% mAP), and hardware-specific mappings (e.g., FPGA-accelerated versions of YOLO) highlight the potential of lightweight models for energy-efficient implementations. The FPGA implementations typically exhibit superior energy efficiency, especially when paired with low-power platforms like the Xilinx UltraScale+ (60 FPS with 3 FPS/W for object detection tasks). These results suggest that FPGAs, coupled with efficient CNN architectures, can outperform traditional GPU solutions in energy efficiency while maintaining competitive performance metrics.

2) DRIVER STATUS, BEHAVIOR, AND IDENTIFICATION

Tables 6 and 7 illustrate the application of CNNs and RNNs for monitoring driver behavior and status. The effectiveness of these models, with accuracies exceeding 90% for driver behavior and status detection, reflects the capability of CNN-based models in real-time applications. However, the choice of platform significantly influences the overall efficiency. For instance, models implemented on NVIDIA Jetson TK1 for driver status monitoring achieve throughput of 14.9 FPS, but energy efficiency remains a challenge, with only 5.5 FPS/W.

Notably, models implemented on FPGAs, such as the driver identification model on the Xilinx XC7k32T, show promise for improving energy efficiency. Although the throughput and energy efficiency data for some FPGA implementations are not provided, the potential for customized hardware acceleration to deliver energy savings is evident.

3) TRAFFIC SIGN RECOGNITION

Tables 8 and 9 present results for traffic sign detection and recognition tasks, where CNN-based models achieve high accuracy, such as 99.07% for traffic sign recognition using a hardware-friendly quantized CNN on the Xilinx ZC706. The implementation of CNNs on FPGAs, particularly those optimized with techniques like pruning and quantization, leads to substantial improvements in efficiency. The Xilinx ZC706 implementation, for example, reaches an impressive energy efficiency of 5.8 FPS/W, which underscores the advantage of leveraging FPGA's parallel processing capabilities and reconfigurability for low-power applications.

In contrast, embedded GPU-based solutions, while achieving reasonable performance (e.g., 15 FPS on NVIDIA Jetson TX2), generally lag in energy efficiency compared to their FPGA counterparts. This observation further supports the idea that FPGAs are well-suited for traffic sign recognition tasks in low-power, real-time edge applications.

4) ROAD DETECTION AND SCENE PERCEPTION

Tables 10 and 11 demonstrate how CNNs can be applied to road detection and scene perception. Across different platforms, models like the multi-scale object identification system on Google Pixel, and lane detection on NVIDIA Drive PX, achieve notable performance (e.g., 89.32% accuracy for lane detection). However, the highest efficiency gains are again realized with FPGA-based implementations. For example, the road detection model on an ASIC ($0.45mm^2$) achieves 241 FPS at 3000 FPS/W, far surpassing the performance and efficiency of conventional platforms.

FPGAs also demonstrate the ability to handle complex tasks like stereo estimation and 3D scene analysis. In stereo estimation, enhanced R^3SGM with RES-BNN on Stratix V achieves 11.8 FPS/W, a remarkable efficiency improvement

over other platforms, particularly for computationally heavy tasks like stereo vision.

In summary, while CNN architectures such as SqueezeNet, YOLO, and Faster R-CNN perform well across various tasks, the best results in terms of energy efficiency and real-time performance are achieved through the use of FPGAs, optimized with advanced techniques such as pruning, quantization, and hardware-specific mappings. These findings highlight the importance of choosing both the right CNN algorithm and the appropriate hardware platform for achieving high-performance, energy-efficient implementations in real-world applications.

B. FUTURE DIRECTIONS

The field of self-driving cars is rapidly evolving, with deep learning playing a pivotal role in this transformation. Here are some promising areas of research:

- **Improved accuracy and robustness of perception algorithms.** While deep learning algorithms have achieved remarkable results in object detection and recognition, there is still potential for enhancement. Future research could focus on developing algorithms that are more robust to environmental changes such as lighting and weather conditions, thereby enhancing the safety of self-driving cars. Safety is a paramount concern in the development and deployment of autonomous vehicles. Recent surveys indicate that public acceptance of self-driving cars is closely tied to their perceived safety [150], [151]. Therefore, future research should also focus on addressing safety concerns and improving the public's trust in this technology.
- **Multi-modal perception.** Current perception algorithms typically rely on a single sensor modality, such as cameras or LiDAR. However, future algorithms could fuse data from multiple sensor modalities to improve perception accuracy and robustness. For example, an algorithm that combines camera data with LiDAR data could create a more comprehensive and accurate representation of the environment, enhancing the vehicle's ability to navigate complex scenarios.
- **Real-time embedded systems.** While deep learning algorithms can be computationally expensive to run, many recent studies have successfully developed real-time embedded systems that can execute these algorithms on resource-limited devices, such as FPGAs and ASICs. This has enabled the deployment of deep learning algorithms in self-driving cars in real time. However, as the complexity and demands of these algorithms continue to grow, future research will focus on further optimizing these systems to maintain real-time performance while accommodating more advanced algorithms. Additionally, the challenge of achieving real-time performance with minimal energy consumption on these resource-limited devices remains an open area of research.
- **Safe and reliable decision-making.** Once perception algorithms have been developed that can accurately and robustly detect and recognize objects in the environment, the next challenge will be to develop safe and reliable decision-making algorithms. These algorithms will need to be able to make decisions in real time that ensure the safety of the vehicle and its occupants.
- **Federated Learning for Distributed Model Training.** Recent advances in connected and autonomous vehicles (CAVs) have highlighted the potential of distributed learning paradigms, particularly federated learning (FL), to optimize model deployment in memory-constrained devices like FPGAs. FL allows multiple vehicles to collaboratively train a global model without sharing raw data, preserving privacy while reducing communication overhead. This approach is particularly promising for CAVs, as it enables continuous model improvement using real-world driving data from diverse environments. For example, FL has shown promise in collaborative controller design for CAVs, allowing vehicles to learn from each other's experiences while maintaining individual privacy. However, challenges remain in implementing FL for real-time applications in CAVs, including issues of latency, energy efficiency, and model convergence in non-IID data scenarios. Future research should focus on addressing these challenges to fully realize the potential of FL to improve the performance and adaptability of autonomous driving systems while ensuring privacy and efficient resource utilization [152], [153].

The field of self-driving cars is still in its early stages, but the potential benefits are enormous. Deep learning is a powerful tool that can be used to improve the perception, localization, and decision-making capabilities of self-driving cars. As research in this area continues, we can expect to see even more impressive advances in the years to come.

XIII. CONCLUSION

This study presents a systematic analysis of CNN-based machine learning applications in self-driving cars using limited-resource hardware. Specifically tailored towards perception-based tasks, including vehicle and pedestrian identification, driver detection and analysis, road sign and driving lane recognition, scene comprehension, and route planning, this article provides an in-depth comparison of different implementations of CNN algorithms on FPGAs for each application area.

Our results indicate that FPGAs can exhibit up to a threefold improvement in energy efficiency, compared to typical CPUs and GPUs, particularly when techniques such as pruning, quantization, and binarization are applied. However, the performance and energy efficiency vary significantly depending on the specific application and the optimization methods employed. As demonstrated in Tables 4 to 11, certain CNN algorithms—when optimized for FPGA architecture—

exhibit superior performance for particular tasks, particularly in real-time, resource-constrained environments.

Despite these considerable advancements, major challenges remain. Latency in DNN inference is a key obstacle in the adoption of self-driving car technology. This is because real-time decision-making, which is crucial for the safe operation of self-driving cars, requires high-speed processing of sensory data. Typically, a frame rate of 30 FPS or more is desirable to ensure smooth and timely responses to dynamic road situations [74]. Therefore, reducing DNN inference latency is a critical area for future research. Moreover, the lack of interpretability and robustness of deep neural networks poses a significant challenge for ensuring the safety and regulation of autonomous vehicles.

Indeed, FPGA represents the optimal structure for addressing the demands of self-driving car applications due to two compelling reasons. Firstly, the algorithms driving autonomous cars evolve rapidly, whereas designing ASIC-based accelerators can take a year or more. FPGAs, in contrast, offer an adaptive solution, facilitating swift updates. Secondly, the vast range of self-driving car applications makes it difficult to achieve cost-effective scaling of ASIC accelerators soon, resulting in FPGAs being a more pragmatic and cost-efficient alternative. Nevertheless, FPGA is not yet the de facto platform for self-driving car applications due to two key limitations. Firstly, programming FPGAs requires a greater degree of technical expertise than software programming, and secondly, despite the existence of HLS tools, further optimization is required for the generated code to better suit the demands of real-world self-driving car applications [126], [154].

REFERENCES

[1] C. Urmson and W. Whittaker, "Self-driving cars and the urban challenge," *IEEE Intell. Syst.*, vol. 23, no. 2, pp. 66–68, Mar. 2008.

[2] S. Thrun, "Toward robotic cars," *Commun. ACM*, vol. 53, no. 4, pp. 99–106, Apr. 2010.

[3] U. Montanaro, S. Dixit, S. Fallah, M. Dianati, A. Stevens, D. Oxtoby, and A. Mouzakitis, "Towards connected autonomous driving: Review of use-cases," *Vehicle Syst. Dyn.*, vol. 57, no. 6, pp. 779–814, Jun. 2019.

[4] T. Toroyan, "Global status report on road safety," *Injury Prevention*, vol. 15, no. 4, p. 286, Aug. 2009.

[5] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," U.S. Dept. Transp., Washington, DC, USA, Traffic Safety Facts Crash Stats, Rep. DOT HS 812 506, Mar. 2018.

[6] T. Luettel, M. Himmelsbach, and H.-J. Wuensche, "Autonomous ground vehicles—Concepts and a path to the future," *Proc. IEEE*, vol. 100, pp. 1831–1839, May 2012.

[7] W. Payre, J. Cestac, and P. Delhomme, "Intention to use a fully automated car: Attitudes and a priori acceptability," *Transp. Res. F, Traffic Psychol. Behav.*, vol. 27, pp. 252–263, Nov. 2014.

[8] P. E. Ross, "Robot, you can drive my car," *IEEE Spectr.*, vol. 51, no. 6, pp. 60–90, Jun. 2014.

[9] S. Kuutti, S. Fallah, R. Bowden, and P. Barber, *Deep Learning for Autonomous Vehicle Control: Algorithms, State-of-the-Art, and Future Prospects* (Synthesis Lectures on Advances in Automotive Technology). Cham, Switzerland: Springer, 2019.

[10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[12] S. M. Veres, L. Molnar, N. K. Lincoln, and C. P. Morice, "Autonomous vehicle control systems—A review of decision making," *Proc. Inst. Mech. Eng. I, J. Syst. Control Eng.*, vol. 225, no. 2, pp. 155–195, 2011.

[13] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, no. 1, pp. 187–210, 2018.

[14] H. Ye and G. Y. Li, "Deep reinforcement learning for resource allocation in V2V communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[15] H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2584–2601, Oct. 2017.

[16] K. Konda and R. Memisevic, "Learning visual odometry with a convolutional network," in *Proc. VISAPP*, 2015, pp. 486–490.

[17] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, Feb. 2016.

[18] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—Part I: Distributed system architecture and development process," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7131–7140, Dec. 2014.

[19] Z. Wan, B. Yu, T. Y. Li, J. Tang, Y. Zhu, Y. Wang, A. Raychowdhury, and S. Liu, "A survey of FPGA-based robotic computing," *IEEE Circuits Syst. Mag.*, vol. 21, no. 2, pp. 48–74, 2nd Quart., 2021.

[20] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113816.

[21] E. Mitleton-Kelly, I. Deschenaux, C. Maag, M. Fullerton, and N. Celikkaya, "Enhancing crowd evacuation and traffic management through Ami technologies: A review of the literature," in *Co-evolution of Intelligent Socio-Technical Systems*. Berlin, Germany: Springer, 2013, pp. 19–41.

[22] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 534–545, Apr. 2015.

[23] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 4, pp. 6–22, Winter 2014.

[24] A. Ziebinski, R. Cupek, D. Grzechca, and L. Chruszczyk, "Review of advanced driver assistance systems (ADAS)," *AIP Conf. Proc.*, vol. 1906, no. 1, 2017, Art. no. 120002.

[25] J. Borrego-Carazo, D. Castells-Rufas, E. Biempica, and J. Carrabina, "Resource-constrained machine learning for ADAS: A systematic review," *IEEE Access*, vol. 8, pp. 40573–40598, 2020.

[26] T. G. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition*, T. Caelli, A. Amin, R. P. W. Duin, D. de Ridder, and M. Kamel, Eds., Berlin, Germany: Springer, 2002, pp. 15–30.

[27] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," in *Emerging Artificial Intelligence Applications in Computer Engineering*. Amsterdam, The Netherlands: IOS Press, 2007, pp. 3–24.

[28] A. Moujahid, M. E. Tantaoui, M. D. Hina, A. Soukane, A. Ortalda, A. ElKhadimi, and A. Ramdane-Cherif, "Machine learning techniques in ADAS: A review," in *Proc. Int. Conf. Adv. Comput. Commun. Eng. (ICACCE)*, Jun. 2018, pp. 235–242.

[29] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 315–329, Mar. 2020.

[30] J. Li, H. Cheng, H. Guo, and S. Qiu, "Survey on artificial intelligence for vehicles," *Automot. Innov.*, vol. 1, no. 1, pp. 2–14, Jan. 2018.

[31] C. Wang and Z. Luo, "A review of the optimal design of neural networks based on FPGA," *Appl. Sci.*, vol. 12, no. 21, p. 10771, Oct. 2022.

[32] D. Castells-Rufas, V. Ngo, J. Borrego-Carazo, M. Codina, C. Sanchez, D. Gil, and J. Carrabina, "A survey of FPGA-based vision systems for autonomous cars," *IEEE Access*, vol. 10, pp. 132525–132563, 2022.

[33] G. Aishwarya, B. Patil, P. V. Joshi, K. M. Sudarsham, K. Vaidyanathan, P. Parandkar, and A. Dsouza, "A survey on use of FPGA in automotive system," in *Proc. Int. Conf. Distrib. Comput., VLSI, Electr. Circuits Robot. (DISCOVER)*, Oct. 2022, pp. 51–56.
[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[35] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[36] H. Nakahara, H. Yonekawa, T. Fujii, and S. Sato, "A lightweight YOLOv2: A binarized CNN with a parallel support vector regression for an FPGA," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2018, pp. 31–40.
[37] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping CNN onto embedded FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 35–47, Jan. 2018.
[38] M. M. Madebo, C. M. Abdissa, L. N. Lemma, and D. S. Negash, "Robust tracking control for quadrotor UAV with external disturbances and uncertainties using neural network based MRAC," *IEEE Access*, vol. 12, pp. 36183–36201, 2024.
[39] W. Ayalew, M. Menebo, C. Merga, and L. Negash, "Optimal path planning using bidirectional rapidly-exploring random tree star-dynamic window approach (BRRT*-DWA) with adaptive Monte Carlo localization (AMCL) for mobile robot," *Eng. Res. Exp.*, vol. 6, no. 3, Jul. 2024, Art. no. 035212, doi: 10.1088/2631-8695/ad61bd.
[40] W. Ayalew, M. Menebo, L. Negash, and C. M. Abdissa, "Solving optimal path planning problem of an intelligent mobile robot in dynamic environment using bidirectional rapidly-exploring random tree star-dynamic window approach (BRRT*-DWA) with adaptive Monte Carlo localization (AMCL)," *TechRxiv*, Dec. 2023. [Online]. Available: h..p://dx.doi.org/10.36227/techrxiv.24623784.v1
[41] Y. Adgo, L. Negash, and C. M. Abdissa, "Enhancing trajectory tracking performance of wheeled mobile robot using backstepping fuzzy sliding mode control," *Eng. Res. Exp.*, vol. 6, no. 4, Oct. 2024, Art. no. 045204. [Online]. Available: https://dx.doi.org/10.1088/2631-8695/ad79b9
[42] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "[DL] a survey of FPGA-based neural network inference accelerators," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 1, pp. 1–26, Mar. 2019.
[43] C. Silvano, D. Ielmini, F. Ferrandi, L. Fiorin, S. Curzel, L. Benini, F. Conti, A. Garofalo, C. Zambelli, E. Calore, S. F. Schifano, M. Palesi, G. Ascia, D. Patti, N. Petra, D. D. Caro, L. Lavagno, T. Urso, V. Cardellini, G. C. Cardarilli, R. Birke, and S. Perri, "A survey on deep learning hardware accelerators for heterogeneous HPC platforms," 2024, *arXiv:2306.15552*.
[44] *NVIDIA A100 TENSOR CORE GPU Data Sheet*, NVIDIA, Santa Clara, CA, USA, Jan. 2021.
[45] *NVIDIA Tesla V100 Data Sheet*, NVIDIA, Santa Clara, CA, USA, Mar. 2018.
[46] *NVIDIA Jetson Xavier NX Series System-on-Module Data Sheet*, NVIDIA, Santa Clara, CA, USA, Jan. 2024.
[47] *NVIDIA Jetson TX2 NX System-on-Module Data Sheet*, NVIDIA, Santa Clara, CA, USA, Jan. 2024.
[48] *NVIDIA Jetson TX2 Series System-on-Module Data Sheet*, NVIDIA, Santa Clara, CA, USA, Apr. 2022.
[49] *NVIDIA Jetson TX1 Series System-on-Module Data Sheet*, NVIDIA, Santa Clara, CA, USA, Aug. 2015.
[50] *UltraScale Architecture and Product Data Sheet: Overview*, Xilinx, San Jose, CA, USA, Mar. 2024.
[51] *Versal Architecture and Product Data Sheet: Overview*, AMD, Santa Clara, CA, USA, Jan. 2024.
[52] *Stratix 10 GX/SX Device Overview*, Intel, Santa Clara, CA, USA, Jun. 2024.
[53] *NVIDIA Tegra K1 Series Processors Data Sheet*, NVIDIA, Santa Clara, CA, USA, Dec. 2019.
[54] *Zynq UltraScale+ MPSoC Data Sheet: Overview*, Xilinx, San Jose, CA, USA, Nov. 2022.
[55] *Stratix V Device Handbook*, Intel, Santa Clara, CA, USA, Dec. 2011.
[56] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[57] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 39, no. 6, pp.1137–1149, Jun. 2017. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2577031
[58] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision—ECCV 2016*. Cham, Switzerland: Springer, Oct. 2016, pp. 21–37.
[59] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[60] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.
[61] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.
[62] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[63] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.
[64] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional Siamese networks for object tracking," in *Computer Vision—ECCV 2016*. Cham, Switzerland: Springer, Oct. 2016, pp. 850–865.
[65] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[67] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.
[68] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5410–5418.
[69] N. Radwan, A. Valada, and W. Burgard, "VLocNet++: Deep multitask learning for semantic visual localization and odometry," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4407–4414, Oct. 2018.
[70] N. Tamrakar, S. Karki, M. Y. Kang, N. C. Deb, E. Arulmozhi, D. Y. Kang, J. Kook, and H. T. Kim, "Lightweight improved YOLOv5s-CGhostnet for detection of strawberry maturity levels and counting," *AgriEngineering*, vol. 6, no. 2, pp. 962–978, Apr. 2024.
[71] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, Apr. 2016.
[72] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5695–5703.
[73] D. ZiWen and Y. Dong, "Multi-objective neural architecture search for efficient and fast semantic segmentation on edge," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 1346–1357, Jan. 2024.
[74] Z. Lu, R. Cheng, S. Huang, H. Zhang, C. Qiu, and F. Yang, "Surrogate-assisted multiobjective neural architecture search for real-time semantic segmentation," *IEEE Trans. Artif. Intell.*, vol. 4, no. 6, pp. 1602–1615, Dec. 2023.
[75] J. Wu, H. Kuang, Q. Lu, Z. Lin, Q. Shi, X. Liu, and X. Zhu, "M-FasterSeg: An efficient semantic segmentation network based on neural architecture search," *Eng. Appl. Artif. Intell.*, vol. 113, Aug. 2022, Art. no. 104962.
[76] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.
[77] R. Verbickas, R. Laganiere, D. Laroche, C. Zhu, X. Xu, and A. Ors, "SqueezeMap: Fast pedestrian detection on a low-power automotive processor using efficient convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 146–154.
[78] D. Tome, L. Bondi, L. Baroffio, S. Tubaro, E. Plebani, and D. Pau, "Reduced memory region based deep convolutional neural network detection," in *Proc. IEEE 6th Int. Conf. Consum. Electron.*, Sep. 2016, pp. 15–19.
[79] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," *Signal Process., Image Commun.*, vol. 47, pp. 482–489, Sep. 2016.
[80] A. Kozlov and D. Osokin, "Development of real-time ADAS object detector for deployment on CPU," in *Intelligent Systems and Applications*, Y. Bi, R. Bhaa, and S. Kapoor, Eds. Cham, Switzerland: Springer, 2019, pp. 740–750.

[81] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, "DSOD: Learning deeply supervised object detectors from scratch," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1919–1927.

[82] B. Wu, A. Wan, F. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 129–137.

[83] *S32V234 Data Sheet*, NXP Semiconductors, Eindhoven, The Netherlands, 2022.

[84] B. Kim, Y. Jeon, H. Park, D. Han, and Y. Baek, "Design and implementation of the vehicular camera system using deep neural network compression," in *Proc. 1st Int. Workshop Deep Learn. Mobile Syst. Appl.*, Jun. 2017, pp. 25–30.

[85] F.-A. Chang, C.-C. Tsai, C.-K. Tseng, and J.-I. Guo, "Embedded multiple object detection based on deep learning technique for advanced driver assistance system," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 172–175.

[86] J. Xu, Y. Nie, P. Wang, and A. M. López, "Training a binary weight object detector by knowledge transfer for autonomous driving," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 2379–2384.

[87] G. Brilli, P. Burgio, and M. Bertogna, "Convolutional neural networks on embedded automotive platforms: A qualitative comparison," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2018, pp. 496–499.

[88] D. Wang, K. Xu, and D. Jiang, "PipeCNN: An OpenCL-based open-source FPGA accelerator for convolution neural networks," in *Proc. Int. Conf. Field Program. Technol. (ICFPT)*, Dec. 2017, pp. 279–282.

[89] J. Chen, J. Zhu, R. Xu, Y. Chen, H. Zeng, and J. Huang, "ORNet: Orthogonal re-parameterized networks for fast pedestrian and vehicle detection," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 2662–2674, Jan. 2024.

[90] G. Borghi, R. Gasparini, R. Vezzani, and R. Cucchiara, "Embedded recurrent network for head pose estimation in car," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1503–1508.

[91] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random forests for real time 3D face analysis," *Int. J. Comput. Vis.*, vol. 101, no. 3, pp. 437–458, Feb. 2013.

[92] M. Venturelli, G. Borghi, R. Vezzani, and R. Cucchiara, "From depth data to head pose estimation: A Siamese approach," 2017, *arXiv:1703.03624*.

[93] A. Saeed and A. Al-Hamadi, "Boosted human head pose estimation using Kinect camera," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 1752–1756.

[94] Z. Xia, W. Zhang, F. Tan, X. Feng, and A. Hadid, "An accurate eye localization approach for smart embedded system," in *Proc. 6th Int. Conf. Image Process. Theory, Tools Appl. (IPTA)*, Dec. 2016, pp. 1–5.

[95] D. Tran, H. M. Do, W. Sheng, H. Bai, and G. Chowdhary, "Real-time detection of distracted driving based on deep learning," *IET Intell. Transp. Syst.*, vol. 12, no. 10, pp. 1210–1219, Dec. 2018.

[96] B. Reddy, Y.-H. Kim, S. Yun, C. Seo, and J. Jang, "Real-time driver drowsiness detection for embedded system using model compression of deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 121–128.

[97] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[98] R. Ahmadian, M. Ghatee, and J. Wahlström, "Discrete wavelet transform for generative adversarial network to identify drivers using gyroscope and accelerometer sensors," *IEEE Sensors J.*, vol. 22, no. 7, pp. 6879–6886, Apr. 2022.

[99] A. Samir, B. A. Mohamed, and B. A. Abdelhakim, "Detection of driver drowsiness based on the Viola & Jones method and logistic regression analysis," in *Proc. Medit. Symp. Smart City Appl.*, Oct. 2017, pp. 1–6.

[100] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE Multimedia Mag.*, vol. 19, no. 2, pp. 4–10, Feb. 2012.

[101] I. del Campo, R. Finker, M. V. Martinez, J. Echanobe, and F. Doctor, "A real-time driver identification system based on artificial neural networks and cepstral analysis," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 1848–1855.

[102] X. Xu, S. Yin, and P. Ouyang, "Fast and low-power behavior analysis on vehicles using smartphones," in *Proc. 6th Int. Symp. Next Gener. Electron. (ISNE)*, May 2017, pp. 1–4.

[103] M. García-García, A. Caplier, and M. Rombaut, "Sleep deprivation detection for real-time driver monitoring using deep learning," in *Image Analysis and Recognition*. Cham, Switzerland: Springer, Jun. 2018, pp. 435–442.

[104] R. Alvarez, R. Prabhavalkar, and A. Bakhtin, "On the efficient representation and execution of deep acoustic models," 2016, *arXiv:1607.04683*.

[105] *TensorFlow Lite*. Accessed: Dec. 1, 2019. [Online]. Available: https://www.tensorflow.org/lite

[106] D. Yudin and D. Slavioglo, "Usage of fully convolutional network with clustering for traffic light detection," in *Proc. 7th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2018, pp. 1–6.

[107] M. Ester, H.-P. Kriegel, S. Jorg, and X. Xu, "A density-based clustering algorithms for discovering clusters," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, vol. 96, 1996, pp. 226–231.

[108] K. Yi, Z. Jian, S. Chen, and N. Zheng, "Feature selective small object detection via knowledge-based recurrent attentive neural network," 2018, *arXiv:1803.05263*.

[109] S. Jagannathan, K. Desappan, P. Swami, M. Mathew, S. Nagori, K. Chitnis, Y. Marathe, D. Poddar, S. Narayanan, and A. Jain, "Efficient object detection and classification on low power embedded systems," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2017, pp. 233–234.

[110] H. Novais and A. R. Fernandes, "Community based repository for georeferenced traffic signs," in *Proc. Encontro Português de Computação Gráfica e Interação (EPCGI)*, Oct. 2017, pp. 1–8.

[111] H. S. Lee and K. Kim, "Simultaneous traffic sign detection and boundary estimation using convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1652–1663, May 2018.

[112] Y. Zhou, Z. Chen, and X. Huang, "A system-on-chip FPGA design for real-time traffic signal recognition system," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 1778–1781.

[113] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham, Switzerland: Springer, Oct. 2015, pp. 234–241.

[114] Z. Lin, M. Yih, J. M. Ota, J. D. Owens, and P. Muyan-Özçelik, "Benchmarking deep learning frameworks and investigating FPGA deployment for traffic sign classification and detection," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 3, pp. 385–395, Sep. 2019.

[115] B. B. Shabarinath and P. Muralidhar, "Convolutional neural network based traffic-sign classifier optimized for edge inference," in *Proc. IEEE REGION 10 Conf. (TENCON)*, Nov. 2020, pp. 420–425.

[116] J. Kim, J.-K. Kang, and Y. Kim, "A low-cost fully integer-based CNN accelerator on FPGA for real-time traffic sign recognition," *IEEE Access*, vol. 10, pp. 84626–84634, 2022.

[117] J. M. Á. Alvarez and A. M. Lopez, "Road detection based on illuminant invariance," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 184–193, Mar. 2011.

[118] M. Mody, D. Kumar, P. Swami, M. Mathew, and S. Nagori, "Low cost and power CNN/deep learning solution for automated driving," in *Proc. 19th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2018, pp. 432–436.

[119] Y. Gu, Q. Wang, and S. Kamijo, "Intelligent driving data recorder in smartphone using deep neural network-based speedometer and scene understanding," *IEEE Sensors J.*, vol. 19, no. 1, pp. 287–296, Jan. 2019.

[120] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, "DeepLanes: End-to-end lane position estimation using deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 38–45.

[121] M. Oeljeklaus, F. Hoffmann, and T. Bertram, "A fast multi-task CNN for spatial understanding of traffic scenes," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2825–2830.

[122] Y. Zhou, Y. Lyu, and X. Huang, "RoadNet: An 80-mW hardware accelerator for road detection," *IEEE Embedded Syst. Lett.*, vol. 11, no. 1, pp. 21–24, Mar. 2019.

[123] X. Wang, K. Huang, and A. Knoll, "Performance optimisation of parallelized ADAS applications in FPGA-GPU heterogeneous systems: A case study with lane detection," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 4, pp. 519–531, Dec. 2019.

[124] S. Nousias, E.-V. Pikoulis, C. Mavrokefalidis, A. S. Lalos, and K. Moustakas, "Accelerating 3D scene analysis for autonomous driving on embedded AI computing platforms," in *Proc. IFIP/IEEE 29th Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2021, pp. 1–6.

[125] S. Nousias, E.-V. Pikoulis, C. Mavrokefalidis, and A. S. Lalos, "Accelerating deep neural networks for efficient scene understanding in multi-modal automotive applications," *IEEE Access*, vol. 11, pp. 28208–28221, 2023.

[126] A. Hosseiny and H. Jahanirad, "Hardware acceleration of YOLOv7-tiny using high-level synthesis tools," *J. Real-Time Image Process.*, vol. 20, no. 4, p. 75, Aug. 2023.

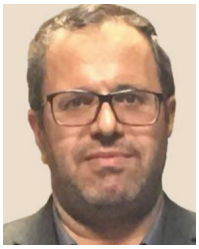
[127] G. Chen, Y. Ling, T. He, H. Meng, S. He, Y. Zhang, and K. Huang, "StereoEngine: An FPGA-based accelerator for real-time high-quality stereo estimation with binary neural network," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 4179–4190, Nov. 2020.
[128] Y. Ling, T. He, Y. Zhang, H. Meng, K. Huang, and G. Chen, "Lite-stereo: A resource-efficient hardware accelerator for real-time high-quality stereo estimation using binary neural network," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5357–5366, Dec. 2022.
[129] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation," in *Proc. Int. Conf. Embedded Comput. Syst., Architectures, Modeling Simulation*, Jul. 2010, pp. 93–101.
[130] O. Rahnama, T. Cavalleri, S. Golodetz, S. Walker, and P. Torr, "R3SGM: Real-time raster-respecting semi-global matching for power-constrained systems," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2018, pp. 102–109.
[131] Q. Gautier, A. Shearer, J. Matai, D. Richmond, P. Meng, and R. Kastner, "Real-time 3D reconstruction for FPGAs: A case study for evaluating the performance, area, and programmability trade-offs of the Altera OpenCL SDK," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2014, pp. 326–329.
[132] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 3562–3568.
[133] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[134] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. 18th Nat. Conf. Artif. Intell.* Washington, DC, USA: American Association for Artificial Intelligence, 2002, pp. 593–598.
[135] M. Gu, K. Guo, W. Wang, Y. Wang, and H. Yang, "An FPGA-based real-time simultaneous localization and mapping system," in *Proc. Int. Conf. Field Program. Technol. (FPT)*, Dec. 2015, pp. 200–203.
[136] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1449–1456.
[137] V. Bonato, E. Marques, and G. A. Constantinides, "A floating-point extended Kalman filter implementation for autonomous mobile robots," *J. Signal Process. Syst.*, vol. 56, no. 1, pp. 41–50, Jul. 2009.
[138] W. Fang, Y. Zhang, B. Yu, and S. Liu, "FPGA-based ORB feature extraction for real-time visual SLAM," in *Proc. Int. Conf. Field Program. Technol. (ICFPT)*, Dec. 2017, pp. 275–278.
[139] R. Liu, J. Yang, Y. Chen, and W. Zhao, "ESLAM: An energy-efficient accelerator for real-time ORB-SLAM on FPGA platform," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 1–6.
[140] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 224–236.
[141] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 118–126.
[142] F. Radenovic, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1655–1668, Jul. 2019.
[143] Z. Xu, J. Yu, C. Yu, H. Shen, Y. Wang, and H. Yang, "CNN-based feature-point extraction for real-time visual SLAM on embedded FPGA," in *Proc. IEEE 28th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, May 2020, pp. 33–37.
[144] S. Liang, X. Ning, J. Yu, K. Guo, T. Lu, C. Tang, S. Zeng, Y. Wang, D. Yang, and H. Yang, "Efficient computing platform design for autonomous driving systems," in *Proc. 26th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2021, pp. 734–741.
[145] J. Yu, F. Gao, J. Cao, C. Yu, Z. Zhang, Z. Huang, Y. Wang, and H. Yang, "CNN-based monocular decentralized SLAM on embedded FPGA," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2020, pp. 66–73.
[146] J. Yu, Z. Xu, S. Zeng, C. Yu, J. Qiu, C. Shen, Y. Xu, G. Dai, Y. Wang, and H. Yang, "INCA: Interruptible CNN accelerator for multi-tasking in embedded robots," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[147] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2018, pp. 340–349.
[148] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1437–1451, Jun. 2018.
[149] J. Yu, G. Ge, Y. Hu, X. Ning, J. Qiu, K. Guo, Y. Wang, and H. Yang, "Instruction driven cross-layer CNN accelerator for fast detection on FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 1–23, Sep. 2018.
[150] A. Chaudhry, P. Liu, A. Hussain, and I. Sanaullah, "Safety perceptions of self-driving cars: A survey study in China and Pakistan," in *Advances in Human Aspects of Transportation*, N. Stanton, Ed., Cham, Switzerland: Springer, 2019, pp. 458–468.
[151] K. Othman, "Public acceptance and perception of autonomous vehicles: A comprehensive review," *AI Ethics*, vol. 1, no. 3, pp. 355–387, Aug. 2021.
[152] V. P. Chellapandi, L. Yuan, C. G. Brinton, S. H. Żak, and Z. Wang, "Federated learning for connected and automated vehicles: A survey of existing approaches and challenges," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 119–137, Jan. 2024.
[153] T. Zeng, O. Semiari, M. Chen, W. Saad, and M. Bennis, "Federated learning for collaborative controller design of connected and autonomous vehicles," in *Proc. 60th IEEE Conf. Decis. Control (CDC)*, Dec. 2021, pp. 5033–5038.
[154] A. Bernardi, G. Brilli, A. Capotondi, A. Marongiu, and P. Burgio, "An FPGA overlay for efficient real-time localization in 1/10th scale autonomous vehicles," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 915–920.



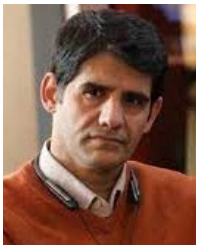
MOHAMMAD CHESHFAR received the M.S. degree from Shahid Rajaee Teacher Training University, Tehran, Iran, in 2013, where he is currently pursuing the Ph.D. degree with the Faculty of Electrical Engineering. He is a Visiting Research Scholar with the Department of Electronics and Telecommunications, Politecnico di Torino, Italy, under the supervision of Prof. Luciano Lavagno and Mihai Lazaresco. His research interests include low-power high-performance computing, hardware accelerators for deep learning inference, and machine learning for electronic design automation.



MOHAMMAD HOSSEIN MAGHAMI was born in Mashhad, Iran, in 1984. He received the B.Sc. degree in electrical engineering from the Ferdowsi University of Mashhad, Mashhad, in 2006, the M.Sc. degree in electrical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2009, and the Ph.D. degree in electrical engineering from the K. N. Toosi University of Technology, Tehran, in 2015. He carried out part of his Ph.D. research work at Polytechnique Montréal as a Visiting Research Scholar. Since September 2016, he has been with Shahid Rajaee Teacher Training University, Tehran, as an Assistant Professor. His main areas of research interests include implantable biomedical microsystems, high-speed low-power A/D converters, and mixed-mode integrated circuits.



PARVIZ AMIRI received the B.Sc. degree in electrical engineering (electronic) from the University of Mazandaran, in 1994, the M.Sc. degree in electrical engineering (electronic) from the K. N. Toosi University of Technology, Tehran, Iran, in 1997, and the Ph.D. degree in electrical engineering (electronic) from Tarbiat Modares University, Tehran, in 2010. He is currently an Associate Professor with the Department of Electrical Engineering, Shahid Rajaee Teacher Training University, Tehran. His main research interest includes analog and digital integrated circuit design. His primary research interests include RF and power electronic circuits, with a focus on highly efficient and highly linear power circuit design.



HOSSEIN GHARAEE GARAKANI received the B.Sc. degree in electrical engineering from the Khajeh Nasir Toosi University of Technology (KNTU), in 1998, and the M.Sc. and Ph.D. degrees in electrical engineering from Tarbiat Modares University, Tehran, Iran, in 2000 and 2009, respectively. Since 2009, he has been with the Department of Information Technology, ICT Research Institute. His research interests include the general area of VLSI, with an emphasis on basic logic circuits for low-voltage low-power applications, digital circuit design (FPGA), DSP algorithms, crypto chips, and intrusion detection and prevention systems.



LUCIANO LAVAGNO (Life Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from UC Berkeley, in 1992. He was an Architect with the POLIS HW/SW co-design tool. Since 1993, he has been a Professor with Politecnico di Torino, Italy. From 2003 to 2014, he was an Architect with the Cadence CtoSilicon high-level synthesis tool. He co-authored four books and over 200 scientific articles. His research interests include the synthesis of asynchronous circuits, HW/SW co-design, high-level synthesis, and design tools for wireless sensor networks.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement