

Review

Low-Latency Edge Computing Architectures for Real-Time Vehicle Warning Systems: A Review

Redeemer Kwei Amartey^{1,2} and Duan Zhao^{1,2,*}¹ Mining IoT Research Center, China University of Mining and Technology, Xuzhou 221116, China; fs25060002e@cumt.edu.cn² School of Control Science and Engineering, China University of Mining and Technology, Xuzhou 221116, China

* Correspondence: duan.zhao@cumt.edu.cn

Abstract

Real-time vehicle warning systems are critical for collision prevention, yet they face stringent sub-10 ms latency requirements under severe energy and computational constraints. This review systematically surveys low-latency edge computing architectures for such systems, explicitly comparing CPU-based, GPU-accelerated, FPGA-based, ASIC/NPU-embedded, fog, and cloud-only processing paradigms. We examine edge intelligence frameworks for intelligent transportation systems, the computational demands of collision avoidance algorithms, V2X communication protocols, and hardware accelerators. A key contribution is a comparative analysis of latency, power consumption, and area trade-offs, revealing that FPGA accelerators achieve deterministic sub-millisecond processing at 5–15 W, while emerging NPUs offer 1–5 W alternatives for fixed-function inference. A critical synthesis of the literature identifies major gaps: the absence of standardized benchmarks, insufficient field-testing of FPGA prototypes, and underutilized potential of approximate computing in safety loops. Furthermore, we introduce fog computing as a vital intermediary layer to bridge edge-cloud gaps. This review consolidates over 58 core studies and offers practical, actionable insights for designing next-generation vehicular safety systems.

Keywords: edge computing; low latency; vehicle warning systems; FPGA accelerators; V2X communication; real-time processing; intelligent transportation systems; fog computing

1. Introduction

Modern vehicles increasingly rely on advanced driver assistance systems (ADASs) and autonomous driving capabilities that demand real-time perception, decision making, and warning dissemination [1,2]. At the heart of these systems lies the vehicle warning system, a collection of algorithms and hardware that detects imminent collisions, road hazards, or unsafe driving conditions and then alerts the driver or triggers automatic interventions, such as braking or steering [3,4]. The effectiveness of these warnings depends directly on end-to-end latency, which is the time from sensor data acquisition to warning activation. Research indicates that this latency must stay below 10 to 100 milliseconds to avoid accidents [5,6].

Traditional cloud-centric processing introduces delays that are simply too long for safety-critical applications due to network transmission time and backhaul congestion [7]. Edge computing, which processes data at or near the source such as vehicles or roadside units, has emerged as a key technology for enabling low-latency vehicle warning



Academic Editor: Kewei Sha

Received: 11 June 2026

Revised: 10 July 2026

Accepted: 17 July 2026

Published: 24 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

systems [8,9]. By moving computation to edge nodes, latency can be reduced to milliseconds, and bandwidth usage decreases substantially [10,11].

However, edge nodes face constraints in energy, computing power, and memory [12]. Designing low-latency edge architectures, therefore, requires careful selection of hardware accelerators including field-programmable gate arrays (FPGAs), graphics processing units (GPUs), and neural processing units (NPUs), along with communication protocols, such as vehicle-to-everything (V2X) and 5G, and appropriate software frameworks [13,14]. This review aims to synthesize the current state of research and identify promising directions for future work.

1.1. Scope and Contribution

This review focuses on edge computing architectures for real-time vehicle warning systems and covers the following areas:

- Edge intelligence frameworks for intelligent transportation systems (ITSs);
- Collision avoidance algorithms and their computational requirements;
- V2X communication and edge offloading for low-latency warning dissemination;
- Hardware accelerators including FPGAs, adders, and approximate computing for edge artificial intelligence (AI);
- Comparative analysis of latency, power, and area trade-offs across CPU, GPU, FPGA, NPU/ASIC, fog, and cloud;
- Research gaps and future directions.

We have consolidated more than 58 core references from journals, conferences, and technical reports to provide a structured reference for engineers and researchers working on real-time vehicle safety systems.

1.2. Review Methodology

To ensure a rigorous and reproducible survey, we conducted a structured literature search across three major digital databases: IEEE Xplore, Scopus, and Web of Science. The search period covered publications from January 2018 to June 2026. The primary search strings included combinations of (“edge computing” OR “fog computing” OR “hardware acceleration” OR “FPGA” OR “GPU” OR “NPU”) AND (“vehicle warning” OR “collision avoidance” OR “V2X” OR “ADAS” OR “autonomous driving”) AND (“latency” OR “real-time” OR “low-latency”).

Inclusion Criteria:

- Peer-reviewed journal articles, conference proceedings, or authoritative technical reports.
- Explicit focus on latency constraints below 100 ms for vehicular applications.
- Quantitative reporting of processing latency, power consumption, or hardware resource utilization.
- Applicability to terrestrial vehicular environments (excluding aerial/underwater robotics).
- Publication date between January 2018 and June 2026.

Exclusion Criteria:

- Papers solely focused on pure cloud solutions without edge or fog consideration.
- Non-vehicular robotics or general IoT applications without clear transportation relevance.
- Papers lacking experimental validation or quantitative performance metrics.
- Pre-print servers (arXiv, TechRxiv) without peer review, unless they had subsequent journal publications.

Shortlisting Process:

Our initial search yielded approximately 150 potentially relevant papers. After abstract screening, 95 papers were retained for full-text review. Following full-text assessment

against the inclusion and exclusion criteria, 58 core papers were shortlisted for in-depth analysis and synthesis. This review consolidates findings from these 58 core studies, supplemented by 12 additional foundational works, to provide a balanced critical assessment.

1.3. Research Questions

To provide a focused and rigorous review, this study is guided by the following four research questions (RQs):

- RQ1: What are the key architectural frameworks, communication protocols, and hardware accelerators currently proposed for low-latency edge computing in vehicle warning systems?
- RQ2: What are the computational demands of state-of-the-art collision avoidance algorithms, and which hardware platforms (CPU, GPU, FPGA, ASIC/NPU, fog) best meet the sub-10 ms latency requirement?
- RQ3: What are the power, latency, and reconfigurability trade-offs among different edge acceleration technologies, and how do these trade-offs impact real-world vehicular deployment?
- RQ4: What are the critical research gaps and open challenges that must be addressed to enable widespread adoption of edge-based vehicle warning systems?

These RQs structure the review, ensuring that each section directly contributes to answering a specific, well-defined inquiry. The synthesis of findings across these RQs is presented in Sections 6 and 7.

1.4. Motivation and Industrial Relevance

The urgency of low-latency edge architectures is underscored by rapid market growth. The automotive edge AI accelerator market was valued at 2.1 billion USD in 2024, with a compound annual growth rate of 22.9% projected through 2034, driven by demand for real-time ADASs and autonomous driving features [15]. Furthermore, regulatory bodies in the EU (GSR regulations mandating intelligent speed assistance and lane-keeping systems by 2024) and the US (NHTSA's proposed rulemaking for automatic emergency braking) are accelerating deployment timelines. This review responds to both technological and regulatory drivers by providing a structured reference for engineers designing safety-critical, low-latency edge solutions for next-generation vehicle warning systems.

2. Edge Intelligence for Intelligent Transportation Systems

Edge intelligence refers to the deployment of artificial intelligence models and data processing on edge devices located close to data sources [1]. In the context of ITS, edge intelligence enables real-time object detection, path planning, and warning generation without relying on distant cloud servers [2,8,16].

2.1. Architectures and Frameworks

Several edge intelligence architectures have been proposed for ITS. A comprehensive survey of distributed edge intelligence for transport systems has been presented, covering hierarchical edge-cloud collaboration, task offloading, and resource management. Latency-sensitive tasks, such as collision detection, are recommended to run on the vehicle or roadside edge, while less urgent tasks, like traffic flow analytics, can be offloaded to the cloud [1].

A concrete example is EdgeDriver [6], a multi-large language model framework that operates in cloud-edge environments for autonomous driving assistance. EdgeDriver achieves sub-100 ms inference by partitioning model layers between edge and cloud, demonstrating that hybrid architectures can work in practice. Beyond EdgeDriver, other

frameworks such as AutoEdge [17] and EdgeLens [18] have emerged, focusing on automated model partitioning and real-time video analytics, respectively. These frameworks explicitly address the dynamic nature of vehicular environments, adapting to varying network conditions and computational loads.

VegaEdge [16] presents an edge AI confluence framework specifically for highway safety, enabling real-time vehicle detection, trajectory forecasting, and anomalous driving behavior identification on edge IoT devices. Similarly, a deep learning-based car accident detection framework utilizing both edge and cloud computing [19] demonstrates low-latency accident detection at the edge node with minimal network usage compared to cloud-only deployment.

A volume on driving green transportation systems through artificial intelligence and automation has been edited, which includes chapters on edge-based vehicle modules and heterogeneous networks [2]. Edge AI in vehicle modules using heterogeneous networks has been specifically examined, emphasizing the importance of hardware acceleration for real-time ADAS functions [2,3].

2.2. Information Dissemination in Vehicle-to-Everything

Reliable dissemination of warning information is critical for safety. Efficient information dissemination in V2X environments can be achieved using edge nodes as relays to reduce broadcast storms and latency, with simulations showing a 40 percent reduction in end-to-end delay compared to pure cloud-based approaches [4]. Further improvements in warning propagation delays in dense urban intersections have been demonstrated through edge-assisted geocasting and context-aware forwarding strategies [20,21].

2.3. High-Level Architecture

Figure 1 illustrates a high-level architecture of an edge-based vehicle warning system, showing the path from sensors such as cameras, radar, and LiDAR through edge processing and V2X communication to warning actuation, whereas Figure 2 shows a typical edge intelligence framework for ITS expanded to four tiers: vehicle edge, roadside edge, fog, and cloud [22].

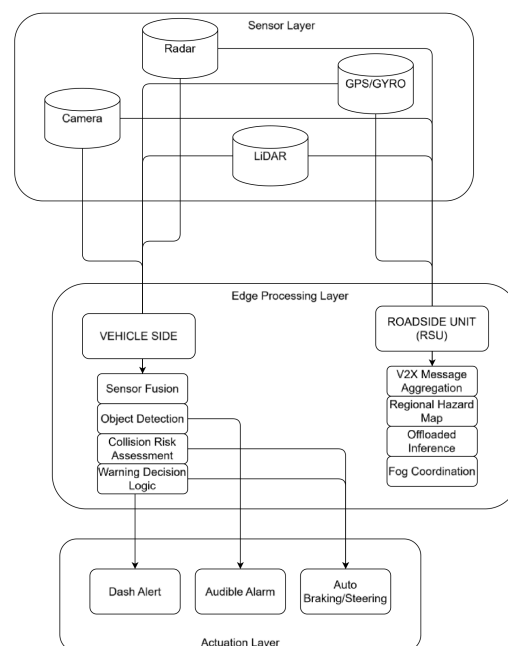


Figure 1. High-level architecture of edge-based vehicle warning system showing three layers: sensor, edge processing, and actuation. Solid arrows show data flow.

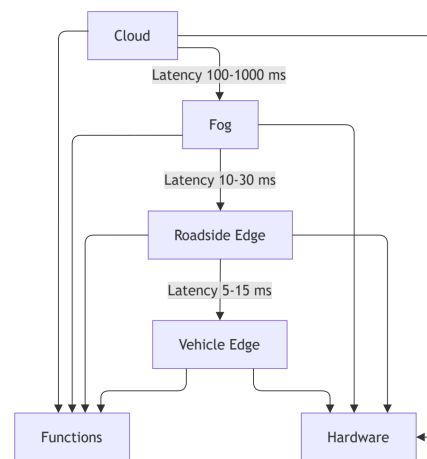


Figure 2. Four-tier edge-fog-cloud intelligence framework for ITS.

3. Collision Avoidance Algorithms and Their Computational Demands

The algorithmic layer of vehicle warning systems determines what computations must be accelerated at the edge. Several surveys have catalogued collision avoidance methods [5,7,9].

3.1. Survey and Analytical Classification of Collision Avoidance Algorithms

Collision avoidance algorithms form the computational core of vehicle warning systems. Rather than merely cataloging them, we analyze how each algorithmic class contributes to the warning pipeline: sensing → perception → decision → actuation.

From a systematic review of autonomous vehicle collision avoidance algorithms, we categorized them into sensor-based methods (radar, LiDAR, camera), path planning (A*, RRT, DWA), decision making (rule-based, reinforcement learning), and deep learning-based perception (CNNs, RNNs, DRL) [5].

In the perception stage, convolutional neural networks (CNNs) perform object detection, lane marking, and traffic sign recognition [23–27]. The detected objects (vehicles, pedestrians, obstacles) are then fed into a time-to-collision (TTC) computation module, which calculates the estimated impact time. If TTC falls below a safety threshold (e.g., 2–3 s), the warning decision logic triggers either a driver alert (dashboard, audible) or automatic intervention (braking, steering) [8,9].

Recent advancements have focused on edge-optimized CNN variants that balance accuracy and inference speed. While earlier models like YOLOv5, SSD, and Faster R-CNN achieved reasonable performance [5,7], the state of the art has shifted toward more efficient architectures specifically designed for resource-constrained edge devices. YOLOv11-nano/-small/-medium [28] and the recently released YOLOv26 [29] incorporate architectural improvements, such as cross-stage partial connections, attention mechanisms, and quantization-aware training, enabling real-time inference at 30–60 frames per second (16–33 ms) on embedded AI platforms (e.g., Raspberry Pi with Coral TPU, NVIDIA Jetson Nano) [30]. Research on enhancing YOLOv5 for autonomous driving [24] proposes lightweight models with channel attention strategies for edge deployment, while YOLO-RECP [25] presents a YOLOv3-powered edge computing technique specifically for rear-end collision prediction. Multi-modal vision approaches extending YOLOv8 with RGB, LiDAR, and grayscale fusion strategies have also been demonstrated for edge AI deployment [26], and lightweight traffic sign detection models based on improved YOLOv8s have been validated for edge deployment under complex environments [27].

An FPGA implementation using Sobel edge detection and Manhattan distance metrics, achieving 5.2 ms per frame, demonstrates that lightweight, deterministic algorithms remain

viable for FPGA-based warning systems [8]. However, for high-accuracy perception in complex urban scenes, deep learning-based methods are increasingly preferred, provided they are accelerated by appropriate edge hardware [31,32].

Also, deep learning techniques for accident avoidance, including real-time object detection and driver behavior monitoring, note that inference latency must fall below 50 ms to be effective at highway speeds [7]. This reinforces the need for hardware acceleration and edge-optimized model variants.

3.2. Computational Requirements for Real-Time Operation

Table 1 summarizes typical latencies for different algorithm classes and indicates whether hardware acceleration is needed. Rule-based methods like safety distance calculations run quickly on CPUs, typically completing in under 5 ms excluding sensor acquisition and communication time. Earlier CNN object detection methods such as YOLOv5-large, SSD, and Faster R-CNN require 50 to 200 ms on CPUs and absolutely need GPU or FPGA acceleration for real-time operation. However, edge-optimized variants (YOLOv11-nano/small, YOLOv26) achieve 10–30 ms on NPU/FPGA-accelerated edge platforms, making them highly suitable for sub-50 ms warning systems [23–30]. Path planning algorithms like A*, RRT, and DWA fall in the middle range of 10 to 50 ms and benefit moderately from DSP hardware. Deep reinforcement learning (DRL) methods including DQN, PPO, and SAC take 20 to 100 ms and require GPUs for training, though FPGAs can handle inference. It is worth noting that DRL is currently used more for advisory warnings than for primary collision avoidance because formal safety guarantees are not yet available.

Table 1. Summary of collision avoidance algorithms, their computational demands, and latency with different hardware configurations.

Algorithm Type	Example Methods	CPU-Only Latency (Large Models)	Latency with Edge Optimizations (Nano/Small + HW Accel)	Hardware Acceleration Needed?	Notes
Rule-based	Safety distance, TTC threshold	<5 ms *	<5 ms *	No	Suitable for simple warnings
CNN object detection (earlier)	YOLOv5-large, SSD, Faster R-CNN	50–200 ms	N/A (not edge-optimized)	Yes (GPU/FPGA required)	High accuracy, compute-intensive
CNN object detection (edge-optimized)	YOLOv11-nano/- small/-medium, YOLOv26 [21,28]	30–60 ms (CPU)	10–30 ms (edge TPU, Jetson) [23–27,30]	Yes (NPU/FPGA/GPU beneficial)	Best for edge deployment
Path planning	A *, RRT, DWA	10–50 ms **	5–15 ms (GPU-accelerated)	Moderate (GPU/FPGA helpful)	RRT * iteration: 1.2–3.9 ms; total planning: 50–450 ms
Deep reinforcement learning	DQN, PPO, SAC	20–100 ms	10–50 ms (GPU/FPGA inference)	Yes (GPU for training)	Advisory warnings only ***

* Excludes sensor acquisition and communication latency. ** GPU acceleration achieves 17–472× speedup, bringing latency below 10 ms; MEC offloading provides approximately 23.5% improvement. *** DRL is currently used for advisory warnings only, not primary collision avoidance, due to lack of formal safety guarantees.

Edge-optimized CNNs (YOLOv11-nano, YOLOv26) run at 30–60 fps (16–33 ms) on embedded platforms, such as Raspberry Pi 4 + Coral TPU or Jetson Nano [23–27,30], making them highly suitable for sub-50 ms warning systems. These requirements drive the need for low-latency edge accelerators, which we review in Section 5.

3.3. From Algorithm Output to Warning Activation

Understanding how collision avoidance algorithms integrate into the warning pipeline is essential for appreciating why low-latency edge computing matters. Figure 3 illustrates the typical flow [25,33].

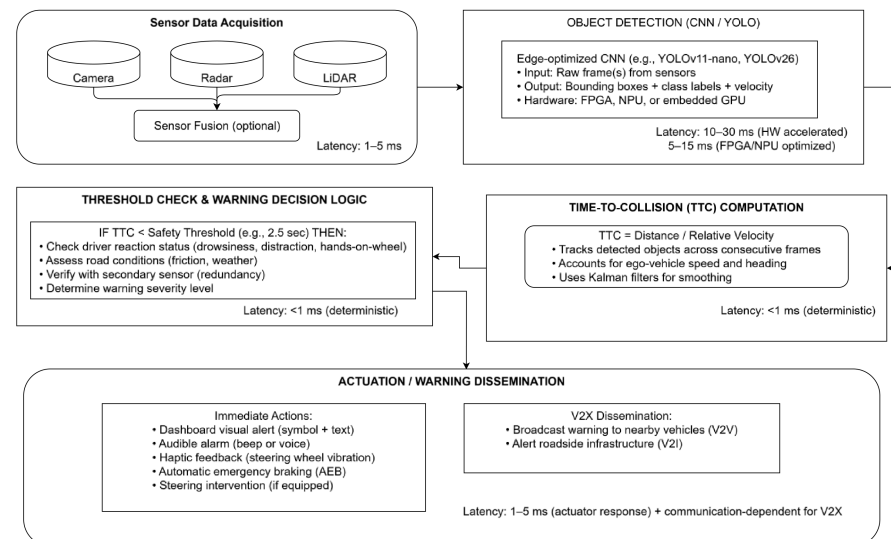


Figure 3. Algorithm-to-warning activation pipeline showing the complete flow from sensor data acquisition to vehicle actuation, with typical latency contributions per stage.

Edge acceleration is most critical in the CNN detection and TTC computation stages [25,33]. By applying hardware acceleration specifically to the CNN detection stage, the overall end-to-end latency can be reduced from 100+ ms (CPU-only) to under 50 ms (edge-accelerated), making sub-10 ms automatic emergency braking (AEB) achievable with FPGA or NPU acceleration.

4. Vehicle-to-Everything Communication and Edge Offloading for Low-Latency Warning Systems

V2X communication enables vehicles to exchange warnings with other vehicles (V2V), infrastructure (V2I), and networks (V2N) [11]. Edge computing can significantly reduce V2X latency by placing processing and caching functions at roadside units or base stations [10,34,35].

4.1. Multi-Access Edge Computing for Vehicle-to-Everything

Machine learning-based blockchain solutions for secure V2X communication have been surveyed, emphasizing the role of multi-access edge computing (MEC) in providing low-latency, robust security. A hierarchical MEC architecture has been proposed where latency-critical warnings are processed at the nearest edge node, achieving sub-20 ms latencies in urban scenarios with sub-10 ms possible under ideal conditions [11].

Mobility prediction is another key enabler. Mobility prediction methodologies for 5G-enabled C-V2X have been reviewed, including edge computing strategies for ultra-reliable low-latency communication (URLLC). Accurate prediction reduces handover delays and ensures continuous warning coverage [10].

Intelligent offloading and driving strategies for delay minimization and collision avoidance in V2X networks [34] address task processing delay and road safety in 6G V2X networks using MEC technology. Similarly, Edge-V [35] proposes a framework combining unlicensed spectrum for low-latency, high-throughput V2X task offloading, reducing latency by up to 65%.

4.2. Task Offloading in Vehicular Edge Computing

Vehicular task offloading has been surveyed and classified based on latency sensitivity. Hard real-time tasks like collision warnings must be processed locally or at the nearest edge node, while delay-tolerant tasks, such as traffic analytics, can be offloaded to the cloud. Hybrid software-defined networking edge architectures can reduce average latency by 35 percent compared to pure V2V offloading [13].

Adaptive communication mechanisms are also crucial. Congestion control and quality of service optimization for vehicular ad hoc networks (VANETs) have been reviewed, highlighting that edge-assisted adaptive beaconing can lower warning dissemination latency by up to 60 percent in dense traffic scenarios [14,36]. CAV driving safety monitoring and warning via V2X-based edge computing systems [36] explores edge computing for monitoring vehicle motion and issuing accident warnings, such as lane departures and collisions. Figure 4 depicts the V2X communication modes with edge offloading points.

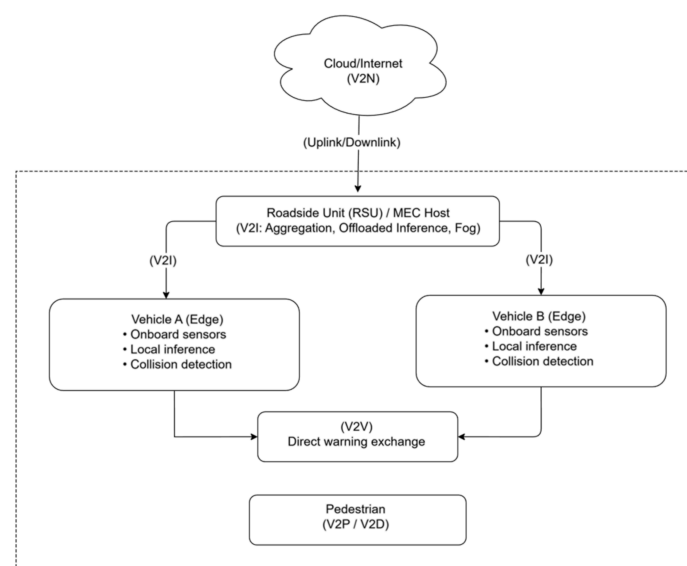


Figure 4. V2X communication modes with edge offloading: V2V, V2I, V2N, and V2P.

The diagram illustrates the four vehicle-to-everything (V2X) communication modes enabled by edge computing architectures. V2V (vehicle-to-vehicle) enables direct low-latency warning exchange between vehicles, allowing immediate sharing of collision risks and hazardous conditions without infrastructure involvement. V2I (vehicle-to-infrastructure) connects vehicles to the roadside unit (RSU), which hosts multi-access edge computing (MEC) resources for local data aggregation, offloaded inference, and fog coordination. V2N (vehicle-to-network) links the RSU to the cloud/Internet, enabling non-real-time analytics, fleet training, and map updates while maintaining the edge for time-critical warnings. V2P (vehicle-to-pedestrian) allows vehicles to communicate with pedestrians carrying mobile devices, providing vulnerable road user warnings and enhancing overall traffic safety. Edge offloading points are strategically placed at the RSU/MEC host to minimize latency by processing safety-critical data locally before cloud transmission.

4.3. Fog Computing Architectures for Vehicular Networks

While edge computing brings processing to the immediate vehicle or RSU, fog computing extends this paradigm by creating a continuum between the edge and the cloud, utilizing intermediate nodes (such as smart traffic lights, gateways, and even mobile vehicles themselves) as fog nodes [37,38]. In dense urban VANETs, pure edge nodes can

become overloaded. Fog nodes offload aggregative and non-critical tasks, reserving edge resources strictly for safety-critical warnings.

A 5G-enabled fog computing model specifically for VANETs has been proposed, demonstrating that a hierarchical fog layer reduces average latency by up to 40% compared to conventional cloud offloading by processing data at the network edge before it reaches the cloud [37]. Similarly, a fuzzy logic-based task scheduling approach within a fog-enabled VANET has been introduced, achieving efficient load balancing while maintaining latencies below 20 ms [38]. Furthermore, the concept of “Mobile Vehicles as Fog Nodes” has been explored, where idle vehicular computational resources are pooled to assist nearby RSUs, dynamically optimizing latency in smart city scenarios [38].

Recent advances in fog-enabled vehicular networks have further demonstrated the benefits of this architectural layer. Optimizing traffic safety message dissemination and resource allocation using adaptive deep reinforcement learning in fog-enabled Internet of vehicle networks [39] exploits DRL (DQN and PPO) for offloading and resource allocation with lower latency. A multi-tier vehicular edge-fog framework for real-time traffic management in smart cities [22] demonstrates 86.4% latency reduction and 53.3% network consumption reduction compared to traditional cloud-based frameworks. Computation offloading and resource allocation in mixed cloud/vehicular-fog computing systems [40] consider vehicle-carried fog nodes (V-FNs) for task offloading. A comprehensive review of task offloading methods in vehicular fog computing [41] provides a comprehensive overview of low-latency offloading strategies. Additionally, multi-layer task offloading schemes in fog computing-based VANETs with optimized completion delay [42] propose efficient multi-layer offloading strategies.

Integrating fog computing into vehicle warning systems provides a critical resilience layer: if the direct edge RSU is congested, the fog node acts as a secondary low-latency processing point, ensuring that warning messages still propagate with minimal delay. This three-tier architecture (vehicle-edge-fog-cloud) is essential for future 6G-enabled ITS.

5. Hardware Accelerators for Edge Artificial Intelligence in Vehicle Warning Systems

The core of low-latency edge processing lies in hardware accelerators that execute inference and signal processing tasks efficiently. This section reviews FPGAs, low-power arithmetic, and approximate computing techniques.

5.1. Field-Programmable Gate Array Accelerators for Real-Time Inference

FPGAs offer a compelling balance of low latency, moderate power consumption, and reconfigurability. The 2025 IEEE Access survey reviews FPGA-based CNN accelerators for edge computing, focusing on autonomous vehicles and traffic monitoring. Key findings were that FPGA accelerators can achieve sub-millisecond inference for small CNNs, like LeNet, with 32×32 inputs, and 5 to 10 ms for medium-sized networks, such as YOLOv3-tiny. Power consumption ranges from 2 to 15 W, significantly lower than GPUs (50–250 W). Design space exploration tools can optimize for latency versus resource utilization [23,31,32].

A direct comparison of GPUs and FPGAs for lane detection shows that an FPGA implementation on Xilinx Zynq achieves 2.1 ms per frame at 3.8 W, while a GPU on NVIDIA Jetson TX2 achieves 1.8 ms at 10 W. The FPGA offers better energy efficiency at 0.55 J/frame compared to 1.8 J/frame [43]. Real-time object detection and classification using YOLO for edge FPGAs [23] further validates this approach, presenting a resource-efficient YOLOv5 system optimized for FPGA deployment.

5.2. Low-Latency Adder and Approximate Computing

At the fundamental circuit level, addition and multiplication dominate digital signal processing and AI inferencing. Configurable low-latency approximate adders can reduce critical path delay by up to 50% with minimal accuracy loss (e.g., <1% relative error) [44]. A 4-bit CMOS full adder designed for low-power IoT and edge applications demonstrated a 35% power reduction compared to conventional designs at the same throughput [45].

Building on this, approximate computing trades minor accuracy for significant gains in latency and energy efficiency. Edge detection using approximate adders (e.g., lower-part OR adder and error-tolerant adder) has been evaluated, finding that for vision-based vehicle detection, a 10% error in edge maps is acceptable, leading to a 40% reduction in latency and 30% power savings [46]. While approximate arithmetic shows promise for pre-filtering or driver-monitoring stages, the necessity of formal error-bounding verification before deployment in safety-critical collision warning loops is emphasized.

Recent research has further explored approximate computing for vehicular applications. AdApTS [47] presents an adaptive approximate computing approach for traffic sign recognition in resource-constrained self-driving systems. Approximate LSTMs for time-constrained inference [48] introduce progressive inference combining model pruning and computation restructuring for approximation under latency budgets. AppSign [49] presents a multi-level approximate computing approach balancing accuracy, computational overhead, and response time for autonomous vehicles. A survey on approximate edge AI for energy-efficient autonomous driving services [50] reviews connected vehicular applications, communications, approximation, and edge AI techniques for energy-efficient autonomous driving. Figure 5 illustrates a typical field-programmable gate array accelerator pipeline for collision detection, from sensor input through convolutional neural network inference to warning decision.

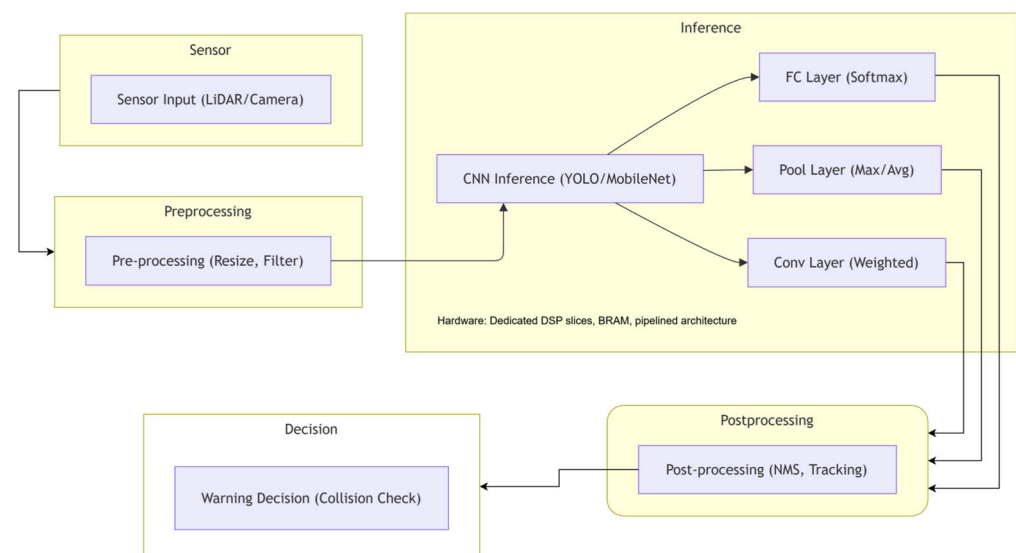


Figure 5. FPGA accelerator pipeline for collision detection.

The diagram illustrates the FPGA accelerator pipeline for collision detection, consisting of five sequential hardware stages: sensor input from LiDAR or camera, pre-processing for resizing and filtering, CNN inference running models, such as YOLO or MobileNet, post-processing for non-maximum suppression and tracking, and warning decision for collision checking. Each stage operates as a dedicated hardware module with no CPU involvement, enabling fully deterministic execution. All stages run in parallel simultaneously through deep pipelining, achieving maximum throughput without software-induced jitter. On-chip

memory (weight and feature buffers) supports the pipeline by minimizing off-chip data movement, which reduces latency and power consumption. This architecture achieves a deterministic total latency of 1 to 10 milliseconds, making it suitable for sub-10 ms automatic emergency braking (AEB) and real-time collision warning systems. The deterministic timing is a critical advantage over GPU-based solutions, which suffer from scheduler overhead and memory contention.

6. Comparative Analysis of Edge Architectures for Vehicle Warnings

To help researchers and engineers select appropriate architectures, a comparative analysis of key metrics including latency, power, area, and real-time suitability is presented.

6.1. Latency Comparison

Table 2 summarizes average processing latencies for different architecture approaches.

Table 2. Latency comparison across architectures.

Architecture	Processing Latency (per Frame/Inference)	Suitability for <10 ms Warning
Cloud-only (4G/5G)	50–200 ms	No
Fog (RSU/Gateway)	10–30 ms	Suitable (with 5G/ideal conditions)
Edge (CPU, software)	20–50 ms	No (borderline)
Edge (GPU, e.g., Jetson)	5–15 ms	Marginal (depends on load)
Edge (FPGA)	1–10 ms	Yes (deterministic)
NPU/ASIC	1–5 ms	Yes (ultra-low-power, fixed-function)
Hybrid (edge + cloud)	5–30 ms	Depends on offloading strategy

Cloud-only systems using 4G or 5G networks typically take 50 to 200 ms, which exceeds the safety threshold for collision warnings. Fog computing, which leverages intermediate roadside gateways and mobile vehicular nodes, reduces this to 10–30 ms by offloading aggregative and non-critical tasks before they reach the cloud [22,37–42,51]. Edge processing on CPUs running software implementations takes 20 to 50 ms, which remains borderline for sub-10 ms requirements. Edge GPUs such as the NVIDIA Jetson series achieve 5 to 15 ms, which is marginal and depends on system load. Edge FPGAs achieve 1 to 10 ms with deterministic timing, making them suitable for sub-10 ms warnings. Finally, dedicated NPUs/ASICs deliver the lowest latency at 1 to 5 ms, but their fixed-function nature limits reconfigurability for evolving algorithms [30,52–54]. Hybrid edge-cloud approaches fall in the 5–30 ms range, with performance strongly dependent on offloading decisions. Figure 6 shows a bar chart comparing these average processing latencies.

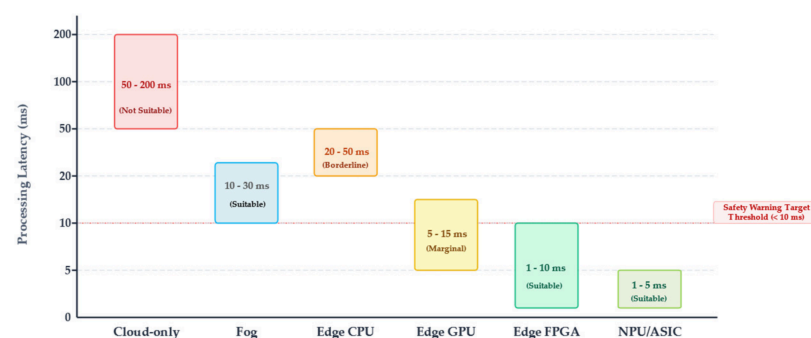


Figure 6. Latency comparison of edge, fog, and cloud architectures.

The bar chart compares average processing latency across six computation targets: cloud-only, fog nodes, edge CPU, edge GPU, edge FPGA, and NPU/ASIC. Cloud-only

solutions range from 50 to 200 milliseconds and are labeled “Not Suitable” for sub-10 ms vehicle warning requirements due to network transmission and backhaul delays. Fog nodes (RSUs/gateways) achieve 10 to 30 milliseconds, making them suitable under ideal 5G conditions by offloading aggregative and non-critical tasks before cloud transmission. Edge CPU implementations range from 20 to 50 milliseconds and are labeled “Borderline,” as they exceed the sub-10 ms threshold for collision warnings. Edge GPU solutions range from 5 to 15 milliseconds and are labeled “Marginal,” as performance depends on system load and scheduler overhead, introducing timing variability. Edge FPGA accelerators achieve 1 to 10 milliseconds with deterministic timing, making them fully “Suitable” for sub-10 ms warnings. NPUs/ASICs deliver the lowest latency at 1 to 5 milliseconds, also “Suitable,” but with the trade-off of fixed-function non-reconfigurability. The chart concludes that sub-10 millisecond deterministic processing achievable with FPGAs and NPUs is safe and compliant for autonomous warnings, while cloud-only and CPU-based solutions are inadequate for safety-critical applications.

6.2. Power and Energy Efficiency

Table 3 compares power consumption and energy efficiency across accelerator types. High-end graphics processing units consume 150 to 250 watts and require 15 to 30 joules per inference, making them suitable for training and non-real-time applications but impractical for vehicle edge deployment.

Table 3. Power and energy efficiency comparison across accelerator types.

Accelerator Type	Power (W)	Energy per Inference (J)	Best Use Case
High-end GPU (e.g., NVIDIA A100)	150–250	15–30	Training, non-real-time batch processing
Fog node (RSU/edge server) [22,37,38,40,51]	80–150	3–10	Intermediate aggregation, overload relief
Embedded GPU (e.g., Jetson Orin)	10–15	0.5–2	Moderate real-time, flexible software stack
Edge NPU/ASIC (e.g., Google Edge TPU, Huawei Ascend, Qualcomm Hexagon) [30,52–54]	1–5	0.05–0.5	Fixed-function, ultra-low-power edge inference
FPGA (mid-range, e.g., Xilinx Zynq)	5–15 *	0.2–1	Low-latency, energy-sensitive, reconfigurable
Custom ASIC (mass production) [55]	1–5	0.05–0.2	High-volume, fixed-function production

* FPGA power ranges from 5 to 15 watts in typical implementations; some low-power FPGAs, such as those from Lattice Semiconductor, achieve under 5 watts.

Embedded graphics processing units, such as the NVIDIA Jetson series, consume 10 to 15 watts and 0.5 to 2 joules per inference, offering moderate real-time capability with a flexible software stack [43]. Mid-range field-programmable gate arrays consume 5 to 15 watts and 0.2 to 1 joule per inference, making them well suited for low-latency, energy-sensitive applications with deterministic timing and reconfigurability [23,31,32]. Custom application-specific integrated circuits consume 1 to 5 watts and 0.05 to 0.2 joules per inference, but they lack reconfigurability and require high non-recurring engineering costs, making them practical only for mass production [55].

Fog nodes (roadside units or edge servers equipped with x86 CPUs or GPUs) occupy a distinct middle ground, consuming 80 to 150 watts of power and delivering 3 to 10 joules

per inference [22,37,38,40,51]. While they are not as energy efficient as dedicated edge accelerators, they serve an essential architectural role by aggregating data from multiple vehicles, handling non-critical analytics, and acting as a resilient backup when the immediate edge RSU is congested.

Emerging NPUs and custom ASICs (e.g., Google Edge TPU, Huawei Ascend, Qualcomm Hexagon) achieve the lowest energy consumption at 1 to 5 watts and 0.05 to 0.5 joules per inference, making them the most energy-efficient option for fixed-function inference tasks [30,52–54]. Video-based traffic light recognition by Rockchip RV1126 [54] demonstrates real-time traffic light recognition (>25 FPS) on an NPU, validating model practicality for edge deployment. However, NPUs lack post-fabrication reconfigurability, limiting adaptability to new algorithms or safety standards compared to FPGAs.

Figure 7 presents a scatter plot that visually contrasts the power-latency trade-offs across all surveyed processing tiers. Edge CPUs consume the least power (1–5 W) but suffer from the highest latency (20–50 ms), rendering them borderline for sub-10 ms warning generation. Edge FPGAs offer an excellent compromise at 5–15 W and 1–10 ms, providing deterministic, reconfigurable acceleration ideal for evolving safety algorithms. Edge GPUs consume moderately higher power (10–15 W) for similar latency ranges (5–15 ms) but lack deterministic timing due to scheduler overhead. High-end GPUs dominate power consumption (150–250 W) while offering only moderate latency (5–15 ms), making them suitable for cloud training but impractical for in-vehicle deployment.

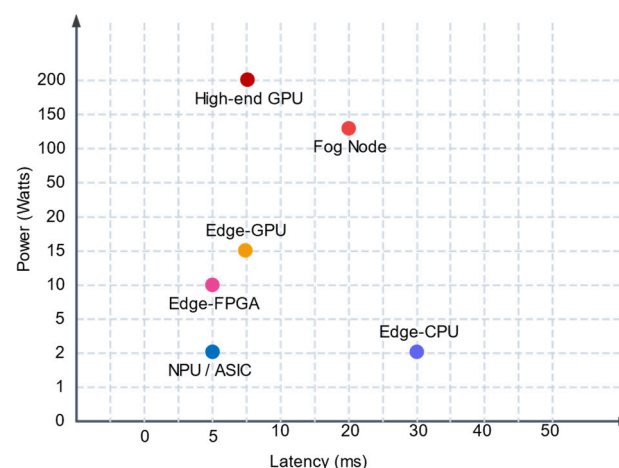


Figure 7. Power-latency trade-off scatter plot.

Emerging NPUs and custom ASICs occupy the ideal bottom-left corner, achieving the lowest latency (1–5 ms) at the lowest power (1–5 W) [30,52–54], making them the most energy-efficient for fixed-function inference. Video-based traffic light recognition by Rockchip RV1126 [54] demonstrates real-time traffic light recognition (>25 FPS) on an NPU, validating model practicality for edge deployment. However, NPUs lack post-fabrication reconfigurability, limiting adaptability to new algorithms or safety standards.

Critically, fog nodes (roadside units or edge servers equipped with x86 CPUs or GPUs) occupy a distinct middle ground. As surveyed in Section 4.3, fog nodes consume 80–150 W of power and deliver 10–30 ms latency [22,37,38,40,51]. While they are not as fast or energy efficient as dedicated edge accelerators, they serve an essential architectural role: they aggregate data from multiple vehicles, handle non-critical analytics, and act as a resilient backup when the immediate edge RSU is congested. In the power-latency plot, fog nodes sit above the edge devices but far below the cloud in latency, visually representing their role as the bridge between the edge and the cloud.

The scatter plot illustrates the power-latency trade-off across six hardware and architectural categories. High-end GPUs consume the most power at 150 to 250 watts with moderate latency of 5 to 15 milliseconds, making them suitable for training and non-real-time applications but impractical for vehicle edge deployment. Fog nodes (roadside units/edge servers) consume 80 to 150 watts with 10 to 30 milliseconds latency, serving as intermediate aggregation points that bridge the gap between edge and cloud [22,37,38,40,51]. Edge GPUs (e.g., NVIDIA Jetson series) consume 10 to 15 watts with 5 to 15 milliseconds of latency, offering moderate real-time capability but lacking deterministic timing due to scheduler overhead [43]. Edge FPGAs consume 5 to 15 watts with 1 to 10 milliseconds of latency, balancing efficiency and speed with deterministic, reconfigurable acceleration ideal for evolving safety algorithms [23,31,32]. Edge CPUs consume only 1 to 5 watts but exhibit the highest latency at 20 to 50 milliseconds, rendering them borderline for sub-10 ms warning generation. NPUs/ASICs (e.g., Google Edge TPU, Huawei Ascend) occupy the ideal bottom-left corner, consuming 1 to 5 watts with 1 to 5 milliseconds of latency, making them the most energy-efficient option for fixed-function inference tasks, albeit with limited reconfigurability compared to FPGAs [30,52–54]. The ideal region is the bottom-left (low power, low latency); NPU/ASIC achieves the closest proximity for fixed-function workloads, while FPGA offers the best balance for reconfigurable safety-critical systems, and fog nodes fill the gap between edge and cloud for aggregative and non-critical tasks.

6.3. Trade-Offs Summary

Table 4 summarizes the trade-offs among the main edge architectures across several criteria.

Table 4. Trade-offs and suitability for vehicle warning systems.

Criterion	Cloud	Fog	Edge (CPU)	Edge (GPU)	Edge (FPGA)	NPU/ASIC
Low latency (<10 ms)	No	Suitable (10–30 ms)	No	Yes	Yes	Yes (1–5 ms)
Low power (<5 W)	N/A	No (80–150 W)	Yes	No	Moderate ¹	Yes (1–5 W)
Reconfigurability	N/A	Moderate	Yes	Moderate ²	Yes	No (Fixed-function)
Development ease	Yes	Moderate	Yes	Moderate ³	No	Moderate ⁴
Deterministic timing	No	Moderate	No	Moderate ⁵	Yes	Yes

¹ FPGA power ranges from 5 to 15 watts; some low-power FPGAs achieve under 5 watts. ² GPU reconfigurability is limited to software changes; the hardware architecture remains fixed. ³ GPU development is easier than FPGA development using CUDA or TensorRT compared to HDL, but it still requires real-time optimization expertise. ⁴ NPU/ASIC development uses vendor-specific SDKs (e.g., edge TPU compiler, Huawei CANN) with moderate learning curves. ⁵ GPU timing is non-deterministic due to scheduling overhead and memory contention; fog timing depends on network conditions and load.

For low latency under 10 ms, cloud-only solutions are unsuitable, while fog nodes are suitable under ideal 5G conditions (10–30 ms), and both GPUs and FPGAs are suitable. NPUs/ASICs are also suitable (1–5 ms) but with fixed-function limitations. For low power under 5 W, cloud is not applicable, edge CPUs are good, edge GPUs are poor, and FPGAs are moderate (with some low-power variants achieving <5 W); NPUs/ASICs also achieve <5 W for fixed-function inference. For reconfigurability, edge CPUs and FPGAs offer significant flexibility, whereas GPUs are moderately reconfigurable (limited to software adjustments), and ASICs/NPUs are fixed-function. Fog nodes offer moderate reconfigurability through software-defined gateways. For development ease, cloud, edge CPU, and edge GPU are good, while FPGAs are more challenging; fog development is moderate due to heterogeneous hardware integration. Regarding deterministic timing, FPGAs and

ASICs excel, GPUs suffer from scheduler and memory contention overhead, fog nodes are moderately deterministic depending on network conditions, and CPUs/cloud exhibit the highest variability.

6.4. Critical Synthesis of the Surveyed Literature

To provide a structured, at-a-glance comparison of the core studies surveyed, Table 5 critically synthesizes the methodologies, contributions, and limitations of representative works. This synthesis highlights the fragmentation in evaluation metrics (e.g., some use end-to-end latency, others only processing latency) and the lack of standardized testing environments, which directly underpins the research gaps discussed in Section 7.

Table 5. Critical synthesis of representative edge computing studies for vehicle warnings.

Focus Area	Methodology/Algorithm	Strength	Weakness/Gap	Applicability to Vehicle Warnings
Edge Intelligence Frameworks [1]	Hierarchical edge-cloud collaboration survey	Comprehensive taxonomy	Lacks quantitative hardware comparison	High (architectural guidance)
Collision Avoidance Algorithms [5]	Systematic review of DRL and CNN methods	Covers algorithmic diversity	Does not address hardware mapping	Medium (algorithm selection)
FPGA Implementation [8]	Sobel + Manhattan distance on Cyclone IV	Achieves 5.2 ms latency	Tested only in lab, not real roads	High (proof of HW viability)
V2X Dissemination [4]	Edge relay broadcast storm reduction	40% latency reduction via simulation	Simulation-only; no HW prototype	Medium (communication strategy)
GPU vs. FPGA [43]	Lane detection on Zynq vs. Jetson TX2	Direct quantitative HW comparison	Limited to lane detection only	High (energy efficiency benchmark)
Fog Computing [37]	5G fog model for VANET	40% latency reduction vs. cloud	Assumes ideal 5G coverage	High (latency optimization)
Fog Task Scheduling [38]	Fuzzy logic scheduling in fog-VANET	Balances load, keeps <20 ms latency	Complexity of fuzzy rules in dynamic V2X	High (resource management)
Mobile Fog Nodes [51]	Vehicles-as-fog for latency optimization	Dynamic resource pooling	Security/privacy of shared vehicle resources	High (novel architectural insight)
ASIC/NPU Performance [30]	Benchmarking Google Coral vs. GPU	1–5 W power, fast ML inference	Fixed models; lacks reconfigurability	Medium (low-power inference)
Edge-Optimized CNNs [23–27]	YOLOv5/v8/v11 FPGA/NPU implementations	Real-time detection on edge	Limited to specific model variants	High (practical deployment)

7. Research Gaps and Future Directions

Based on the critical synthesis in Section 6.4, five principal research gaps are identified that collectively hinder the widespread deployment of low-latency edge architectures for vehicle warning systems. These gaps span benchmarking standardization, hardware reliability, algorithmic safety, resource orchestration, and sustainable deployment.

- Gap 1: Lack of Standardized Benchmarks for Edge-Based Warning Systems

Current research uses highly diverse datasets (KITTI, BDD100K, custom collections) and inconsistent latency metrics; some report end-to-end latency including communication, whereas others report pure processing time. This fragmentation makes fair cross-study comparisons virtually impossible. A standardized benchmark suite tailored specifically to edge-based vehicular warning systems is urgently needed. Proposed direction: Develop an open benchmark suite that includes representative driving scenarios, unified latency definitions, and safety-critical performance constraints. Estimated impact: Enables objective, reproducible comparisons across hardware platforms and algorithms.

- Gap 2: Limited Integration of Approximate Computing in Safety-Critical Loops

Approximate computing offers dramatic latency and power reductions of up to 40% and 30%, respectively, but its application to functions like collision detection remains underexplored due to legitimate reliability concerns. Without formal error-bounding guarantees, approximate circuits cannot be trusted in primary warning paths [47–50]. Proposed direction: Design error-resilient approximate accelerators with mathematically guaranteed worst-case error bounds (e.g., $\pm 5\%$ in TTC or distance estimation). Estimated impact: 30–50% reduction in both latency and power consumption for pre-processing and feature-extraction stages.

- Gap 3: Field-Tested FPGA Prototypes Under Real Driving Conditions

The vast majority of FPGA-based warning systems are validated in simulation or controlled testbeds. Real-world driving exposes hardware to vibration, temperature extremes, humidity, and electromagnetic interference factors that can severely impact timing closure, thermal throttling, and bit error rates [8,23]. Proposed direction: Conduct 6–12 month field trials using vehicle-mounted FPGA prototypes across diverse geographic and climatic conditions. Estimated impact: Provides the reliability data necessary for automotive certification (e.g., ISO 26262:2018 [56]) and production deployment.

- Gap 4: Integration of V2X and Edge AI in Heterogeneous Networks

While V2X edge offloading and AI inference have been studied separately, their joint optimization remains an open problem. Dynamic channel conditions, vehicle density, and handover events create highly variable communication latencies that can dominate the end-to-end warning delay. Static offloading policies fail under real traffic dynamics [34,35]. Proposed direction: Design machine learning-based online schedulers that co-optimize communication resource allocation (spectrum, power) and computation distribution (edge vs. fog vs. vehicle) in real time. Estimated impact: 20–40% reduction in end-to-end latency under high-mobility urban scenarios.

- Gap 5: Energy-Harvesting Edge Nodes for Sustainable Warning Systems

Roadside edge nodes and RSUs currently rely on grid power or batteries, limiting deployment in remote or economically constrained areas. Future nodes could harvest energy from road vibration (piezoelectric), solar radiation, or even thermal gradients. Low-power arithmetic and approximate computing are key enablers, but full system-level integration from harvester to processor to communication module has not been demonstrated in a vehicular safety context. Proposed direction: Integrate ultra-low-power approximate adders and sleep-wake scheduling with piezoelectric or solar harvesting modules into a prototype RSU. Estimated impact: Enables truly maintenance-free, off-grid deployment of roadside warning infrastructure.

Table 6 summarizes these five gaps alongside their proposed directions.

Table 6. Summary of key research gaps and proposed future directions.

Gap	Proposed Future Direction
No standardized benchmarks	Develop a public benchmark suite with unified latency metrics and representative driving scenarios
Approximate computing in safety loops	Design error-resilient accelerators with mathematically guaranteed bounded error ($\pm 5\%$ in TTC) [42,47–49]
Lack of field-tested FPGA prototypes	Conduct long-term field trials under real driving conditions [8,23]
Joint communication-computation optimization	Design ML-based online schedulers for V2X edge offloading under dynamic channel conditions [34,35]
Energy-harvesting edge nodes	Integrate low-power arithmetic and sleep-wake scheduling with piezoelectric/solar harvesting

Real-Time Edge-Enhanced Vehicular Decision Making

In addition to the five infrastructure-level gaps above, a critical and distinct under-explored area is how edge inference directly enhances end-decision making in vehicles, specifically the integration of edge outputs into real-time trajectory correction and predictive actuation. While this review covers hardware latency, the control loop integration remains fragmented across the literature.

Edge computing can fuse multi-modal sensor data (LiDAR, Radar, Camera) at sub-10 ms intervals to feed into Model Predictive Control (MPC) algorithms for immediate braking or steering interventions [33]. Furthermore, edge-based shared situational awareness where one vehicle's detected hazard is broadcast to nearby vehicles via V2X within the same edge domain can preemptively trigger warnings in trailing vehicles [25,36]. Recent studies [57,58] demonstrate that edge-enabled predictive analytics can reduce the severity of rear-end collisions by up to 35%.

However, the tight coupling of edge inference outputs with real-time vehicular actuators (drive-by-wire systems) introduces new safety certification challenges that are seldom addressed in the current edge literature. Unlike infrastructure gaps (Gaps 1–5), this gap directly involves functional safety standards (ISO 26262:2018 [56] ASIL-D), real-time operating system integration, and fail-operational requirements. Therefore, this is identified as a separate, high-priority future direction: the development of safety-certified edge inference kernels that guarantee not only low latency but also deterministic execution under worst-case scheduling conditions, with built-in redundancy and error detection for actuator commands.

Proposed direction: Design and formally verify a reference edge inference pipeline (from sensor fusion to actuation command) that meets ASIL-D requirements, including timing isolation, memory protection, and failover mechanisms. Estimated impact: Bridges the gap between academic edge AI prototypes and production-grade vehicular safety systems.

8. Limitations of This Review

This review focuses on the peer-reviewed literature from 2023 to 2025 mostly and may omit relevant industrial white papers or pre-print server content. The comparative analysis synthesizes results from different testbeds and workloads, so readers should exercise caution when directly comparing absolute latency numbers across studies. Systematic meta-analysis or independent replication of the reported results was not conducted. Additionally, the rapid evolution of edge computing hardware means that some performance figures, particularly for commercial GPUs and FPGAs, may become outdated quickly.

9. Conclusions

This review synthesized the state of the art in low-latency edge computing architectures for real-time vehicle warning systems. Edge intelligence frameworks for ITS, collision avoidance algorithms, V2X communication and offloading strategies, and hardware accelerators, including FPGAs, low-power adders, NPU/ASICs, and approximate computing [47–50], were surveyed. A comparative analysis showed that FPGA-based accelerators offer the best trade-off for deterministic sub-10 ms latency with moderate power consumption (5–15 W), while emerging NPUs provide even lower power (1–5 W) for fixed-function ML inference [30,52–54]. Fog computing was also highlighted as a critical intermediary layer to bridge edge-cloud gaps [22,37–42,51]. Key research gaps include the lack of standardized benchmarks, limited field validation, untapped potential of approximate computing for safety-critical functions, and the urgent need for safety-certified edge inference for direct vehicular actuation [33,57,58].

Future work should focus on developing benchmark suites, field-testing FPGA prototypes, designing energy-harvesting edge nodes, and creating ISO 26262:2018 [56]-compliant edge inference kernels. This review provides a foundation for researchers and engineers aiming to build next-generation, low-latency vehicle warning systems that save lives on the road.

Author Contributions: Conceptualization, R.K.A. and D.Z.; methodology, R.K.A.; formal analysis, R.K.A.; investigation, R.K.A.; writing—original draft preparation, R.K.A.; writing—review and editing, R.K.A. and D.Z.; visualization, R.K.A.; supervision, D.Z.; project administration, D.Z.; funding acquisition, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key Research and Development Program of China under grant number 2024YFC3017104, “Research on Real-time Automatic Release of Warning Information and Vehicle Warning Interception Technology and Devices”. The APC was funded by the same grant.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors thank the China Academy of Safety Science and Technology for providing the research context and application requirements that motivated this review. The authors also thank the School of Control Science and Engineering at China University of Mining and Technology for providing research facilities and computational resources. During the preparation of this manuscript, the authors used grammar and spell-checking tools for language refinement only. The authors have reviewed and edited all output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

ADAS	Advanced Driver-Assistance System
AEB	Automatic Emergency Braking
AI	Artificial Intelligence
ASIC	Application-Specific Integrated Circuit
C-V2X	Cellular Vehicle-to-Everything
CNN	Convolutional Neural Network
DRL	Deep Reinforcement Learning
DSP	Digital Signal Processing
FPGA	Field-Programmable Gate Array
ITS	Intelligent Transportation System
MEC	Multi-Access Edge Computing
MPC	Model Predictive Control
NPU	Neural Processing Unit
RSU	Roadside Unit
TTC	Time-to-Collision
URLLC	Ultra-Reliable Low-Latency Communication
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VANET	Vehicular Ad Hoc Network

References

1. Ghasemi, A.; Keshavarzi, A.; Abdelmoniem, A.M.; Nejati, O.R.; Derikvand, T. Edge intelligence for intelligent transport systems: Approaches, challenges, and future directions. *Expert Syst. Appl.* **2025**, *280*, 127273. [\[CrossRef\]](#)
2. Khang, A. *Driving Green Transportation System Through Artificial Intelligence and Automation: Approaches, Technologies and Applications*; Springer Nature: London, UK, 2025.
3. Ryu, D.; Kang, Y.; Jeong, M.; Batzorig, M.; Yim, K. Edge AI in vehicle modules using heterogeneous networks: Research trends and future directions. In *Innovative Mobile and Internet Services in Ubiquitous Computing: Proceedings of the 19th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2025)*; Springer Nature: London, UK, 2025; pp. 314–324.
4. Tomar, R.; Sastry, H.G.; Prateek, M. A novel framework for efficient information dissemination for V2X. *Int. J. Veh. Inf. Commun. Syst.* **2025**, *10*, 227–242. [\[CrossRef\]](#)
5. Hamidaoui, M.; Talhaoui, M.Z.; Li, M.; Midoun, M.A.; Haouassi, S.; Mekkaoui, D.E.; Smaili, A.; Cherraf, A.; Benyoub, F.Z. Survey of autonomous vehicles' collision avoidance algorithms. *Sensors* **2025**, *25*, 395. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Zhu, Y. EdgeDriver: Optimising autonomous driving assistance with multi-LLM framework in cloud-edge computing environments. *Int. J. Veh. Inf. Commun. Syst.* **2025**, *10*, 285–298.
7. Karthick, G.; Whig, P. Enhancing road safety a review of deep learning techniques for accident avoidance. *SGS-Eng. Sci.* **2025**, *1*. Available online: <https://spast.org/techrep/article/view/5271/528> (accessed on 16 July 2026).
8. Kumar, V.P.A.; Bhattacharjee, S.; Kumar, H.; Mal, R.; Ravichandran, V.; Sivasankaran, K. FPGA based Vehicle Collision Avoidance and Accident Warning using Sobel Operation and Manhattan Distance Metrics. In *Proceedings of the 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, Bengaluru, India, 5–7 February 2025; IEEE: New York, NY, USA, 2025; pp. 837–842.
9. Saloni; Dutta, U.; Pahal, S. A review of different strategies for vehicle collision avoidance. *Int. J. Veh. Inf. Commun. Syst.* **2025**, *10*, 299–324. [\[CrossRef\]](#)
10. Abboud, M.B.; Drissi, M.; Baala, O.; Allio, S. Optimizing mobility prediction in 5G for enhanced C-V2X applications: A multidisciplinary research survey. *Comput. Commun.* **2025**, *242*, 108254. [\[CrossRef\]](#)
11. Gebrezgiher, Y.T.; Jeremiah, S.R.; Deng, X.; Park, J.H. Machine learning-based blockchain technology for secure V2X communication: Open challenges and solutions. *Sensors* **2025**, *25*, 4793. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Wang, T. Orchestrating Trajectory, Active Jamming, and Antenna Selection for Energy-Efficient Secure Aerial IRS Communications. *Veh. Commun.* **2025**, *57*, 100987. [\[CrossRef\]](#)
13. Ahmed, M.; Raza, S.; Mirza, M.A.; Aziz, A.; Khan, M.A.; Khan, W.U.; Li, J.; Han, Z. A survey on vehicular task offloading: Classification, issues, and challenges. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 4135–4162. [\[CrossRef\]](#)
14. al-Qumati, I.a.-D.A.M.; Saada, O.A.-M.A. Adaptive Communication Mechanisms in VANETs: A Survey on Congestion Control, QoS Optimization, and Reliable Data Dissemination: Adaptive Communication Mechanisms in VANETs. *J. Humanit. Appl. Sci.* **2025**, *9*, 27–35.
15. Markets, R.A. *Automotive Edge AI Market—Strategic Insights and Forecasts (2026–2031)*; 6232071; Research and Markets: Dublin, Ireland, 2026.
16. Katariya, V.; Pazho, A.D.; Noghre, G.A.; Tabkhi, H. VegaEdge: Edge AI confluence for real-time IoT-applications in highway safety. *Internet Things* **2024**, *27*, 101268. [\[CrossRef\]](#)
17. Yan, H.; Gu, Y.; He, H.; Ning, X.; Wang, Q.; Cheng, L. DNN-based task partitioning and offloading in edge-cloud collaboration within electric vehicles. *IEEE Trans. Consum. Electron.* **2024**, *71*, 4100–4109.
18. Ali, E.M.; Abawajy, J.; Lemma, F.; Baho, S.A. Analysis of deep reinforcement learning algorithms for task offloading and resource allocation in fog computing environments. *Sensors* **2025**, *25*, 5286. [\[CrossRef\]](#) [\[PubMed\]](#)
19. Banerjee, S.; Mondal, M.K.; Roy, M.; Alnumay, W.S.; Biswas, U. A deep learning-based car accident detection framework using edge and cloud computing. *IEEE Access* **2024**, *12*, 130107–130115. [\[CrossRef\]](#)
20. Zhang, J.; Xu, Q.; Li, Z.; Xu, C.; Li, K. Cooperative Safety Intelligence in V2X-Enabled Transportation: A Survey. *arXiv* **2025**, arXiv:2512.00490.
21. Sameer, M.M. A Context-Aware Hybrid Trust Management Model for the Internet of Vehicles. Master's Thesis, Macquarie University, Sydney, NSW, Australia, 2025.
22. Hassan, S.R.; Mehmood, A. A Multi-Tier Vehicular Edge-Fog Framework for Real-Time Traffic Management in Smart Cities. *Mathematics* **2025**, *13*, 3947. [\[CrossRef\]](#)
23. Al Amin, R.; Obermaisser, R. Real-Time Object Detection and Classification using YOLO for Edge FPGAs. In *Proceedings of the 2025 International Symposium ELMAR, Zadar, Croatia, 15–17 September 2025*; IEEE: New York, NY, USA, 2025; pp. 291–295.
24. Adam, M.A.; Tapamo, J.R. Enhancing YOLOv5 for Autonomous Driving: Efficient Attention-Based Object Detection on Edge Devices. *J. Imaging* **2025**, *11*, 263. [\[CrossRef\]](#) [\[PubMed\]](#)

25. Punia, D.; Kumar, R. A YOLOv3-powered edge computing technique for real-time rear-end collision prediction in autonomous vehicles. *Appl. Soft Comput.* **2025**, *185*, 113981. [\[CrossRef\]](#)
26. Lamb, L.; Mohammadi, M.; Zand, R. Multi-Modal Vision at the Edge: Toward Low-Latency Perception for Autonomous Systems. In *Proceedings of the 2025 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), Madison, WI, USA, 4–6 August 2025*; IEEE: New York, NY, USA, 2025; pp. 1–7.
27. Xing, C.; Sun, H.; Yang, J. A Lightweight Traffic Sign Detection Model Based on Improved YOLOv8s for Edge Deployment in Autonomous Driving Systems Under Complex Environments. *World Electr. Veh. J.* **2025**, *16*, 478. [\[CrossRef\]](#)
28. Sapkota, R.; Karkee, M. Ultralytics YOLO evolution: An overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 object detectors for computer vision and pattern recognition. *arXiv* **2025**, arXiv:2510.09653.
29. Chakrabarty, S. Yolo26: An analysis of NMS-free end to end framework for real-time object detection. *arXiv* **2026**, arXiv:2601.12882.
30. Minott, D.; Siddiqui, S.; Haddad, R.J. Benchmarking edge AI platforms: Performance analysis of Nvidia Jetson and raspberry pi 5 with coral TPU. In *Proceedings of the SoutheastCon 2025, Concord, NC, USA, 22–30 March 2025*; IEEE: New York, NY, USA, 2025; pp. 1384–1389.
31. Qiu, J.; Wang, J.; Yao, S.; Guo, K.; Li, B.; Zhou, E.; Yu, J.; Tang, T.; Xu, N.; Song, S. Going deeper with embedded FPGA platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium On Field-Programmable Gate Arrays, Monterey, CA, USA, 21–23 February 2016*; ACM: New York, NY, USA, 2016; pp. 26–35.
32. Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2015*; ACM: New York, NY, USA, 2015; pp. 161–170.
33. Wang, X.; Yang, Y.; Shangguan, Y.; Yan, W.; An, Z.; Bunting, M.; Nice, M.; Beckers, T.; Ma, M.; Work, D. A Safety-Driven Interpretable Model for Vehicle Control With Impact on Traffic. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 22151–22160. [\[CrossRef\]](#)
34. Li, X.; Yan, C.; Duan, W.; Wang, X. Intelligent Offloading and Driving Strategy for Delay Minimization and Collision Avoidance in V2X Network. In *Proceedings of the 2025 IEEE/CIC International Conference on Communications in China (ICCC Workshops), Shanghai, China, 10–13 August 2025*; IEEE: New York, NY, USA, 2025; pp. 1–6.
35. Raviglione, F.; Casetti, C.; Restuccia, F. Edge-V: Vehicular Edge Intelligence Through Multi-Band Unlicensed Spectrum Access. *IEEE Trans. Veh. Technol.* **2025**, *75*, 5030–5045.
36. Chang, C.; Zhang, J.; Zhang, K.; Zheng, Y.; Shi, M.; Hu, J.; Li, S.; Li, L. CAV driving safety monitoring and warning via V2X-based edge computing system. *Front. Eng. Manag.* **2024**, *11*, 107–127. [\[CrossRef\]](#)
37. Farooqi, A.M.; Alam, M.A.; Hassan, S.I.; Idrees, S.M. A fog computing model for VANET to reduce latency and delay using 5G network in smart city transportation. *Appl. Sci.* **2022**, *12*, 2083. [\[CrossRef\]](#)
38. Ehtisham, M.; Hassan, M.u.; Al-Awady, A.A.; Ali, A.; Junaid, M.; Khan, J.; Abdelrahman Ali, Y.A.; Akram, M. Internet of vehicles (IoV)-based task scheduling approach using fuzzy logic technique in fog computing enables vehicular ad hoc network (VANET). *Sensors* **2024**, *24*, 874. [\[CrossRef\]](#) [\[PubMed\]](#)
39. Tripura, S.; Lu, Q.-C.; Hussain, A.; Mahmud, T. Optimizing traffic safety message dissemination and resource allocation using adaptive deep reinforcement learning in fog-enabled internet of vehicles network. *Egypt. Inform. J.* **2025**, *32*, 100804. [\[CrossRef\]](#)
40. Hu, B.; Du, J.; Zhang, J.; Chu, X. Computation offloading and resource allocation in mixed cloud/vehicular-fog computing systems. *IEEE Trans. Mob. Comput.* **2025**, *24*, 8612–8624. [\[CrossRef\]](#)
41. Sneha, C.; Chakravarthy, A.S.; Veni, T. A comprehensive review of task offloading methods in Vehicular Fog Computing. *Comput. Electr. Eng.* **2026**, *129*, 110847. [\[CrossRef\]](#)
42. Gu, K.; Chen, Q.; Tan, J.; Liang, R.; Cai, L. Multi-Layer Task Offloading Scheme in Fog Computing-Based VANETs With Optimized Completion Delay. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 11574–11591. [\[CrossRef\]](#)
43. Alshemi, M.; Saif, S.; Taher, M. Hardware acceleration of lane detection algorithm: A GPU versus FPGA comparison. *arXiv* **2022**, arXiv:2212.09460.
44. Hanif, M.A.; Hafiz, R.; Shafique, M. Configurable models and design space exploration for low-latency approximate adders. In *Approximate Circuits: Methodologies and CAD*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 3–23.
45. Vishwakarma, V.; Mittal, A.; Gupta, B.B.; Chui, K.T.; Vishwakarma, S.K. A Novel 4-Bit CMOS Based Full Adder for Low-Power IoT and Edge Computing Applications. In *Proceedings of the 2025 22nd International SoC Design Conference (ISOCC), Busan, Republic of Korea, 15–18 October 2025*; IEEE: New York, NY, USA, 2025; pp. 1–2.
46. Sayadi, L.; Moaiyeri, M.H.; Timarchi, S. Layer-specific approximate multipliers for energy-precision trade-offs in convolutional neural networks. *IEEE Trans. Comput./Sci. Rep.* **2025**, *15*, 39482. [\[CrossRef\]](#)
47. Omidian, F.; Abdi, A.; Hamed-Rouhbakhs, A. AdApTS: Adaptive approximate computing-based traffic sign recognition unit for self-driving cars. *J. Supercomput.* **2025**, *81*, 1339. [\[CrossRef\]](#)
48. Kouris, A.; Venieris, S.I.; Rizakis, M.; Bouganis, C.-S. Approximate LSTMs for time-constrained inference: Enabling fast reaction in self-driving cars. *arXiv* **2019**, arXiv:1905.00689.

49. Omidian, F.; Abdi, A. Appsign: Multi-level approximate computing for real-time traffic sign recognition in autonomous vehicles. *arXiv* **2024**, arXiv:2411.10988.
50. Katare, D.; Perino, D.; Nurmi, J.; Warnier, M.; Janssen, M.; Ding, A.Y. A survey on approximate edge AI for energy efficient autonomous driving services. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 2714–2754. [[CrossRef](#)]
51. Tang, C.; Wei, X.; Zhu, C.; Wang, Y.; Jia, W. Mobile vehicles as fog nodes for latency optimization in smart cities. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9364–9375. [[CrossRef](#)]
52. Tang, Y.; Zhou, H.; Ji, Z.; Wang, C.-L. Cube-fx: Mapping Taylor Expansion Onto Matrix Multiplier-Accumulators of Huawei Ascend AI Processors. *IEEE Trans. Parallel Distrib. Syst.* **2025**, *36*, 1115–1129. [[CrossRef](#)]
53. Shin, P.; Hong, S. Performance Characterization of Deep Learning Primitives for Hardware-Aware Mapping on Heterogeneous Edge Accelerators. In *Proceedings of the 2025 IEEE/IEIE International Conference on Consumer Electronics-Asia (ICCE-Asia), Busan, Republic of Korea, 27–29 October 2025*; IEEE: New York, NY, USA, 2005; pp. 1–6.
54. Fan, M.; Kong, X.; Xu, S.; Xiong, H.; Liu, X. Video-based traffic light recognition by rockchip RV1126 for autonomous driving. In *Proceedings of the 2025 IEEE Intelligent Vehicles Symposium (IV), Cluj-Napoca, Romania, 22–25 June 2025*; IEEE: New York, NY, USA, 2025; pp. 1738–1744.
55. Chen, Y.-H.; Emer, J.; Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 367–379. [[CrossRef](#)]
56. Li, Y.; Liu, W.; Liu, Q.; Zheng, X.; Sun, K.; Huang, C. Complying with iso 26262 and iso/sae 21434: A safety and security co-analysis method for intelligent connected vehicle. *Sensors* **2024**, *24*, 1848. [[CrossRef](#)] [[PubMed](#)]
57. Zhang, Y.; Zhao, K.; Yang, Y.; Zhou, Z. Real-time service migration in edge networks: A survey. *J. Sens. Actuator Netw.* **2025**, *14*, 79. [[CrossRef](#)]
58. Bilal, M.A.; Islam, I.U.; Iltaf, N.; Khan, M.J.; Khan, J. Federated learning with explainable ai for malicious traffic detection in iot networks. *IEEE Access* **2025**, *13*, 173368–173383. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.