

A Hardware–Software Co-Designed Lightweight CNN–MLP Accelerator for CAN Intrusion Detection on FPGA

Manel Hentati

Sfax University

Raida Hentati

`raida.hentati@enetcom.usf.tn`

Sfax University

Olfa Gaddour

Sfax University

Research Article

Keywords:

Posted Date: October 23rd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-8902085/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

A Hardware–Software Co-Designed Lightweight CNN–MLP Accelerator for CAN Intrusion Detection on FPGA

Manel Hentati^{1,2*}, Raida Hentati^{1,2} and Olfa Gaddour^{1,3}

^{1*}CES Laboratory, Sfax University, Tunisia.

²National School of Electronics and Telecommunications of Sfax, Sfax University, Tunisia.

³National School of Engineers of Sfax, Sfax University, Tunisia.

*Corresponding author(s). E-mail(s): manel.hentati@enetcom.usf.tn;
Contributing authors: raida.hentati@enetcom.usf.tn;
olfa.gaddour@enis.tn;

Abstract

The increasing connectivity of modern vehicles exposes CAN buses to critical cybersecurity threats. While deep learning-based intrusion detection systems (IDS) achieve high detection accuracy, their deployment in embedded automotive platforms remains constrained by strict latency, power, and resource limitations. Existing FPGA-based implementations either prioritize detection performance without hardware-aware optimization or rely on aggressive quantization strategies that may compromise numerical robustness. As a result, the trade-off between detection accuracy and hardware efficiency remains insufficiently characterized. This paper proposes a hardware-software co-designed CAN intrusion detection system targeting an experimentally validated balance between classification performance and FPGA implementation cost. A compact hybrid CNN-MLP architecture is jointly optimized with a heterogeneous fixed-point quantization strategy and implemented using a fully pipelined streaming dataflow on a Xilinx Zynq-7020 FPGA-SoC. Experimental results on publicly available CAN intrusion datasets demonstrate a detection accuracy of 99.28% while achieving an inference latency of 0.138 ms and a power consumption of 1.793 W. Comparative analysis against others FPGA-based approaches shows that the proposed design provides a more favorable hardware-accuracy efficiency trade-off, without relying on extreme quantization or complex sequential architectures. These results confirm that high detection performance and embedded hardware efficiency can be jointly achieved through an integrated co-design

methodology, making the proposed solution suitable for real-time automotive cybersecurity applications.

Keywords: CAN Bus Security; Intrusion Detection System (IDS); Embedded Artificial Intelligence; FPGA Acceleration; Hardware–Software Co-Design; Lightweight Deep Learning; CNN–MLP Architecture.

1 Introduction

Modern vehicles have evolved into highly interconnected cyber-physical systems integrating dozens of Electronic Control Units (ECUs) that exchange critical information through in-vehicle communication networks [1]. Among these networks, the CAN remains the dominant protocol due to its simplicity, deterministic arbitration mechanism, and low implementation cost. CAN supports real-time communication for safety-critical functions such as braking, steering, and powertrain control [2].

Despite its architectural robustness, the CAN protocol was not originally designed with cybersecurity mechanisms such as authentication, encryption, or access control [3]. As a consequence, any node with bus access can inject malicious messages, enabling attacks such as denial-of-service (DoS), fuzzy injection, or ECU impersonation. These vulnerabilities pose significant safety risks and have motivated extensive research on intrusion detection systems (IDS) tailored to in-vehicle networks [4][5].

Machine learning and deep learning techniques have demonstrated strong potential for CAN intrusion detection. Convolutional neural networks (CNNs), recurrent neural networks (RNNs), and attention-based architectures have demonstrated strong intrusion detection performance on publicly available datasets, owing to their ability to capture complex temporal dependencies and structural patterns within network traffic [6]. However, most of these approaches are evaluated in software environments using CPUs or GPUs, without explicitly addressing the strict hardware constraints imposed by automotive embedded platforms.

Deploying AI-based IDS directly within vehicle electronic architectures introduces several challenges. Embedded systems must satisfy deterministic latency requirements, limited memory availability, constrained power budgets, and strict reliability standards [7]. While CPUs and GPUs offer flexibility and high raw performance, they often suffer from non-deterministic timing behavior and higher energy consumption. Microcontrollers, although energy efficient, typically lack sufficient computational capacity to sustain real-time deep learning inference. In contrast, Field Programmable Gate Arrays (FPGAs) provide fine-grained parallelism, predictable execution timing, and favorable energy efficiency, making them attractive candidates for embedded AI acceleration in safety-critical automotive environments.

Recent research has explored FPGA-based implementations of CAN IDS. Some works directly map neural network models onto programmable logic with limited architectural modification. Others adopt aggressive compression strategies such as binarized neural networks (BNNs) or heavily pruned architectures [8] to reduce hardware cost. Although these approaches improve implementation feasibility, they often prioritize

either detection accuracy or hardware efficiency in isolation. Consequently, the fundamental trade-off between detection performance and hardware resource utilization remains insufficiently characterized.

This observation motivates the central question addressed in this work:

Can a jointly optimized neural architecture, numerical representation, and FPGA streaming dataflow achieve near state-of-the-art intrusion detection accuracy while significantly improving hardware efficiency under embedded automotive constraints?

To address this question, this paper proposes a hardware–software co-designed FPGA-based intrusion detection system for CAN bus security. Rather than treating the neural network as a purely software artifact later translated into hardware, the proposed approach adopts an integrated design methodology in which:

- the model architecture is explicitly selected for hardware-friendly parallelism,
- numerical precision is optimized through heterogeneous fixed-point quantization,
- and the accelerator is implemented using a fully streaming AXI4-Stream dataflow with DMA-based communication.

A compact hybrid CNN–MLP architecture is designed to capture short-term temporal dependencies in CAN traffic while avoiding the sequential computational overhead of recurrent models. The model is mapped onto a Xilinx Zynq-7020 FPGA-SoC using an HLS-based workflow, enabling tight integration between the processing system and programmable logic.

The contributions of this work are summarized as follows:

- A holistic hardware–software co-design methodology for CAN intrusion detection targeting a measurable accuracy–hardware trade-off.
- A lightweight CNN–MLP architecture optimized for FPGA deployment through heterogeneous fixed-point quantization.
- A fully pipelined streaming accelerator architecture integrated via AXI-DMA for deterministic low-latency inference.
- A complete FPGA implementation on a Xilinx Zynq-ZC702 platform with detailed evaluation of accuracy, latency, throughput, resource utilization, and power consumption.
- An experimental analysis demonstrating that high detection performance and hardware efficiency can be jointly achieved within a unified embedded framework.

The remainder of this paper is organized as follows. Section 2 reviews related work and identifies the research gap addressed by this study. Section 3 presents the proposed hardware–software co-design methodology and system architecture. Section 4 reports experimental results, including both detection performance and FPGA implementation metrics. Section 5 discusses limitations and future directions, and Section 6 concludes the paper.

2 Related Work and Research Gap

Intrusion Detection Systems (IDS) for CAN bus have been extensively studied to mitigate the intrinsic security limitations of in-vehicle networks, notably the absence of built-in authentication and encryption mechanisms. Early research primarily focused

on software-based machine learning techniques deployed on general-purpose processors. More recent studies, however, have shifted toward hardware-aware designs capable of satisfying the stringent real-time, latency, power, and resource constraints imposed by automotive embedded environments. In this context, FPGA-based IDS implementations have attracted growing attention due to their inherent parallelism, deterministic low-latency processing, and favorable energy efficiency.

2.1 CAN Bus Intrusion Detection Approaches

Initial CAN IDS solutions predominantly relied on traditional machine learning and deep learning models implemented in software. Javed *et al.* [9] proposed CANIntelli-IDS, which integrates convolutional attention mechanisms with gated recurrent units (GRU) to capture both spatial and temporal characteristics of CAN traffic. Moulahi *et al.* [10] conducted a comparative study of conventional classifiers, including support vector machines (SVM) and decision trees (DT), demonstrating their effectiveness when applied to engineered CAN features. More recently, Rai *et al.* [11] evaluated multiple recurrent neural network architectures alongside a deep CNN-based VGG16 model to learn temporal attack patterns in CAN messages.

Overall, deep learning approaches such as convolutional neural networks (CNN), recurrent neural networks (RNN), and hybrid architectures have achieved high detection accuracy on public CAN intrusion datasets. Nevertheless, these models are typically evaluated on high-performance computing platforms without accounting for embedded deployment constraints, including strict latency requirements, limited memory capacity, and power consumption budgets. As a result, despite their strong detection performance, the direct deployment of such software-based solutions in real-time automotive systems remains impractical.

2.2 FPGA-Based Neural IDS Implementations

To overcome the limitations of software-only approaches, several studies have explored lightweight IDS implementations on embedded hardware platforms. Khandelwal *et al.* [12] proposed a unified ECU architecture that integrates a CAN IDS alongside conventional ECU functionalities. Their system employs two quantized multilayer perceptron (QMLP) models to detect denial-of-service, fuzzy, and spoofing attacks. Implemented on a hybrid FPGA platform and accelerated using Xilinx’s Deep Learning Processing Unit (DPU), the solution demonstrates low computational overhead and practical feasibility for in-vehicle deployment. However, the reliance on vendor-specific accelerators and proprietary inference engines may constrain architectural flexibility and limit portability across heterogeneous embedded platforms.

To further reduce hardware complexity and power consumption, Rangasikunpam *et al.* investigated FPGA-based IDS architectures leveraging binarized and ultra-lightweight neural networks. In [13], the authors proposed a Binarized Neural Network (BNN) combined with a two-stage Coarse-to-Fine (C2F) detection framework. The Coarse stage rapidly filters benign traffic, while the Fine stage performs detailed attack classification only when suspicious behavior is detected. Implemented on a low-cost

FPGA, the architecture achieves high detection accuracy while maintaining real-time performance and low power consumption.

In a subsequent study [14], the same authors introduced an optimized variant targeting ultra-low-power automotive applications. By applying network pruning techniques within the BNN-based C2F framework, the number of model parameters was reduced by 91.07% with only a marginal degradation in detection accuracy. A comparative evaluation across CPU, GPU, and FPGA platforms demonstrated the superior efficiency of the FPGA implementation in terms of latency and energy consumption. Despite these advantages, BNN-based solutions may suffer from limited numerical precision, potentially affecting generalization capability and robustness under complex or evolving attack scenarios.

An alternative design paradigm combines lightweight processors with hardware acceleration to balance flexibility and efficiency. Liu *et al.* [15] proposed a CAN IDS deployed on an FPGA-based RISC-V platform using the energy-efficient Zero-Riscy core. Instead of fully mapping the neural network onto custom hardware, a BNN is executed on the embedded processor, while performance is enhanced through the introduction of custom instructions for convolution, pooling, and fully connected operations. This approach significantly reduces memory access overhead and energy consumption while preserving real-time detection capability.

Similarly, Xu *et al.* [16] introduced MULSAM, an IDS architecture that combines multidimensional LSTM (MD-LSTM) with a self-attention mechanism to improve detection accuracy when operating on small data batches. By exploiting parallelism and implementing the design on an FPGA platform, the authors demonstrated the feasibility of deploying more expressive neural architectures in embedded automotive environments.

2.3 Limitations of Existing FPGA-Based Intrusion Detection Designs

Despite significant progress in FPGA-based intrusion detection systems (IDS) for CAN bus security, several limitations remain when considering embedded deployment constraints.

Many existing approaches directly translate pre-trained neural network models into FPGA implementations with minimal architectural adaptation. While this strategy simplifies deployment, it often neglects hardware–software co-optimization principles. As a result, the generated accelerators may suffer from inefficient resource utilization, increased latency, and suboptimal energy efficiency. The absence of early-stage hardware-aware design limits the achievable trade-off between detection accuracy and implementation cost.

To reduce hardware complexity, some studies rely on aggressive quantization strategies, particularly binarized neural networks (BNNs). Although these methods significantly decrease memory usage and computational requirements, they may introduce numerical instability and reduce model robustness when exposed to complex or evolving attack patterns. The loss of representational precision can also limit generalization capability, especially in heterogeneous vehicular environments.

Recurrent neural networks, including LSTM and GRU models, have demonstrated strong detection performance due to their ability to capture temporal dependencies. However, their sequential computation patterns introduce challenges for efficient FPGA implementation. Limited parallelism, increased control complexity, and higher inference latency reduce their suitability for deterministic, low-latency embedded systems where real-time processing is critical.

Collectively, these limitations highlight the need for a holistic co-design approach that jointly optimizes neural architecture, numerical precision, and hardware dataflow to achieve balanced detection performance and embedded hardware efficiency.

2.4 Positioning of the Proposed Work

To address the identified limitations, this work introduces a hardware–software co-designed intrusion detection framework specifically tailored for real-time CAN bus monitoring on FPGA platforms.

A structured comparison with prior FPGA-based IDS solutions is provided through a comprehensive evaluation table 1 summarizing model types, deployment strategies, quantization methods, and hardware performance metrics. This comparative analysis reveals that existing works typically prioritize either detection accuracy or hardware efficiency in isolation. In contrast, the proposed framework targets a measurable balance between both dimensions by integrating architectural simplicity, heterogeneous fixed-point quantization, and streaming-oriented accelerator design.

The central hypothesis guiding this work is that a compact hybrid CNN–MLP architecture, combined with heterogeneous fixed-point precision and a fully pipelined streaming FPGA implementation, can achieve near state-of-the-art detection accuracy while significantly improving latency, power efficiency, and hardware resource utilization. Rather than treating the neural network as a static software artifact, the proposed methodology adopts a unified co-design process in which model architecture, numerical representation, and hardware dataflow are jointly optimized from the early design stages.

Through this positioning, the proposed approach aims to demonstrate that high detection performance and embedded hardware efficiency are not mutually exclusive objectives, but can be simultaneously achieved within a unified FPGA-based intrusion detection framework.

3 Hardware–Software Co-Design Methodology

This study proposes a hardware-accelerated IDS for monitoring of CAN traffic and detection of malicious behavior in vehicle networks. The IDS is designed to identify common CAN-based attack categories, including denial-of-service (DoS), fuzzy injection, and impersonation attacks. To achieve an effective balance between detection accuracy and implementation efficiency, a lightweight hybrid CNN–MLP architecture is adopted, making the proposed solution well suited for deployment in resource-constrained automotive embedded environments.

Table 1: Comparison of Software-Based and FPGA-Based CAN Intrusion Detection Systems

Work	Model Type	Platform	Quantization	HW Metrics	Limitations
Khandelwal <i>et al.</i> [12]	QMLP	FPGA + DPU	INT8	Latency, Power	Vendor-specific accelerator
Rangsikunpum <i>et al.</i> [13]	BNN (C2F)	FPGA	Binary	Latency, Power	Limited precision, robustness
Rangsikunpum <i>et al.</i> [14]	Pruned BNN (C2F)	FPGA	Binary	Latency, Energy	Accuracy degradation risk
Liu <i>et al.</i> [15]	BNN + RISC-V	FPGA	Binary	Energy, Latency	Software-driven inference
Xu <i>et al.</i> [16]	MD-LSTM + Attention	FPGA	Fixed-point	Latency	High architectural complexity
This work	CNN-MLP	FPGA	Heterogeneous fixed-point quantization	Latency, Power, Resources	Balanced accuracy-efficiency trade-off

To satisfy stringent real-time and energy efficiency requirements, the trained IDS model is deployed as a dedicated hardware accelerator on an FPGA-based system-on-chip (SoC). The proposed architecture exploits the heterogeneous computing capabilities of the platform by combining a general-purpose Processing System (PS) with a custom neural network accelerator implemented in the PL. This partitioning enables computationally intensive inference tasks to be offloaded to hardware, thereby reducing latency and power consumption while preserving system-level flexibility.

3.1 System Architecture Overview and Threat Model

The CAN bus is a widely used in-vehicle communication protocol characterized by a multi-master, message-based architecture in which multiple Electronic Control Units (ECUs) share a common broadcast medium [17]. Each transmitted CAN frame includes an identifier (ID) that simultaneously defines message priority and the arbitration process. When multiple ECUs attempt to transmit concurrently, arbitration is resolved deterministically, ensuring that higher-priority messages gain immediate access to the bus while lower-priority transmissions are deferred [18]. This mechanism guarantees reliable and low-latency communication, which is essential for safety-critical automotive applications.

Despite these advantages, the CAN protocol inherently lacks fundamental security features such as message authentication, sender identification, and data encryption.

Consequently, any node with access to the CAN bus can inject arbitrary frames, exposing the network to a wide range of cyberattacks. In practice, adversaries may exploit multiple attack surfaces, including the OBD-II diagnostic interface, infotainment systems, telematics units, or direct physical access to the vehicle wiring [19]. Once connected, an attacker can leverage the broadcast nature of the CAN bus to transmit malicious messages capable of disrupting or manipulating critical vehicle functions, such as braking, acceleration, or steering [20].

3.2 Heterogeneous FPGA-Based Hardware Architecture

Figure 1 presents the system-level dataflow of the proposed FPGA-based IDS, illustrating the interaction between the Processing System, external DDR memory, the AXI Direct Memory Access (DMA) engine, and the custom neural network hardware IP. This dataflow highlights the streaming-based inference mechanism that enables efficient and low-latency processing of CAN traffic.

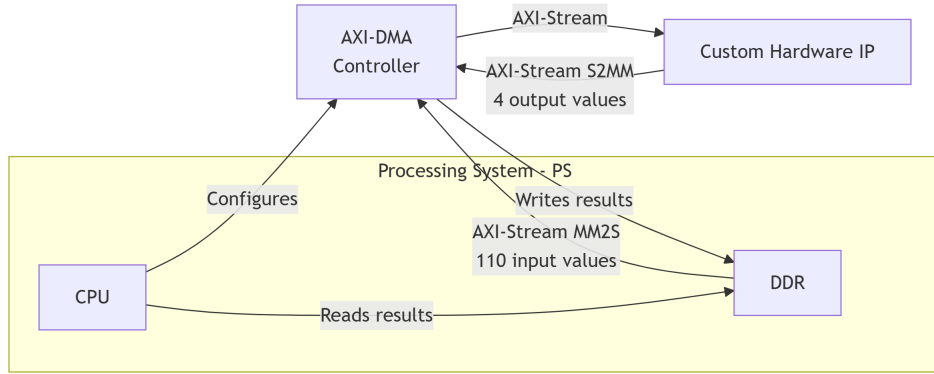


Fig. 1: Block diagram of the proposed FPGA-based CAN intrusion detection system, showing the PS, external DDR memory, AXI-DMA controller, and the custom CNN-MLP hardware accelerator implemented in the PL.

The block diagram illustrates a heterogeneous FPGA-based system-on-chip architecture in which computation and data movement are partitioned between software and hardware domains. The PS executes control oriented tasks, including system configuration, memory management, and coordination of data transfers. Feature vectors extracted from CAN traffic are stored in external DDR memory and streamed to the PL through the AXI-DMA engine [21]. Within the PL, a custom CNN-MLP accelerator performs inference using a fully pipelined and parallel hardware architecture. The classification results are subsequently transferred back to DDR memory and accessed by the PS for decision handling or higher-level system integration.

In the proposed design, the PS is responsible for system initialization, configuration, and control. Specifically, the CPU configures the AXI-DMA engine and the neural

network accelerator IP, allocates memory buffers in external DDR, and orchestrates data transfers between software and hardware components.

During inference, a feature vector composed of 110 input attributes is transferred from DDR memory to the hardware accelerator via the memory mapped to stream (MM2S) channel of the AXI-DMA. The accelerator processes the streamed data and performs neural network inference entirely in hardware. The resulting output vector, containing four class scores corresponding to normal traffic and three attack types, is subsequently written back to DDR memory through the stream-to-memory-mapped (S2MM) channel.

By offloading bulk data movement to the AXI-DMA subsystem, the proposed architecture minimizes CPU intervention during inference execution. This design choice significantly reduces software overhead, improves overall throughput, and enables deterministic low latency operation, an essential requirement for safety-critical automotive systems.

Furthermore, the streaming-based communication model allows the hardware accelerator to operate at full throughput, making efficient use of FPGA parallelism while maintaining compatibility with standard SoC interconnects. This modular design also facilitates future scalability, enabling the integration of additional detection modules or alternative neural network architectures with minimal system-level modifications.

The hardware–software co-design workflow adopted in this work follows a structured deployment pipeline that integrates model development, hardware-aware optimization, and FPGA implementation within a unified design process. As illustrated in Figure 2, the workflow ensures consistency between software-level model training and hardware-level accelerator generation through automated high-level synthesis.

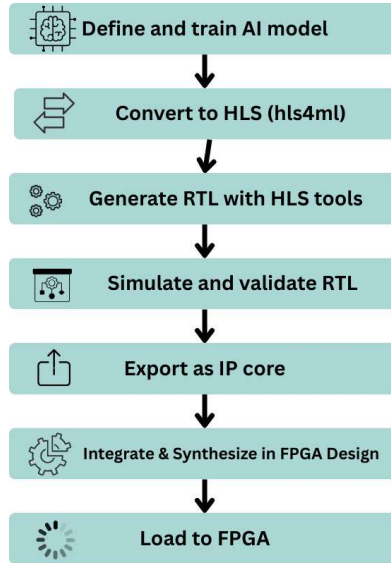


Fig. 2: Design flow for deploying the proposed IDS on FPGA

By tightly coupling model development with hardware generation, this workflow enables early hardware-aware optimization and ensures an efficient transition from software training to real-time FPGA deployment. section Lightweight CNN-MLP Architecture Design and Optimization

3.3 OTIDS Dataset Description and Preprocessing Strategy

The OTIDS dataset, introduced by Lee *et al.* [22], is adopted in this work due to its well-structured labeling, reproducibility, and suitability for benchmarking lightweight intrusion detection systems. The dataset provides clearly defined attack scenarios and controlled acquisition conditions, enabling systematic evaluation while maintaining computational feasibility for embedded and FPGA-based platforms. Although more recent datasets incorporate increasingly sophisticated attack vectors, OTIDS offers a practical trade-off between realism and implementation complexity for the proposed hardware-oriented IDS framework. The evaluation will be extended to newer datasets in future work to assess generalization capability.

The dataset was collected by logging CAN traffic through the OBD-II interface of a production vehicle (KIA SOUL) under both normal driving conditions and controlled message injection attacks. It includes four traffic classes: normal (free state), denial-of-service (DoS), fuzzy, and impersonation attacks [23]. Figure 3 illustrates the operational mechanisms of the three considered attack types.

- **DoS attack:** flooding the CAN bus with high-priority messages (e.g., CAN ID 0x000), thereby blocking legitimate traffic.
- **Fuzzy attack:** injection of messages with random or undefined CAN identifiers, commonly used to probe ECU behavior.
- **Impersonation attack:** transmission of forged messages using the arbitration ID of a legitimate ECU (e.g., CAN ID 0x164).

These attack scenarios represent critical security threats to in-vehicle networks, as they compromise system integrity and may directly impact functional safety.

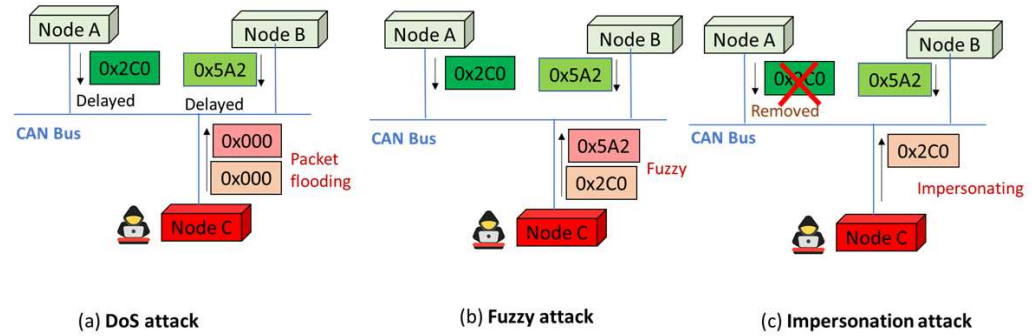


Fig. 3: Illustration of denial-of-service, fuzzy, and impersonation attack scenarios on the CAN bus.

Table 2: Class distribution of the OTIDS dataset

Traffic class	Samples
Normal traffic	2,369,868
Impersonation attack	995,472
DoS attack	656,579
Fuzzy attack	591,990

The OTIDS dataset provides 12 features per CAN frame, including the timestamp, data length code (DLC), and eight data bytes. In this work, 11 features are retained, while the CAN identifier is excluded to reduce input dimensionality and hardware complexity.

- **Timestamp:** recorded time in seconds.
- **DLC:** number of data bytes (0–8).
- **DATA[0–7]:** payload bytes of the CAN frame.

Table 2 summarizes the distribution of samples across the four traffic classes.

The dataset exhibits significant class imbalance, which can negatively affect learning performance. Furthermore, preserving the temporal order of CAN frames is essential, as DoS, fuzzy, and impersonation attacks manifest as time-dependent patterns rather than isolated events.

To address class imbalance while maintaining temporal consistency, the Synthetic Minority Oversampling Technique (SMOTE) [24] is applied to augment minority classes. SMOTE generates synthetic samples through interpolation between neighboring instances, improving class representation without altering the underlying feature distribution. This strategy enhances model robustness while remaining compatible with the sequential nature of CAN traffic.

3.4 Model Training Procedure and Hyperparameter Configuration

A compact hybrid CNN-MLP architecture is proposed to balance detection accuracy and hardware efficiency. As illustrated in Figure 4, the model processes short temporal windows composed of 10 consecutive CAN frames, enabling the extraction of time-localized intrusion patterns with minimal computational overhead.

The feature extraction stage consists of two one-dimensional convolutional layers. The first layer employs 12 filters to capture local temporal dependencies, while the second layer uses 24 filters with a stride of 2 to down sample the temporal dimension and extract higher level representations. A max pooling layer further reduces feature dimensionality, followed by a flattening operation.

The resulting feature vector is processed by a lightweight multilayer perceptron comprising a hidden layer with 24 neurons and ReLU activation, followed by a Softmax output layer that classifies the input sequence into one of four categories: normal,

Table 3: Training and model configuration parameters for the proposed CNN–MLP

Parameter	Value
Input window length	10 CAN frames
Number of input features	11
Batch size	32
Maximum training epochs	100
Early stopping patience	10 epochs
Optimizer	Adam
Learning rate	1×10^{-4}
Loss function	Categorical cross-entropy
Hidden layer activation	ReLU
Output activation	Softmax
Number of output classes	4

DoS, fuzzy, or impersonation. This architecture combines efficient temporal feature extraction with a low-complexity decision stage, making it well suited for real-time inference on embedded FPGA platforms.

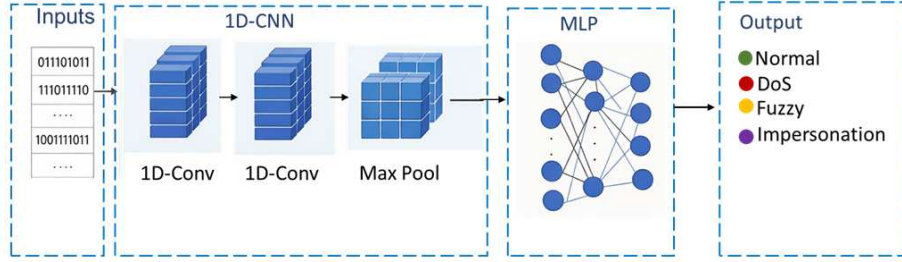


Fig. 4: Proposed lightweight CNN–MLP architecture for CAN bus intrusion detection.

The CNN-MLP model is trained using the Adam optimizer with a learning rate of 10^{-4} and categorical cross-entropy loss. A batch size of 32 is selected to achieve a balance between convergence stability and memory efficiency. Training is performed for a maximum of 100 epochs, with early stopping [25] applied after 10 consecutive epochs without validation improvement to prevent overfitting. During training and inference, the model operates on fixed-length temporal windows of 10 consecutive CAN frames represented by 11 input features, and produces probabilistic outputs over four traffic classes. A summary of the training and model configuration parameters is provided in Table 3.

Table 4: Hardware Specifications of the Xilinx ZC702 FPGA Platform

Specification	Description
FPGA device	Zynq-7000 XC7Z020-1CLG484
Processing system	Dual-core ARM Cortex-A9
Programmable logic resources	~85K logic cells, 560 KB BRAM, 220 DSP slices
Slice count	~13.3K slices
External memory	1 GB DDR3, SD card interface
Python support	No native PYNQ support
Development toolchain	Vivado, Vitis, Xilinx SDK
Target application	Embedded AI acceleration, HLS, AXI-DMA-based systems

3.5 FPGA Accelerator Architecture and Implementation

This section describes the complete hardware-oriented deployment flow of the proposed intrusion detection system (IDS) on an FPGA-based platform. It details the adopted development toolchain, the target FPGA-SoC architecture, and the experimental setup used for implementation and validation. Furthermore, the end-to-end deployment workflow is summarized through a flowchart that outlines the sequential stages required to map the trained IDS model onto the FPGA fabric.

3.5.1 Development Toolchain and Hardware Platform

The FPGA implementation of the proposed CAN bus IDS was carried out using the *hls4ml* framework . This framework enables the translation of trained neural network models into synthesizable C++ descriptions and automatically generates a Vivado HLS project tailored for FPGA deployment. High-Level Synthesis (HLS), Register Transfer Level (RTL) generation, and system integration were performed using the Xilinx Vivado Design Suite [26].

For hardware deployment, the Xilinx ZC702 evaluation board was selected as the target platform [27]. The board is based on the Zynq-7000 XC7Z020 System-on-Chip (SoC), which tightly integrates a dual-core ARM Cortex-A9 PS with Artix-7 class programmable logic (PL). This heterogeneous architecture enables efficient hardware–software co-design, where computationally intensive inference tasks are offloaded to custom FPGA accelerators, while control and data management are handled by the embedded ARM processors.

The ZC702 platform provides a balanced combination of logic resources, on-chip memory, DSP slices, and external DDR3 memory, making it well suited for real-time CAN intrusion detection under embedded and automotive-like constraints. Its support for AXI-based interconnects and Direct Memory Access (DMA) further facilitates high-throughput data exchange between the PS and PL, which is essential for low-latency IDS inference.

Table 4 summarizes the main hardware specifications of the ZC702 FPGA-SoC used in this work.

3.5.2 Hardware Implementation and Accelerator Integration

The hardware implementation process translates the optimized IDS model into a streaming FPGA accelerator integrated within a heterogeneous embedded system. The implementation methodology focuses on achieving low-latency inference, efficient resource utilization, and deterministic execution suitable for real-time CAN bus monitoring.

Model Conversion and Hardware-Aware Optimization

The trained neural network model is converted into a hardware-compatible description using an automated high-level synthesis conversion workflow `hls4ml` [28]. During this stage, hardware-aware optimizations are introduced, including fixed-point quantization [29] and architectural simplification, to reduce computational complexity and memory requirements while preserving detection accuracy.

Heterogeneous numerical precision is adopted across network layers to balance hardware efficiency and numerical stability. Resource sharing and pipeline configuration parameters are jointly optimized to achieve an effective trade-off between latency, throughput, and hardware footprint. The resulting hardware description forms the basis for subsequent synthesis and integration steps.

Hardware Generation and Streaming Accelerator Architecture

Following conversion, the hardware description is synthesized into a register-transfer level (RTL) implementation representing a pipelined neural inference engine. The accelerator adopts a streaming dataflow architecture that enables continuous processing of input feature vectors with minimal control overhead.

Standardized streaming interfaces are employed to support high-throughput data transfer between system components. The design emphasizes parallel execution of neural network layers while maintaining a modular architecture that facilitates scalability and integration into larger embedded systems.

Verification and IP Core Integration

Functional verification is performed to ensure consistency between software inference results and the hardware implementation. The validated accelerator is packaged as a reusable IP core compliant with standard interface protocols, enabling seamless integration into a heterogeneous FPGA-SoC system.

The modular IP-based design approach promotes reusability and simplifies system assembly by allowing the accelerator to be incorporated into a broader hardware-software architecture using standardized interconnect mechanisms.

System Integration and Deployment

The accelerator IP core is integrated into a complete embedded system composed of the processing system, memory interfaces, and communication infrastructure. The resulting architecture enables efficient data exchange between software-managed control tasks and hardware-accelerated inference operations.

After synthesis and implementation, the final configuration bitstream is deployed onto the FPGA platform. A lightweight embedded software layer orchestrates accelerator execution and manages data transfers between memory and hardware modules. Experimental validation confirms correct end-to-end functionality and demonstrates real-time IDS operation under representative CAN traffic conditions. The resulting embedded IDS architecture for CAN bus intrusion detection is illustrated in Figure 5. The block diagram highlights the modularity and simplicity of the proposed design, which relies on a minimal set of IP cores to achieve efficient hardware utilization. This streamlined architecture reduces integration complexity and contributes to improved timing closure and resource efficiency.

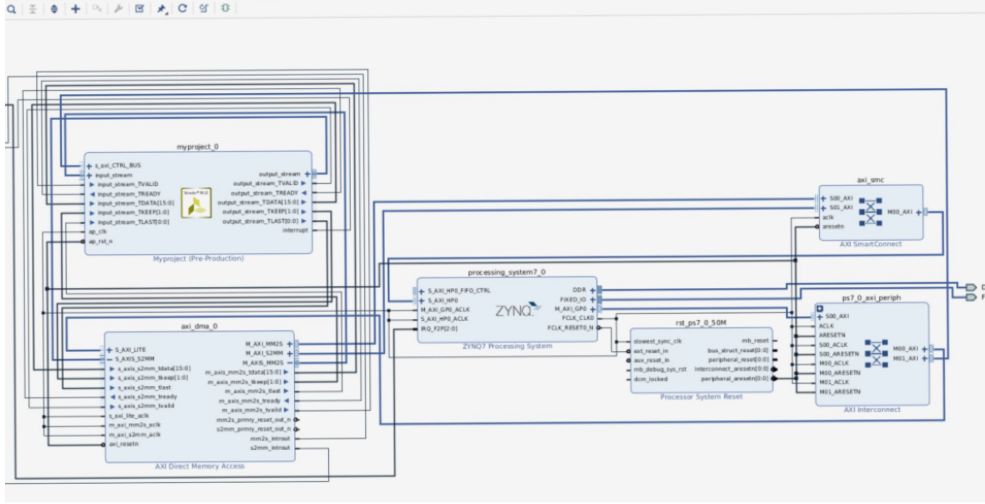


Fig. 5: Hardware-software co-design architecture of the proposed FPGA-based CAN bus intrusion detection system.

After deployment, the application is executed directly on the FPGA-SoC platform. CAN feature vectors are streamed to the accelerator through the DMA interface, and the corresponding inference results are retrieved and displayed via the SDK console. This final step validates the end-to-end functionality of the proposed IDS under real hardware conditions, confirming correct interaction between software control logic and the FPGA-based neural network accelerator.

Table 5: Classification performance of the CNN-MLP model on the OTIDS test set

Class	Precision	Recall	F1-score
Normal traffic	0.98	0.99	0.97
DoS attack	0.97	0.98	0.97
Fuzzy attack	1.00	0.99	0.99
Impersonation attack	1.00	1.00	1.00
Macro average	0.99	0.99	0.98

4 Experimental Performance Evaluation of FPGA-Based Intrusion Detection Systems

This section presents the implementation results, including a comparison with related work, followed by experimental deployment results, evaluated using standard AI performance metrics.

4.1 Software-Level Detection Performance Evaluation

After completing the training phase, the proposed CNN-MLP model is evaluated on an independent test set to assess its classification performance. Standard metrics, including precision, recall, and F1-score [30], are used to quantify detection accuracy across normal traffic and the three considered attack classes. These metrics provide a comprehensive view of the model’s ability to correctly identify intrusions while minimizing false alarms, which is critical for real-time deployment in safety-critical automotive environments.

Table 5 summarizes the classification performance of the proposed CNN-MLP model across the four traffic classes using precision, recall, and F1-score metrics. The results indicate consistently high detection accuracy for both normal traffic and all considered attack types, demonstrating the effectiveness of the proposed model in distinguishing malicious CAN messages from benign traffic.

Figure 6 illustrates the training and validation accuracy and loss of the proposed model over successive epochs. The convergence behavior shows stable learning dynamics with no significant divergence between training and validation curves, indicating effective generalization and limited overfitting. This behavior confirms that the selected model capacity and training configuration are well aligned with the dataset characteristics and the targeted lightweight deployment.

Figure 7 presents the confusion matrix obtained on the test dataset. The matrix shows a strong diagonal dominance, confirming that the model correctly classifies the vast majority of samples across all classes. Misclassification rates remain marginal, particularly for attack classes, which is essential for reducing false alarms and missed detections in real-time in-vehicle intrusion detection systems.

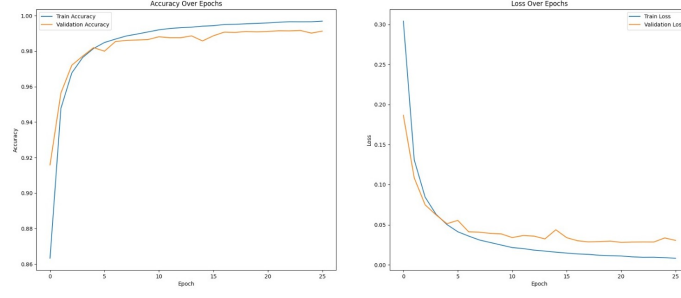


Fig. 6: Training and validation accuracy and loss of the proposed CNN-MLP model.

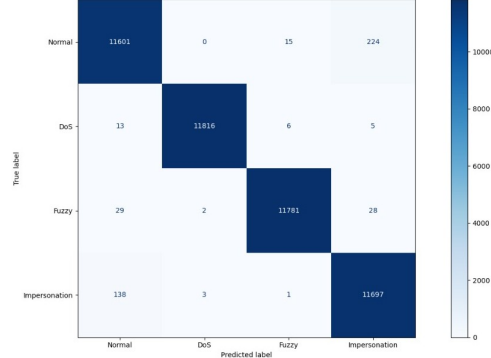


Fig. 7: Confusion matrix of the CNN-MLP model evaluated on the OTIDS test set.

4.2 Comparison with Existing CAN IDS Approaches

To position the proposed model with respect to existing CAN intrusion detection approaches, we first compare its software-level classification performance against representative machine learning and deep learning models evaluated on the OTIDS dataset. This comparison focuses exclusively on detection capability metrics, including precision, recall, F1-score, and accuracy, under a software-based evaluation setting. It aims to demonstrate that the proposed lightweight CNN-MLP architecture achieves competitive classification performance prior to hardware deployment. Hardware efficiency, real-time inference behavior, and FPGA implementation constraints are evaluated separately in the subsequent experimental sections dedicated to embedded deployment analysis.

As shown in Table X, the proposed CNN-MLP model achieves competitive detection performance compared to both traditional machine learning classifiers and more complex deep learning architectures. Despite its compact structure and reduced computational complexity, the model maintains high precision, recall, and overall accuracy, confirming its suitability for embedded deployment. It is important to note that the models listed in this table were primarily evaluated in software environments and do not necessarily consider hardware constraints such as latency, power consumption, or

Table 6: Software-Level Detection Performance Comparison on OTIDS Dataset

Reference Loss	Model	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
[9]	CNN + AGRU	93.69	93.91	93.79	–
–					
[10]	SVM	97.28	96.55	–	97.28
–					
	DT	98.19	98.16	–	98.19
–					
[11]	LSTM	100.00	99.23	99.93	98.07
0.03					
	GRU	92.88	85.07	82.16	92.88
0.08					
	Bi-LSTM	73.58	85.11	78.90	93.37
0.03					
	VGG16	94.37	89.95	90.86	89.95
0.01					
This work 0.027	Lightweight CNN-MLP	98.9	99.5	99.2	99.28

resource utilization. Therefore, a dedicated FPGA-based experimental evaluation is conducted in the following sections to analyze the hardware-accuracy trade-off and assess real-time embedded performance under practical deployment conditions. In contrast, the proposed lightweight CNN-MLP architecture achieves a favorable trade-off between detection accuracy and computational efficiency, attaining an accuracy of 99.28% and an F1-score of 99.2%. These results demonstrate that the proposed design preserves high intrusion detection performance while maintaining a compact and hardware-friendly model structure, making it well suited for low-latency and energy-efficient in-vehicle intrusion detection on FPGA-based platforms.

4.3 Evaluation of the IDS Deployment on the Xilinx Zynq-ZC702 FPGA

This subsection evaluates the runtime behavior of the proposed intrusion detection system after deployment on the Xilinx Zynq-ZC702 FPGA platform. The trained CNN-MLP model is executed as a hardware accelerator within the programmable logic, while inference control and result handling are managed by the processing system.

Figure 8 illustrates representative inference outputs obtained from the deployed IDS when processing CAN bus traffic. For benign traffic samples, the system correctly classifies the input as *Free* with a confidence score of 0.9023. Similar

behavior is observed for malicious traffic, where the accelerator consistently produces high-confidence predictions for denial-of-service, fuzzy, and impersonation attack scenarios.

These results confirm the functional correctness of the FPGA-based implementation and demonstrate the ability of the proposed IDS to reliably perform real-time CAN traffic classification. The tight integration between the processing system, AXI-DMA, and the custom neural network accelerator enables efficient inference execution, making the solution suitable for low-latency and resource-constrained in-vehicle environments.

```

Connected to: Serial ( /dev/ttyUSB0, 115200, 0, 8 )
Input: 110 x int16 (Q6.10) | Output: 4 x int16 (Q6.10)
DMA initialized (simple mode), MM bus = 32-bit, AXIS to IP = 16-bit via DWC.
Packing 110 int16 (Q6.10) as 110 x 16-bit AXIS beats...

-- Inference start (IP AXIS = 16-bit) --
HLS CTRL (pre) = 0x00000004 [start=0 done=0 idle=1 ready=0]
Flushing caches...
Starting HLS IP (ap_start)...
Starting S2MM (PL->PS), 8 bytes...
Starting MM2S (PS->PL), 220 bytes...
Waiting for DMA completion...
Invalidating output cache...

Classification results (Q6.10->float):
Free      : 0.9023 (raw=0x039C)
DoS Attack : 0.0049 (raw=0x0005)
Fuzzy Attack : 0.0117 (raw=0x000C)
Impersonation Attack: 0.0000 (raw=0x0000)

Predicted: Free (value=0.9023)

-- Inference done --
=== All done ===

```

Fig. 8: Inference results of the FPGA-based intrusion detection system for representative benign and attack-related CAN traffic samples.

To evaluate the efficiency of the proposed CNN-MLP intrusion detection accelerator, the synthesized design was implemented on the Xilinx Zynq-ZC702 FPGA platform using Vitis HLS. The resulting performance metrics are summarized in Table ???. The accelerator operates at a clock frequency of 137.5 MHz with an initiation interval (II) of one, enabling the pipeline to accept a new input every clock cycle.

The overall pipeline latency is 1904 clock cycles, corresponding to an inference latency of approximately 13.8 μ s. Owing to the fully pipelined dataflow architecture, the accelerator achieves a sustained inference throughput of 8.24×10^4 inferences per second. These results demonstrate that the proposed hardware accelerator satisfies real-time constraints for CAN bus intrusion detection while maintaining efficient utilization of the FPGA fabric.

The resource utilization results reported in Table 7 indicate that the proposed accelerator maintains a moderate hardware footprint while achieving high inference performance. The design utilizes 18% of BRAM resources and does not require UltraRAM, confirming that large on-chip memory blocks are unnecessary for this implementation. DSP and logic utilization remain within acceptable limits, leaving sufficient headroom for system integration components such as AXI-DMA engines, interconnects, and processing system interfaces.

Table 7: FPGA resource utilization of the proposed IDS accelerator on Zynq-ZC702

Resource	Utilization	Usage (%)
BRAM (18K)	51 blocks	18%
DSP48E	–	18%
Flip-Flops (FF)	–	59%
Lookup Tables (LUT)	–	17%
UltraRAM (URAM)	0	0%

Overall, the synthesized HLS-based accelerator fits comfortably within the available resources of the Zynq-ZC702 FPGA while delivering low-latency and high-throughput inference. These characteristics make the proposed architecture well suited for real-time embedded intrusion detection in automotive CAN networks and provide scalability for future extensions.

FPGA Resource Utilization Analysis

Figure 9 presents the post-implementation resource utilization of the proposed IDS accelerator on the Xilinx Zynq-ZC702 FPGA, providing insight into the efficiency and balance of the hardware mapping.

The utilization of Look-Up Tables (LUTs) reaches 33%, reflecting a moderate consumption of the available logic resources, while flip-flops (FFs) account for 18%, indicating an efficient and well-balanced register allocation. Memory resources are used conservatively, with Block RAM (BRAM) and LUTRAM utilization limited to 14% and 15%, respectively, confirming that the design is not memory-intensive. The highest utilization is observed in Digital Signal Processing (DSP) blocks at 60%, highlighting the arithmetic-centric nature of the CNN–MLP inference pipeline. This is expected given the dominance of convolution and matrix-multiplication operations in the proposed architecture. In contrast, global clock buffer (BUFG) usage remains minimal at 3%, indicating an efficient clock distribution strategy.

Overall, these results demonstrate that the design is DSP-intensive while remaining well within the available FPGA resource limits, leaving sufficient headroom for scalability and the integration of additional system components.

Power Consumption Analysis

To assess the energy efficiency of the proposed FPGA-based IDS, post-implementation power analysis was performed using Xilinx Vivado. The resulting power consumption breakdown is illustrated in Figure 10.

The total on-chip power consumption is measured at 1.793 W, with dynamic power accounting for approximately 92% of the total. The PS dominates the overall power profile, consuming nearly 91%, whereas the PL—including DSPs, LUTs, and BRAM—contributes less than 10%. Notably, DSP blocks account for only about 3%

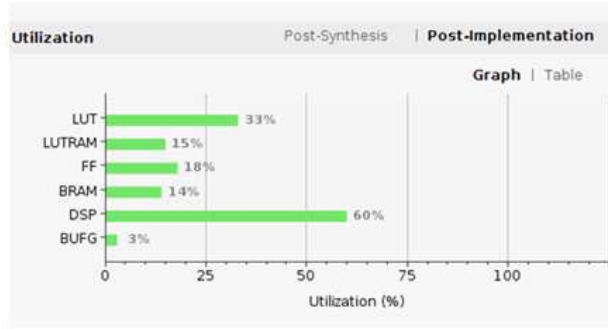


Fig. 9: Post-implementation FPGA resource utilization of the proposed IDS accelerator

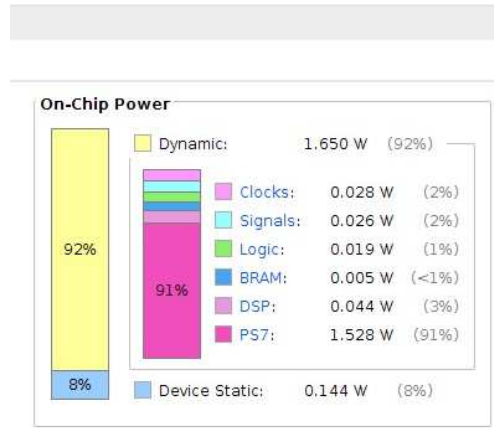


Fig. 10: Post-implementation power consumption report generated by Vivado

of the total power, confirming that the CNN-MLP inference engine mapped to the programmable logic is highly energy-efficient. The junction temperature stabilizes at 45.7 °C, which is well below critical thermal thresholds, ensuring reliable operation without the need for additional cooling mechanisms.

These results demonstrate that the proposed FPGA implementation achieves low overall power consumption, with the majority of energy expenditure attributed to the ARM processing cores rather than the intrusion detection accelerator itself. This characteristic is particularly advantageous for energy-constrained, real-time in-vehicle deployment scenarios.

!

Table 8: Comparison of FPGA-based CAN intrusion detection systems under embedded constraints

Work	Model	Dataset	Platform	Quant.	Acc. (%)	Lat. (ms)	Power (W)	Key Limitation
[12] (2022)	QMLP	Custom CAN	Zynq + DPU	INT8	98.7	0.41	2.6	Vendor-specific accelerator
[13] (2024)	BNN (C2F)	Car-Hacking	Zynq-ZC702	Binary	99.83	0.259	2.29	Extreme quantization
[14] (2025)	Pruned BNN	Car-Hacking	FPGA	Binary	99.5	0.31	1.95	Accuracy degradation risk
[15] (2025)	BNN + RISC-V	Car-Hacking	FPGA SoC	Binary	98.9	0.48	2.1	Software-driven inference
[16] (2025)	MD-LSTM + Att.	Custom CAN	FPGA	Fixed-point	99.6	>0.5	>3.0	High architectural complexity
This work	CNN-MLP	OTIDS	Zynq-ZC702	Het. FP	99.28	0.138	1.793	Slight accuracy gap vs. BNN

Comparison with Related FPGA-Based CAN IDS Solutions

Table 8 summarizes a comparative analysis of representative FPGA-based CAN intrusion detection systems under embedded automotive constraints. The selected baselines cover a range of architectural paradigms, including quantized multilayer perceptrons, binarized neural networks, and recurrent or attention-based models, which are commonly adopted to reduce computational cost or improve detection accuracy. For each approach, the table reports the dataset, target hardware platform, numerical precision, and key performance indicators in terms of accuracy, inference latency, and power consumption. This comparison highlights that prior works typically prioritize either detection performance or hardware efficiency in isolation. In contrast, the proposed CNN-MLP solution achieves a balanced compromise by maintaining near state-of-the-art detection accuracy while significantly reducing latency and power consumption, without relying on extreme quantization or highly complex sequential architectures. The results confirm that a hardware-software co-design strategy is essential to effectively characterize and optimize the accuracy-efficiency trade-off for real-time CAN bus intrusion detection on FPGA platforms.

Hardware-Accuracy Trade-off Analysis

The objective of this work is not to maximize detection accuracy in isolation, but to experimentally demonstrate a balanced trade-off between detection performance and embedded hardware efficiency. As shown in Table 8, existing FPGA-based CAN intrusion detection systems tend to favor either extreme quantization strategies, such as

binarized neural networks, or complex sequential architectures, such as recurrent and attention-based models. While these approaches can achieve very high detection accuracy, they often introduce limitations in terms of numerical robustness, architectural complexity, inference latency, or power consumption.

In contrast, the proposed CNN-MLP architecture adopts a hardware–software co-design methodology combining a compact feed-forward topology, heterogeneous fixed-point quantization, and a fully pipelined streaming FPGA implementation. This design enables near state-of-the-art detection accuracy (99.28%) while achieving significantly reduced inference latency (0.138 ms) and lower power consumption (1.793 W) compared to representative FPGA-based IDS solutions.

Although binarized models may slightly outperform in terms of raw accuracy, the proposed approach avoids extreme quantization and multi-stage detection pipelines, thereby preserving numerical stability and implementation simplicity. These results confirm that high detection performance and embedded hardware efficiency are not mutually exclusive, but can be jointly achieved through an integrated co-design strategy tailored for real-time automotive intrusion detection.

To quantitatively evaluate the trade-off between detection performance and embedded hardware efficiency, we define a composite Hardware–Accuracy Efficiency Index (EI), expressed in Eq. (??). This index jointly captures detection accuracy, inference latency, and power consumption, which are the dominant constraints in real-time in-vehicle intrusion detection systems. By consolidating these metrics into a single figure of merit, EI enables an objective comparison of FPGA-based IDS solutions that differ in architectural complexity, numerical precision, and deployment strategy.

Using the performance metrics reported in Table 8, the proposed CNN-MLP implementation achieves an EI value of

$$EI_{\text{CNN-MLP}} = \frac{99.28}{0.138 \times 1.793} \approx 401.5,$$

whereas the representative BNN-based approach [14] yields

$$EI_{\text{BNN}} = \frac{99.83}{0.259 \times 2.29} \approx 168.3.$$

Despite the marginal difference in raw detection accuracy, the proposed design significantly outperforms binarized solutions in terms of overall efficiency, achieving more than a twofold improvement in EI. This result demonstrates that the proposed hardware–software co-design strategy effectively optimizes the embedded deployment objective by reducing latency and power consumption while preserving near state-of-the-art detection performance. Consequently, the proposed CNN-MLP architecture offers a more favorable and practically relevant accuracy-efficiency trade-off for FPGA-based CAN bus intrusion detection.

5 Discussion

The experimental evaluation demonstrates that the proposed FPGA-accelerated CNN-MLP intrusion detection system achieves a balanced compromise between detection performance and embedded hardware efficiency. Unlike conventional software-centric IDS solutions, which primarily optimize classification accuracy, the proposed framework adopts a hardware-software co-design perspective in which neural architecture, numerical precision, and accelerator dataflow are jointly optimized to satisfy real-time automotive constraints. The obtained results confirm that this integrated design methodology enables deterministic low-latency inference while maintaining high detection accuracy across multiple CAN attack categories.

From a hardware architecture perspective, the compact CNN-MLP topology contributes significantly to the observed efficiency. In contrast to recurrent neural networks, which rely on sequential dependencies that limit parallelism and increase control overhead, the feed-forward convolutional pipeline enables extensive spatial parallelization on FPGA fabric. The fully pipelined streaming dataflow further reduces idle cycles and allows the accelerator to process new inputs continuously, leading to predictable execution timing. This architectural choice explains the low inference latency achieved without requiring aggressive network compression or complex scheduling mechanisms.

The use of heterogeneous fixed-point quantization also plays a key role in balancing numerical robustness and hardware cost. While binarized neural networks can achieve extremely low memory and computational requirements, they may suffer from reduced representational capacity and sensitivity to distribution shifts in CAN traffic. By assigning layer-specific precision levels, the proposed design preserves critical numerical information in sensitive network layers while reducing arithmetic complexity in less critical stages. The resulting implementation demonstrates that moderate quantization strategies can achieve competitive hardware efficiency while maintaining stable detection performance.

A key objective of this work is to experimentally demonstrate the trade-off between detection accuracy and embedded hardware efficiency. Compared with representative FPGA-based CAN IDS approaches, the proposed accelerator achieves near state-of-the-art classification performance while substantially reducing latency and power consumption. The results indicate that architectural simplicity combined with pipeline-oriented execution can provide a more favorable efficiency profile than either highly compressed binarized models or complex recurrent architectures. This observation supports the central hypothesis that hardware-aware neural design can simultaneously address performance and resource constraints rather than treating them as conflicting objectives.

In terms of system integration, embedding the IDS directly within the FPGA programmable logic enables on-board, real-time monitoring of CAN traffic without dependence on external processing units or cloud connectivity. This capability enhances system resilience and reduces communication overhead, which is essential for safety-critical automotive environments. Furthermore, the moderate utilization of memory and logic resources indicates that the architecture retains sufficient

scalability margins for integrating additional detection modules, extended feature representations, or multi-bus monitoring capabilities.

Despite these strengths, several limitations must be acknowledged. First, the experimental validation relies primarily on a single publicly available dataset. Although the dataset provides controlled and reproducible evaluation conditions, it may not fully represent the variability of real-world vehicular environments. Future work should incorporate additional datasets and real vehicle measurements to assess robustness across diverse traffic patterns and attack scenarios. Second, while the accelerator demonstrates strong hardware efficiency, a significant portion of total system power consumption originates from the processing system rather than the programmable logic. Further optimization of PS-PL task partitioning and increased hardware offloading could improve overall energy efficiency. Finally, the current implementation employs a fixed temporal window and feature representation, which simplifies hardware optimization but may reduce adaptability to evolving attack strategies. Future research may explore adaptive architectures and lightweight attention mechanisms to improve temporal modeling without substantially increasing hardware complexity.

Overall, the discussion highlights that the effectiveness of the proposed IDS stems not from a single algorithmic innovation but from the integration of hardware-aware neural architecture design, streaming accelerator implementation, and heterogeneous numerical optimization within a unified co-design framework. The results demonstrate that carefully balancing these elements enables real-time intrusion detection that meets both cybersecurity and embedded deployment requirements in modern automotive systems.

6 conclusion

This work presented a lightweight, FPGA-accelerated intrusion detection system designed for Controller Area Network (CAN) security in automotive environments. By combining a compact CNN-MLP architecture with hardware-aware design choices, the proposed approach achieves accurate detection of DoS, fuzzy, and impersonation attacks while meeting the strict latency and power constraints of in-vehicle embedded platforms.

The proposed model was mapped to a Xilinx Zynq-ZC702 FPGA using a high-level synthesis flow, enabling efficient integration of the neural network accelerator within a heterogeneous processing architecture. Experimental evaluation demonstrates that the resulting system achieves high classification performance with an accuracy of 99.28% and a low loss of 0.027, while maintaining sub-millisecond inference latency and moderate FPGA resource utilization. The measured power consumption of approximately 1.793 W further confirms the suitability of the design for continuous, real-time monitoring in automotive electronic control units.

Beyond individual performance metrics, this work highlights the importance of architectural simplicity and pipeline-oriented execution for achieving deterministic and energy-efficient inference. The proposed framework demonstrates that carefully selected neural topologies combined with moderate quantization and streaming dataflow can provide an effective trade-off between accuracy, latency, and resource

utilization. This contribution extends existing FPGA-based IDS research by emphasizing integrated optimization rather than isolated improvements in either model performance or hardware efficiency.

From an application perspective, the presented IDS architecture is well suited for deployment within electronic control units or automotive gateway modules, enabling continuous on-board monitoring of CAN traffic without reliance on external computation or cloud connectivity. The modular design and high-level synthesis workflow further support reproducibility and scalability toward future embedded AI security applications.

Future research directions include extending experimental validation to diverse datasets and real vehicular environments, investigating adaptive and online intrusion detection mechanisms, and enhancing robustness against evolving and adversarial attack strategies. Additional work will also explore further hardware–software partitioning optimizations and deployment on next-generation heterogeneous computing platforms to improve system-level efficiency and scalability. Overall, the proposed co-designed FPGA-based IDS provides a practical and efficient pathway toward secure, real-time embedded AI solutions for connected and autonomous vehicles.

References

- [1] Özdemir, Ö., İsyapar, M.T., Karagöz, P., Schmidt, K.W., Demir, D., Karagöz, N.A.: A survey of anomaly detection in in-vehicle networks. *arXiv preprint arXiv:2409.07505* (2024)
- [2] Buscemi, A., Turcanu, I., Castignani, G., Panchenko, A., Engel, T., Shin, K.G.: A survey on controller area network reverse engineering. *IEEE Communications Surveys & Tutorials* **25**(3), 1445–1481 (2023)
- [3] Lo, W., Alqahtani, H., Thakur, K., Almadhor, A., Chander, S., Kumar, G.: A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic. *Vehicular Communications* **35**, 100471 (2022)
- [4] El-Rewini, Z., Sadatsharan, K., Selvaraj, D.F., Plathottam, S.J., Ranganathan, P.: Cybersecurity challenges in vehicular communications. *Vehicular Communications* **23**, 100214 (2020)
- [5] İleri, K., Rakib, A., Djahel, S.: Metacan: An optimized adaptive hybrid metaheuristic-based intrusion detection system for can bus security. *Vehicular Communications*, 100956 (2025)
- [6] Jablaoui, R., Liouane, N.: Network security based combined cnn-rnn models for iot intrusion detection system. *Peer-to-Peer Networking and Applications* **18**(3), 129 (2025)
- [7] Velasquez, J.D., Cadavid, L., Franco, C.J.: Emerging trends and strategic opportunities in tiny machine learning: A comprehensive thematic analysis.

- [8] Tzach, Y., Meir, Y., Gross, R.D., Tevet, O., Koresh, E., Kanter, I.: Advanced deep architecture pruning using single-filter performance. *Physical Review E* **111**(6), 065307 (2025)
- [9] Javed, A.R., Rehman, S., Khan, M.U., Alazab, M., Reddy, T.: Canintelliids: Detecting in-vehicle intrusion attacks on a controller area network using cnn and attention-based gru. *IEEE transactions on network science and engineering* **8**(2), 1456–1466 (2021)
- [10] Moulahi, T., Zidi, S., Alabdulatif, A., Atiquzzaman, M.: Comparative performance evaluation of intrusion detection based on machine learning in in-vehicle controller area network bus. *IEEE Access* **9**, 99595–99605 (2021)
- [11] Rai, R., Grover, J., Sharma, P., Pareek, A.: Securing the can bus using deep learning for intrusion detection in vehicles. *Scientific Reports* **15**(1), 13820 (2025)
- [12] Khandelwal, S., Shreejith, S.: A lightweight fpga-based ids-ecu architecture for automotive can. In: 2022 International Conference on Field-programmable Technology (ICFPT), pp. 1–9 (2022). IEEE
- [13] Rangsikunpum, A., Amiri, S., Ost, L.: An fpga-based intrusion detection system using binarised neural network for can bus systems. In: 2024 IEEE International Conference on Industrial Technology (ICIT), pp. 1–6 (2024). IEEE
- [14] Rangsikunpum, A., Amiri, S., Ost, L.: Ultra-lightweight and highly efficient pruned binarised neural networks for intrusion detection in in-vehicle networks. *Electronics* **14**(9), 1710 (2025)
- [15] Liu, Q., Amiri, S., Ost, L.: Low-cost intrusion detection system for can bus networks using risc-v with binarised neural networks. In: 2025 IEEE International Conference on Industrial Technology (ICIT), pp. 1–6 (2025). IEEE
- [16] Xu, H., Shi, X., Liu, H., Wang, Y., Lu, J., Zeng, H., Li, R., Wu, D.: Mulsam: Multi-dimensional attention with hardware acceleration for efficient intrusion detection on vehicular can bus. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025)
- [17] Al-Jarrah, O.Y., Maple, C., Dianati, M., Oxtoby, D., Mouzakitis, A.: Intrusion detection systems for intra-vehicle networks: A review. *Ieee Access* **7**, 21266–21289 (2019)
- [18] Boland, H.M., Burgett, M.I., Etienne, A.J., Stwalley III, R.M.: An overview of can-bus development, utilization, and future potential in serial network messaging for off-road mobile equipment. *Technology in Agriculture* (2021)

- [19] Kang, T.U., Song, H.M., Jeong, S., Kim, H.K.: Automated reverse engineering and attack for can using obd-ii. In: 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), pp. 1–7 (2018). IEEE
- [20] Verma, M.E., Bridges, R.A., Iannacone, M.D., Hollifield, S.C., Moriano, P., Hespeler, S.C., Kay, B., Combs, F.L.: A comprehensive guide to can ids data and introduction of the road dataset. PLoS one **19**(1), 0296879 (2024)
- [21] Sullivan, W., Steinkamp, D.: Characterization of throughput on the axi dma bus for burst data transfer over ethernet. Technical report, Fermi National Accelerator Laboratory (FNAL), Batavia, IL (United States) (2024)
- [22] Lee, H., Jeong, S.H., Kim, H.K.: Otds: A novel intrusion detection system for in-vehicle network by using remote frame. In: 2017 15th Annual Conference on Privacy, Security and Trust (PST), pp. 57–5709 (2017). IEEE
- [23] Patil, S., Jain, S., Kelkar, S.: Enhancing automotive security: A comparative study of machine learning model for anomaly detection in can bus system. In: 2025 IEEE 6th India Council International Subsections Conference (INDISCON), pp. 1–6 (2025). <https://doi.org/10.1109/INDISCON66021.2025.11254357>
- [24] Maklin, C.: Synthetic minority over-sampling technique (smote). Medium. url: <https://medium.com/@corymaklin/synthetic-minorityover-sampling-technique-smote-7d419696b88c> (2022)
- [25] Ferro, M.V., Mosquera, Y.D., Pena, F.J.R., Bilbao, V.M.D.: Early stopping by correlating online indicators in neural networks. Neural Networks **159**, 109–124 (2023)
- [26] Design, M.: Vivado design suite user guide (2012)
- [27] zc702. <https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/ek-z7-zc702-g.html>/. Accessed: 2026-02-6
- [28] Fahim, F., Hawks, B., Herwig, C., Hirschauer, J., Jindariani, S., Tran, N., Carloni, L.P., Di Guglielmo, G., Harris, P., Krupa, J., et al.: hls4ml: An open-source code-sign workflow to empower scientific low-power machine learning devices. arXiv preprint arXiv:2103.05579 (2021)
- [29] Wu, H., Judd, P., Zhang, X., Isaev, M., Micikevicius, P.: Integer quantization for deep learning inference: Principles and empirical evaluation. arXiv preprint arXiv:2004.09602 (2020)
- [30] Vakili, M., Ghamsari, M., Rezaei, M.: Performance analysis and comparison of machine and deep learning algorithms for iot data classification. arXiv preprint arXiv:2001.09636 (2020)