chips

MDPI

Article

Optimised Extension of an Ultra-Low-Power RISC-V Processor to Support Lightweight Neural Network Models

Qiankun Liu and Sam Amiri *

Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough LE11 3TU, UK; q.liu2@lboro.ac.uk
* Correspondence: s.amiri@lboro.ac.uk

Abstract: With the increasing demand for efficient deep learning models in resource-constrained environments, Binary Neural Networks (BNNs) have emerged as a promising solution due to their ability to significantly reduce computational complexity while maintaining accuracy. Their integration into embedded and edge computing systems is essential for enabling real-time AI applications in areas such as autonomous systems, industrial automation, and intelligent security. Deploying BNN on FPGA using RISC-V, rather than directly deploying the model on FPGA, sacrifices detection speed but, in general, reduces power consumption and on-chip resource usage. The AI-extended RISC-V core is capable of handling tasks beyond BNN inference, providing greater flexibility. This work utilises the lightweight Zero-Riscy core to deploy a BNN on FPGA. Three custom instructions are proposed for convolution, pooling, and fully connected layers, integrating XNOR, POPCOUNT, and threshold operations. This reduces the number of instructions required per task, thereby decreasing the frequency of interactions between Zero-Riscy and the instruction memory. The proposed solution is evaluated on two case studies: MNIST dataset classification and an intrusion detection system (IDS) for in-vehicle networks. The results show that for MNIST inference, the hardware resources required are only 9% of those used by state-of-the-art solutions, though with a slight reduction in speed. For IDS-based inference, power consumption is reduced to just 13% of the original, while resource usage is only 20% of the original. Although some speed is sacrificed, the system still meets real-time monitoring requirements.

Keywords: RISC-V; machine learning; binarised neural network; FPGA

check for updates

Academic Editors: Andrea Ballo and Orazio Aiello

Received: 28 February 2025
Revised: 28 March 2025
Accepted: 1 April 2025
Published: 3 April 2025

Citation: Liu, Q.; Amiri, S. Optimised Extension of an Ultra-Low-Power RISC-V Processor to Support Lightweight Neural Network Models. *Chips* **2025**, *4*, 13. https://doi.org/10.3390/chips4020013

Copyright: © 2025 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) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Deep learning has been widely adopted across various domains, including embedded systems, industrial automation, and edge computing applications [1]. However, conventional deep neural networks (DNNs) require significant computational power, memory bandwidth, and energy resources, making their deployment challenging in resource-constrained environments [2]. Edge devices and embedded systems often have strict limitations on power consumption and available hardware resources, necessitating more efficient solutions for neural network inference [3].

To address these constraints, BNNs have been explored as a promising alternative. By binarising both weights and activations to 1-bit precision, BNNs eliminate the need for floating-point multiplications and instead rely on efficient XNOR and POPCOUNT operations [4]. These operations significantly reduce computational complexity, memory access, and power consumption, making BNNs well suited for resource-limited hardware

platforms such as FPGAs and embedded processors [5]. While BNNs can achieve competitive accuracy in various applications, their deployment requires a suitable hardware architecture that fully utilises their computational efficiency [6].

Among the available hardware architectures, RISC-V has emerged as a flexible and power-efficient solution for deploying deep learning models in embedded environments [7]. Unlike proprietary ISAs, the open-source RISC-V ISA allows for custom instruction extensions, enabling designers to optimise hardware for specific workloads [8]. This capability is particularly beneficial for BNN inference, where dedicated instructions can streamline key operations such as binary convolution, pooling, and activation functions, reducing instruction overhead, memory access latency, and execution cycles [9]. The flexibility of RISC-V has been demonstrated in various low-power AI accelerators, where customised processing pipelines have significantly improved power efficiency compared with other general-purpose processors [10]. For example, Zero-Riscy, a lightweight 32-bit RISC-V implementation featuring architectural modifications, achieves lower power consumption and enhanced instruction execution efficiency, enabling a more optimised deployment of BNNs [11].

Integrating BNN inference within a RISC-V core provides additional advantages beyond domain-specific hardware acceleration. Unlike traditional FPGA-based deep learning accelerators that rely solely on dedicated processing units, an FPGA-deployed RISC-V processor can execute BNN inference while simultaneously handling other instructions, eliminating the need for an additional processing unit [12]. This hybrid capability has been shown to enhance real-time AI applications, particularly in scenarios requiring dynamic task execution alongside deep learning inference [13].

Deploying a RISC-V processor on an FPGA for BNN inference offers a practical and effective approach to achieving cost- and energy-efficient deep learning acceleration [14]. Moreover, the adaptability of RISC-V-based architectures, combined with FPGA-based implementations, enables continuous refinement to ensure compatibility with evolving AI workloads while maintaining power and resource efficiency [15].

This work presents a low-cost, energy-efficient RISC-V processor based on Zero-Riscy, specifically designed to accelerate BNN inference. This design is implemented and validated on a cost-effective FPGA platform, prioritising resource and energy efficiency, while meeting the real-time performance requirements of practical applications rather than emphasising maximum inference speed. Although further performance enhancements, such as increasing pipeline stages or utilising cache memory, could improve inference speed, a trade-off was applied to prioritise low resource and energy consumption. This approach ensures suitability for BNN applications while avoiding unnecessary speed-ups. Three custom instructions are introduced for convolution, pooling, and fully connected layers, optimising execution efficiency without adding substantial resource overhead relative to the baseline processor.

Key contributions of the paper include the following:

- A RISC-V processor optimised for BNN inference that embeds custom instructions to accelerate key layers while minimising hardware cost.
- Extensive validation through two case studies, one on MNIST dataset classification and another on an automotive in-vehicle network intrusion detection system, demonstrating efficiency across diverse workloads.
- A comprehensive comparison and analysis of the implementations, which reveal a power-efficient, resource-aware design suited for edge AI applications. The design significantly reduces on-chip resource usage and power consumption compared with existing FPGA-based solutions, all while maintaining real-time performance.

The rest of this paper is structured as follows. Section 2 introduces the background and key techniques underlying the processor architecture. Section 3 details the hardware implementation. Section 4 presents experimental results on the MNIST and CAN bus datasets, comparing performance with alternative designs. Finally, Section 5 concludes with key findings and outlines future research directions.

2. Background and Related Works

This section provides the fundamental knowledge and technical background for constructing the proposed RISC-V-based low-cost hardware acceleration solution for BNN inference. It covers the principles of BNNs, highlights the advantages of implementing low-cost neural network inference on RISC-V, and introduces Zero-Riscy, a minimalistic RISC-V processor.

2.1. Binary Neural Networks

DNNs have achieved remarkable success in various applications, but their high computational cost and memory requirements limit their deployment on resource-constrained devices. BNNs mitigate this challenge by reducing weights and activations to just two states, +1 and -1, replacing computationally expensive floating-point operations with efficient bitwise logic. This significantly reduces power consumption and computational and storage requirements [16]. Given a weight matrix W and an input activation X, BNNs approximate the standard matrix multiplication as:

$$Y = sign(X) \cdot sign(W) \tag{1}$$

where $sign(\cdot)$ binarises the inputs to ± 1, allowing inference to be performed using *XNOR* and *POPCOUNT* operations instead of traditional multiplications [17]:

$$X \cdot W = POPCOUNT(XNOR(X, W)) \tag{2}$$

In addition to significantly reducing inference computational complexity through highly parallelisable bitwise operations, BNNs achieve up to 32× memory compression compared with full-precision DNNs [16,18]. However, the aggressive quantisation introduces accuracy degradation due to the loss of representational capacity. To mitigate this, techniques like real-valued scaling factors, gradient approximation via Straight-Through Estimator (STE), and hybrid quantisation have been explored to balance efficiency and accuracy [19].

Due to their efficiency benefits, BNNs are well suited for edge computing and embedded AI applications, especially when deployed on an AI-enhanced processor. Open-source RISC-V architectures have gained prominence in domain-specific acceleration due to their flexibility in hardware customisation. Recent advancements include custom instructions for various domain-specific operations, reducing instruction overhead and enhancing execution efficiency on RISC-V-based FPGA accelerators [5]. Integrating these optimisations enables real-time inference while maintaining minimal power consumption.

As BNN research advances, improvements in training algorithms, hardware-aware quantisation, and specialised instruction sets are bridging the gap between power-efficient inference and competitive accuracy. The convergence of BNNs and RISC-V architectures is driving the development of ultra-low-power, high-performance AI solutions for real-world applications, particularly in security-critical and autonomous embedded systems [5].

2.2. RISC-V Architecture

RISC-V has rapidly emerged as a competitive alternative to proprietary instruction set architectures (ISAs), offering a fully open, modular, and scalable framework suited for a wide range of computing applications. Unlike x86 and ARM, which are tightly controlled by industry giants, RISC-V's open-source nature allows for extensive hardware customisation, making it particularly well suited for AI acceleration and deep-learning inference on embedded and edge devices [20]. Its simplicity and efficiency, combined with rapidly growing industry support, have driven its adoption in low-power AI computing.

The architecture's reduced instruction set computing (RISC) design allows for streamlined instruction execution, leading to higher efficiency in processing vectorised and parallel workloads. AI inference, particularly neural network acceleration, benefits from RISC-V's extensible vector processing capabilities. The RISC-V Vector Extension (RVV) enables hardware-accelerated tensor operations by processing multiple data points in parallel, significantly boosting the throughput of convolutional and fully connected layers in deep learning models [20]. Compared with traditional scalar architectures, RISC-V vector units provide higher computational density with lower power consumption, making them ideal for low-energy AI accelerators and embedded neural networks.

With the rise of quantised neural networks (QNNs) and energy-efficient AI models, RISC-V has proven particularly effective in handling low-precision arithmetic. The support for sub-byte data types, including int4, int8, and float16, allows for efficient deep-learning inference with reduced memory bandwidth and computation overhead [5]. This capability is crucial in resource-constrained environments where power efficiency is as important as computational performance. Additionally, hardware implementations of RISC-V-based AI accelerators, such as those integrating systolic array architectures or in-memory computing techniques, have demonstrated significant gains in throughput and efficiency [14].

RISC-V's capability to integrate vector processing, mixed-precision support, and modular extensions makes it a strong candidate for efficient AI acceleration, particularly in edge computing and embedded systems where power and cost constraints are critical [14].

2.3. Zero-Riscy: A Lightweight and Ultra-Low Power RISC-V Processor

Zero-Riscy [21] is a 32-bit RISC-V processor core developed as part of the PULP (Parallel Ultra-Low-Power) platform at ETH Zurich. The design goal of Zero-Riscy is to provide an efficient computing solution for resource-constrained embedded devices, making it particularly well suited for applications in IoT, sensor nodes, and wearable devices [22]. Zero-Riscy employs a simplified two-stage pipeline architecture consisting of the Instruction Fetch (IF) and Execute (EX) stages. This design prioritises reducing pipeline stalls and improving energy efficiency while maintaining sufficient computational capacity to support typical embedded applications. The structure of Zero-Riscy, including its two pipeline stages, is illustrated in Figure 1.

Additionally, Zero-Riscy supports hardware interrupts, low-power modes, and clock gating, enabling efficient operation across various application scenarios. For instance, in a sensor network, Zero-Riscy can wake up and execute data processing tasks upon detecting a sensor event, then enter a low-power mode until the next event occurs.

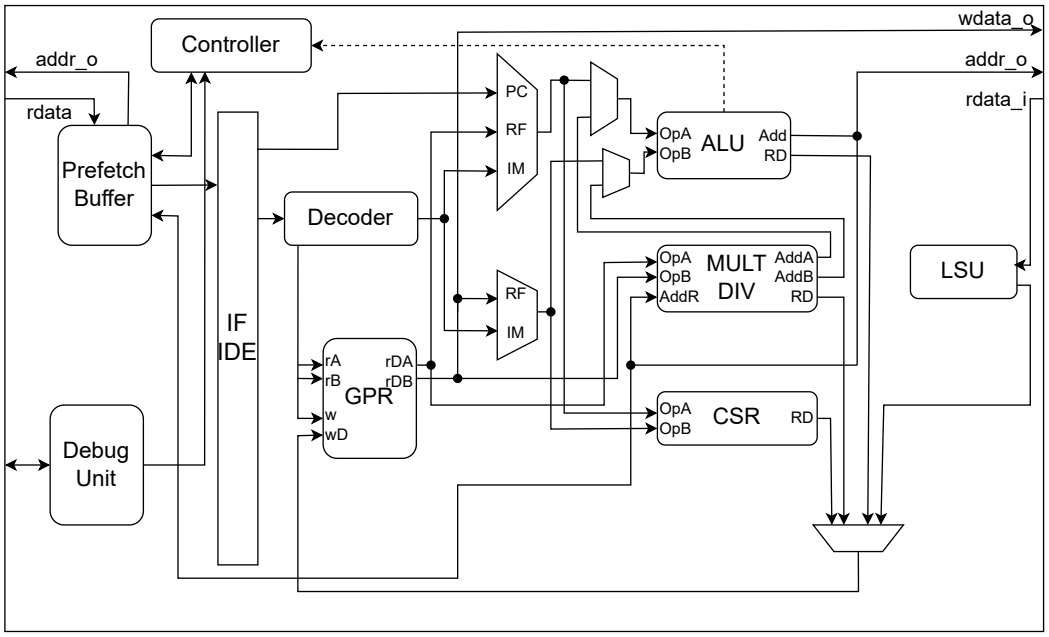


Figure 1. The structure of Zero-Riscy [23].

3. Proposed Ultra-Low-Power RISC-V Processor with BNN Support

This section introduces our novel ultra-low-power RISC-V processor, optimised for lightweight neural network inference. The hardware design is first introduced, followed by the integration of three custom instructions for BNN acceleration.

3.1. Zero-Riscy with the Proposed BNN Unit

Using an RISC-V processor to implement the inference phase of binarised neural networks is a more flexible approach compared with directly deploying BNNs on FPGAs with frameworks such as Xilinx FINN. This allows RISC-V to be used for both standard applications and neural network tasks without dedicating any additional area specifically for neural network operations. RISC-V is an open-source ISA that can be customised for specific applications. Developers can add custom instructions to optimise specific neural network operations, providing valuable flexibility during the development and debugging phases. This allows for easy code modification and algorithm updates.

This study aims to implement the fully binarised neural network inference phase using Zero-Riscy. Given the large computational workload and high data reuse, three custom instructions are introduced for acceleration. These instructions are specifically designed for the convolutional layer, max pooling layer, and fully connected layer. The functionality supporting these instructions is encapsulated within the BNN unit. The structure of Zero-Riscy, incorporating the BNN unit, is shown in Figure 2. Since the custom instructions require multiple lines of input data, three dedicated output ports were added to the register file to efficiently transfer all necessary data to the BNN unit within a single instruction. These output ports exclusively serve the BNN unit, ensuring they do not interfere with other functions within Zero-Riscy.

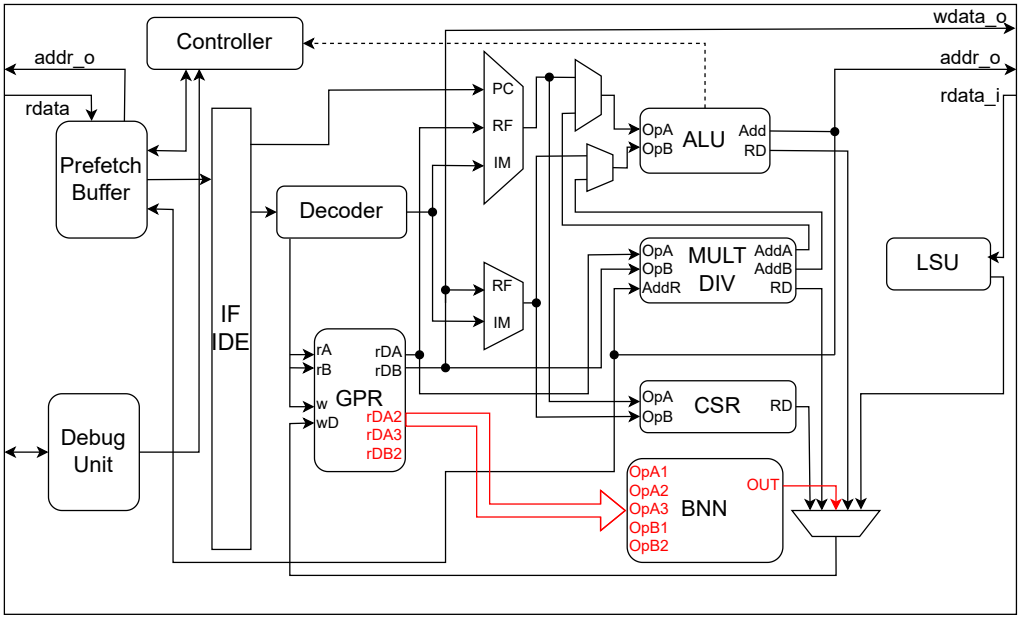


Figure 2. The structure of Zero-Riscy with the proposed BNN unit.

3.2. Proposed Custom Instructions

The proposed custom instructions are tailored for the convolutional, pooling, and fully connected layers in the BNN model. The convolutional layer and fully connected layer combine sliding window, XNOR, POPCOUNT, thresholding, and accumulation into a single instruction, though they differ in their implementation. Details of these approaches will be discussed in the following paragraphs. The custom instruction for max pooling incorporates a sliding window and OR operations to efficiently compute the maximum value matrix. The format of these three custom instructions is presented in Table 1. This design maximises data reuse, enhances computational parallelism, and improves overall efficiency.

Table 1. Encoding of proposed custom instructions for BNN acceleration in Zero-Riscy processor.

	31:25	24:20	19:15	14:12	11:7	6:0
CONV	funct7 (0000000)	rs2	rs1	funct3 (011)	rd	op (1111011)
Max Pooling	funct7 (0000000)	rs2	rs1	funct3 (100)	rd	op (1111011)
FC	funct7 (0000000)	rs2	rs1	funct3 (101)	rd	op (1111011)

The custom convolutional instruction integrates sliding window, XNOR, POPCOUNT, accumulation, and thresholding operations. This integration increases the utilisation of a single instruction, reduces the total number of instructions needed, and enhances the reuse of weights. The operation of the proposed binarised convolutional instruction is illustrated in Figure 3. In this example, the filter window size is 3×3, meaning each input feature map (IFM) corresponds to a 9-bit weight and a threshold in a node. The convolutional operation involves extracting each 9-bit data segment from the IFM using the filter window and performing an XNOR operation with the corresponding weight. The result is then processed using a POPCOUNT to count the number of ones, which is stored in an intermediate matrix. Once all IFMs corresponding to a node have been

processed, the POPCOUNT results stored in the intermediate matrix are summed. The final value is compared with the threshold: if it is greater than or equal to the threshold, the corresponding position in the output feature map (OFM) is set to 1; otherwise, it is set to 0.

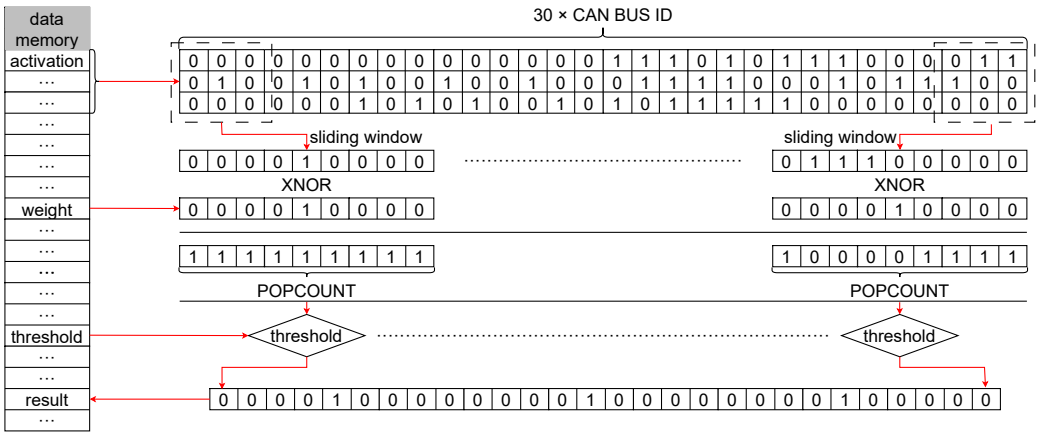


Figure 3. High-level view of the operations in the proposed binarised convolutional layer.

The input to the max pooling layer comes from the previous convolution layer. For instance, with a max pooling window of size 2 × 2 and no padding, the operation reduces the input dimensions by half. The primary function of this layer is to select the maximum value within the window. Given that the elements in the matrix are binary (0 or 1), the output will be 1 if at least one element in the window is 1; otherwise, it remains 0. As a result, binary max pooling can be simplified to an OR operation across the four elements within the window. The window moves with a stride of 2, ensuring non-overlapping pooling regions. The custom max pooling instruction takes two consecutive rows of the input matrix and generates a single output row with dimensions reduced by half, storing the results in the designated memory location.

The main operation of the fully connected layer involves performing an XNOR operation between the input and the weights, followed by a POPCOUNT on the results. Since the fully connected layer is the final stage of the model, its input comes from the output of the preceding max pooling layer. In our example, this output is a 5 × 5 matrix. The 5 × 5 matrix is stored in five consecutive bytes, with the lower 5 bits of each byte holding a row of matrix data, while the upper 3 bits are padded with 0s. Similarly, the weights are stored in the lower 5 bits of consecutive bytes, but the upper 3 bits are set to 1s. This padding ensures correct computation because if the upper bits of both the input and the weights were 0, their XNOR result would be 1, incorrectly increasing the POPCOUNT value by 3. The operation of the fully connected layer is illustrated in Figure 4.

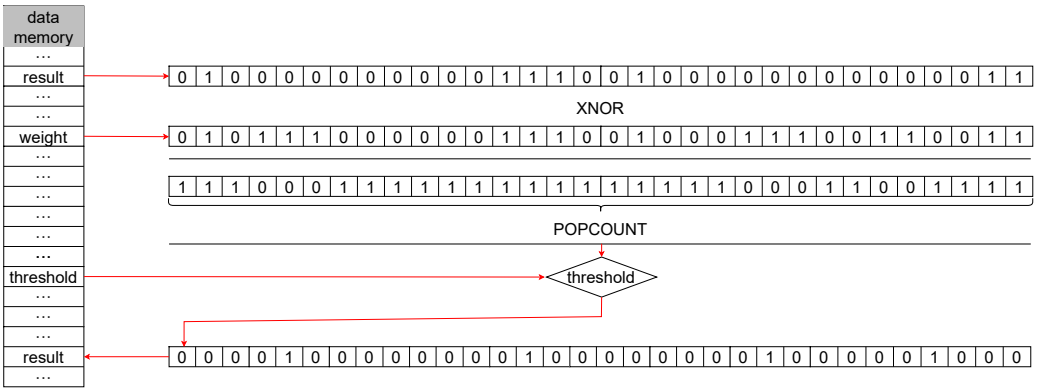


Figure 4. High-level view of the operations in the proposed binarised fully connected layer.

4. Experimental Results

In this study, we conducted two experiments. The first experiment involved handwritten Arabic numeral classification using a BNN model trained on the grayscale MNIST dataset. The BNN model was trained and evaluated using Larq. The second experiment focused on an automotive intrusion detection system (IDS) model trained on the Car-Hacking dataset. The Car-Hacking data used in this evaluation are sourced from [24], while the IDS model is based on [25]. This experiment specifically assessed the deployment of the model on the proposed low-power RISC-V processor to evaluate its performance and efficiency.

The target device in this study is the ZedBoard, a development platform equipped with a Xilinx Zynq-7000 SoC. This SoC combines a dual-core ARM Cortex-A9 processor with programmable logic, offering a versatile solution for FPGA-based system development. The ZedBoard provides a balance between performance, flexibility, and cost-effectiveness, making it well suited for prototyping embedded systems and hardware-accelerated designs.

4.1. BNN Inference on the MNIST Dataset Using the Proposed RISC-V Processor

The MNIST dataset is a widely used benchmark for handwritten digit recognition, containing 70,000 grayscale images (60,000 for training and 10,000 for testing). Each image is a 28 × 28-pixel representation of a handwritten digit (0–9) [26]. Originally introduced by LeCun et al., MNIST provides a structured and standardised dataset that has become a fundamental testbed for evaluating neural network architectures, feature extraction techniques, and hardware acceleration strategies [27]. While modern deep learning models have significantly surpassed human-level performance on MNIST, the dataset remains a key reference for network optimisation, quantised inference, and low-power AI applications [28].

This case study utilises a BNN model trained with Larq, where the MNIST dataset is binarised at an early stage of training. The model consists of multiple convolutional layers, pooling layers, batch normalisation layers, and fully connected layers, totalling 93.6 k parameters and achieving 97.62% accuracy. The architecture is illustrated in Figure 5. The parameters of the BNN model trained with Larq are transferred to the Xilinx Vitis environment, synthesised, and implemented by Xilinx Vitis HLS 2022. When implemented on the target device Zedboard, the system operates at 66 MHz and a temperature of 28.4 °C, with a dynamic power consumption of 0.187 W and a static power consumption of 0.107 W, resulting in a total power consumption of only 0.294 W, as shown in Figure 6. The comparison between the number of basic instructions required to complete one MNIST classification task before using the proposed instructions and the number of instructions required after optimisation is shown in Table 2. The total number of instructions has been reduced by 85.1%.

Table 2. Instruction cycles for MNIST dataset classification.

Normal Instruction Cycles	Proposed Instruction Cycles
135,551	20,276 (↓ 85.1%)

Figure 7 presents a comparison of FPGA-based MNIST accelerators in terms of inference time, power consumption, and resource utilisation. Prior works [29–31] have focused on optimising throughput using parallel pipelines, floating-point computation, and memory-efficient architectures, achieving low latency at the expense of high DSP and LUT utilisation.

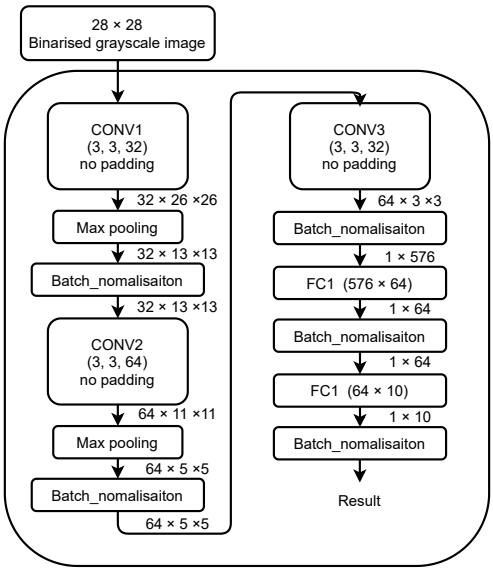


Figure 5. BNN model architecture trained with Larq on the MNIST dataset.

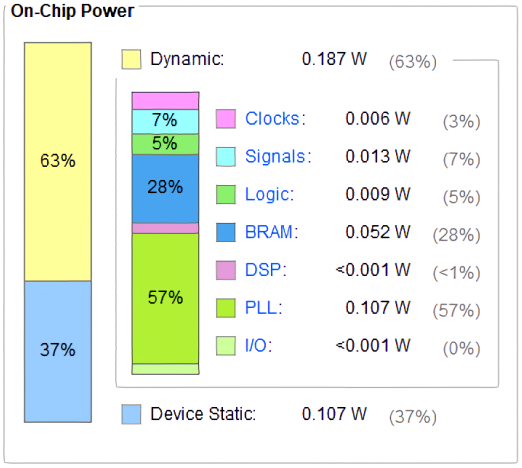


Figure 6. Power consumption report from Xilinx Vivado for the MNIST classification experiment.

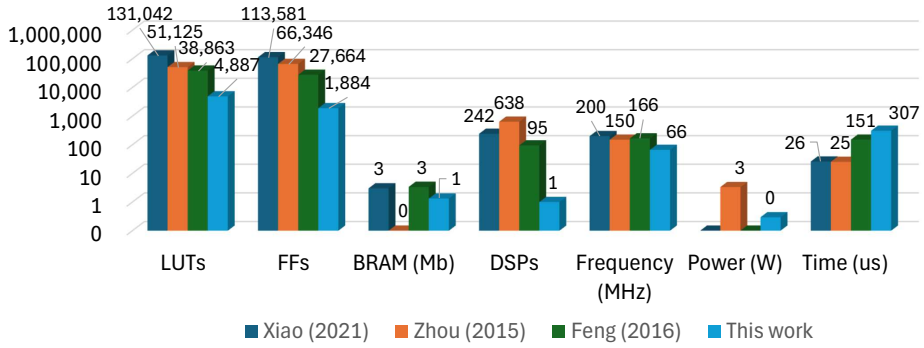


Figure 7. Comparison of timing, power, and resource utilisation between Xiao (2021) [29], Zhou (2015) [31], Feng (2016) [30] and this work for MNIST classification.

The accelerator in [29] operates at 200 MHz, using a multi-channel pipeline to achieve a 25.9 µs inference time but consuming 242 DSPs and over 130 k LUTs, making it resource-intensive. In [30], inference time is further reduced to 25.4 µs by incorporating floating-point operations with line buffering; however, this approach demands 638 DSPs, limiting its feasibility for edge computing. Meanwhile, ref. [31] prioritises memory efficiency by

employing fixed-point computation and utilising 3.38 Mb of BRAM, reducing logic usage but increasing inference time to 151 µs.

Our approach adopts a different strategy, prioritising hardware efficiency over raw throughput. Directly comparing the inference time of processors running at different frequencies can be misleading. To address this, we employ a normalised time method, defined by the following formula:

$$\text{Normalised time} = \frac{\text{Actual time}}{\text{Clock cycle}} \tag{3}$$

where "Actual time" refers to the duration required to execute a task, and "Clock cycle" is calculated as 1/Processor frequency.

The normalized times calculated for [29–31] and this work are 5180, 3810, 25,066, and 25,075.2, respectively. Compared with other FPGA-based implementations of MNIST neural network inference, our proposed method significantly reduces hardware overhead, consuming only 9% of the power required by [30] while also eliminating the need for a large number of DSP units. However, this reduced hardware footprint results in an increased inference time, with a normalised execution time approximately eight times longer than [30], but nearly equivalent to [31], which also does not rely on DSP units. Despite this tradeoff, our method achieves an accuracy of 97.62%, making it a scalable and power-efficient solution for embedded AI applications. Despite its lower performance, our ultra-lightweight and power-efficient approach still meets the real-time requirements of many edge-AI applications, providing a resource-efficient solution for real-world problems.

4.2. Implementing a Low-Cost IDS for In-Vehicle Networks Using BNN

The increasing number of sensors used in smart vehicles has significantly raised concerns about in-vehicle network security. The machine learning (ML)-based Intrusion Detection System (IDS) designed for the Controller Area Network (CAN) protocol has gained significant popularity due to its excellent performance in detecting attacks without introducing additional overhead [32].

4.2.1. CAN Bus Message

The CAN is a serial communication protocol developed by Bosch in the 1980s for automotive and industrial automation applications. Due to its low cost, it has become a widely adopted standard [33]. However, despite its emphasis on reliability during design, the lack of encryption or any authentication mechanisms in the CAN protocol has led to significant security vulnerabilities [34]. For instance, security researchers have demonstrated remote control of a Jeep Cherokee, manipulating critical systems such as the steering and braking system [35].

To address these security concerns, various defensive measures have been proposed, including ECU authentication [36] and hardware-enforced isolation [37]. IDSs have received widespread attention as an effective approach, as they do not introduce additional overhead to the CAN bus.

Unlike traditional point-to-point communication methods, the CAN bus is a multi-master system that allows multiple nodes to exchange data on the same bus. In a CAN bus system, data are transmitted in the form of frames, each consisting of several parts: Start of Frame (SOF), CAN ID, control bits, Remote Transmission Request (RTR), data field, Cyclic Redundancy Check (CRC), acknowledgement (ACK) field, and End of Frame (EOF). The CAN ID determines the priority of the data and their transmission target, while the data field contains the actual information being transmitted. There are two frame types,

standard and extended, differentiated by identifier length: 11 bits for standard frames and 29 bits for extended frames [12]. The structure of a CAN message is illustrated in Figure 8.

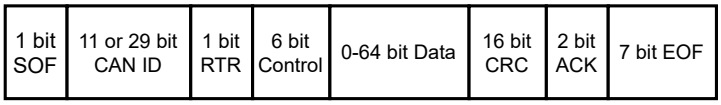


Figure 8. Framework of a CAN bus message [12].

The CAN bus uses priority-based arbitration to resolve access conflicts, ensuring the messages with the lowest identifier transmit first. This mechanism guarantees the timely delivery of critical data and optimises bus utilisation. It also features robust error detection and correction, including bit monitoring, bit stuffing, format checking, and cyclic redundancy check (CRC) validation, to ensure reliable data transmission. Additionally, faulty nodes are automatically isolated to prevent network disruption.

Reference [38] outlines several common attack methods. A Denial of Service (DoS) attack floods the bus with high-priority intrusive messages, delaying or blocking legitimate communication. Fuzzing attacks involve a malicious node randomly generating and sending messages with different CAN IDs to disrupt normal operations. Spoofing attacks specifically target certain CAN IDs, such as forging RPM or gear position data, potentially misleading drivers and compromising driving safety.

4.2.2. Model Structure and Performance

A promising FPGA-based IDS using BNN implements a two-stage Coarse-to-Fine (C2F) architecture [38], where a lightweight model initially detects potential intrusions, triggering a fine-grained model only when necessary. The structure of the C2F model is shown in Figure 9. The model's architecture and parameters are identical to those in [39]. Therefore, the model's accuracy, precision, and F1 score in this study remain consistent with the original work.

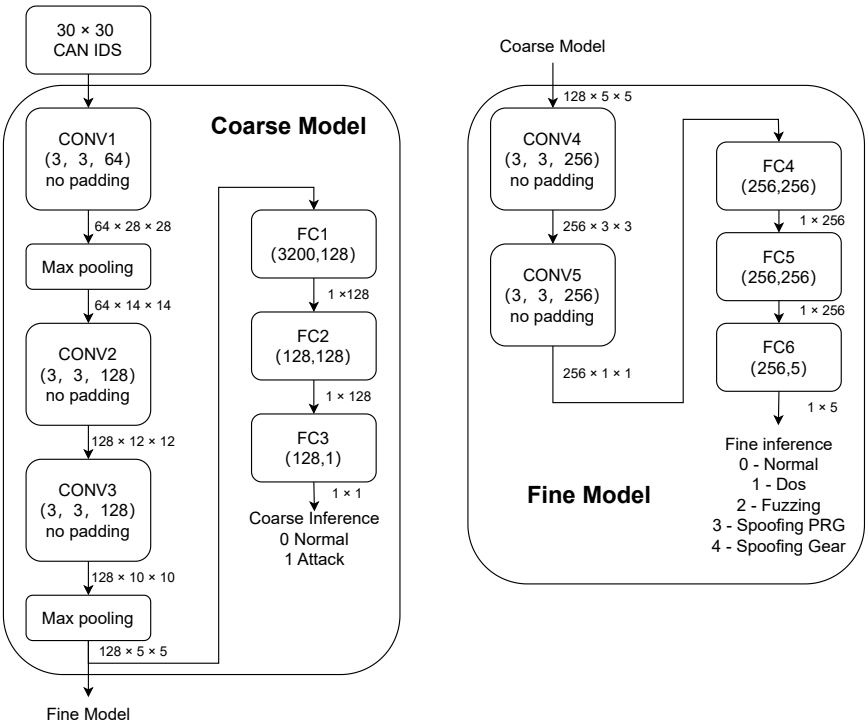


Figure 9. The Coarse and Fine models used in the C2F framework [39].

Table 3 presents a performance comparison of the Coarse model, a binary classification model within the C2F framework. The CANTransfer model achieves the lowest accuracy due to its use of a newly developed dataset tailored for novel attacks. In contrast, the other models achieve accuracy, precision, recall, and F1 scores exceeding 99%, with variations of less than 1%.

Table 3. Detection performance comparison of the Coarse model with state-of-the-art approaches.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
ACGAN [40]	-	99.23	99.24	99.23
DCNN [24]	99.94	99.98	99.87	99.92
CAAE [41]	99.90	99.97	99.72	99.84
CANTransfer [42]	99.37	94.84	95.57	95.25
MTH-IDS [43]	99.99	99.99	99.99	99.99
QMLP [12]	99.96	99.91	99.92	99.91
Coarse [39]	99.83	99.88	99.64	99.76

4.2.3. Resource Utilisation and Power Consumption

The inference time for the Coarse model on Zero-Riscy was measured at 13.37 ms using Vivado's behavioural simulation. Since the model requires assembling CAN IDs from 30 consecutive CAN messages to form a 2D array as input, real-time detection requires completing inference within the delay introduced by these 30 CAN messages. Based on the CAN message period statistics in the CH dataset [25], summarised in Table 4, the input delay for the normal dataset is calculated as $30 \times 0.512 = 15.36$ ms. Since this delay is greater than the inference time achieved with Zero-Riscy, the study meets the requirements for real-time detection. However, the model can only determine which CAN ID data block contains an attack, without pinpointing the specific CAN ID within the block that is under attack.

Table 4. CAN message transmission periods for each sub-dataset in the CH dataset.

Dataset	Attack	Mean (ms)	Standard Deviation (ms)
CH [25]	Normal	0.512	0.712
	DoS	0.767	1.446
	Fuzzy	0.768	1.465
	Spoofing RPM	0.547	0.896
	Spoofing gear	0.527	0.889

In addition to the Zero-Riscy processor, the instruction and data memory required for model implementation are realised using BRAM IP blocks in Vivado. The resource utilisation and power consumption of this study, compared with the QMLP and original C2F implementations, are shown in Figure 10. The proposed approach significantly reduces both resource usage and power consumption compared with these methods. Specifically, QMLP consumes 11 times the LUTs used in this study, while C2F consumes 6 times as many. The flip-flop (FF) usage in this study is less than 5% of that in both QMLP and C2F. DSP resources are not required for model inference. The single DSP utilised in this study is due to the multiplication module inherently present in Zero-Riscy. Since the model and computational methods used in this study are identical to those in C2F, the intermediate values generated during computation remain the same. However, the BRAM usage in this study is only one-third of that in C2F. This reduction is due to the parallelism employed in C2F [39], where multiple intermediate results from different layers are stored in the data

memory simultaneously. Figure 11 illustrates the implementation of this study on the Zynq SoC of the ZedBoard, with the utilised resources highlighted in blue.

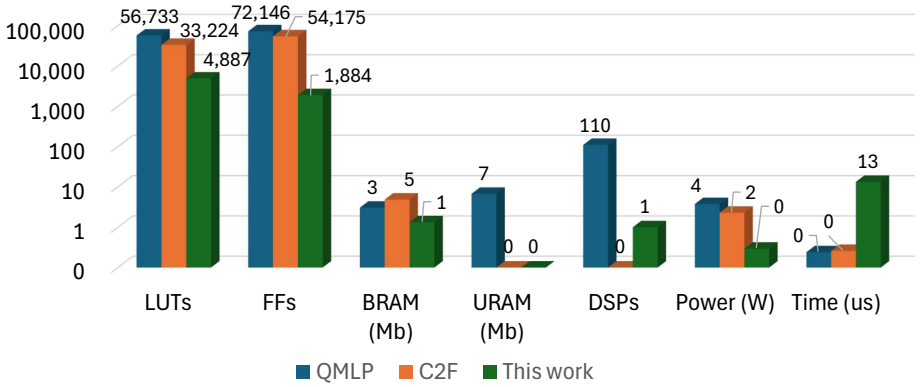


Figure 10. Comparison of hardware resource utilisation and power consumption among the implementations in this study, QMLP, and original C2F.

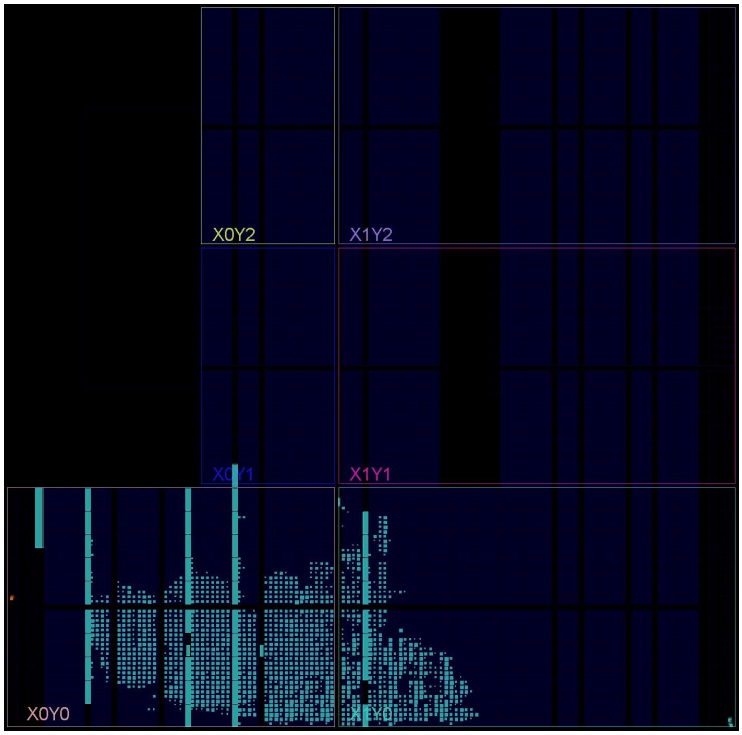


Figure 11. Implementation layout of this study on Zedboard's Zynq SoC.

By utilising BNNs in this study, multiplications are replaced with XNOR and POPCOUNT operations, and the computation order is optimised. As a result, the number of instructions required to complete an IDS detection task is also significantly reduced. The comparison between the number of basic instructions required to complete one IDS detection task before using the proposed instructions and the number required after optimisation is shown in Table 5. The total number of instructions has been reduced by 89.8%.

Table 5. Instruction cycles for IDS detection task.

Normal Instruction Cycles	Proposed Instruction Cycles
864.6 k	88.3 k (↓ 89.8%)

As mentioned in the previous case study, the system operates at 65 MHz, with a dynamic power consumption of 0.187 W and a static power consumption of 0.107 W, resulting in a total power consumption of only 0.294 W, which is less than 13% of the power consumption in the original C2F implementation. The low power consumption and resource utilisation of the enhanced RISC-V processor make it feasible to deploy IDS systems across multiple critical ECUs. Figure 12 illustrates the interfaces and functionalities of a distributed IDS.

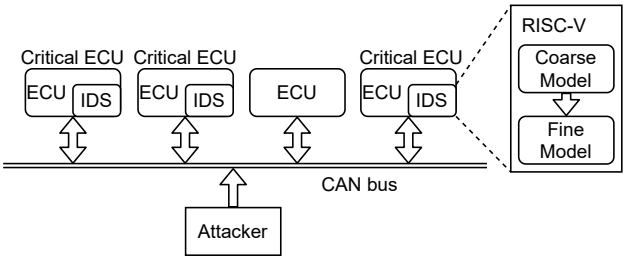


Figure 12. Diagram of an FPGA-based CAN bus IDS using RISC-V deployed across vehicle ECUs.

5. Conclusions

This paper proposes deploying an ultra-low-power RISC-V processor to support lightweight neural network models, utilising BNNs to simplify complex matrix operations into 1-bit XNOR and POPCOUNT operations, significantly reducing computational costs. The proposed BNN unit is integrated into the lightweight Zero-Riscy processor with a two-stage pipeline, ensuring efficient instruction execution.

To further optimise performance, three custom instructions are introduced, enabling 29 XNOR and POPCOUNT operations per data fetch without impacting Zero-Riscy's critical path or operating frequency. These enhancements lead to substantial savings in power and hardware resources, with experiments demonstrating that the proposed design achieves over 80% hardware resource reduction while maintaining 97.62% accuracy in FPGA-based MNIST BNN inference. The proposed method consumes only 13% of the power and 20% of the hardware resources compared with state-of-the-art IDS implementations. Despite an increase in inference time, the approach meets real-time detection requirements, making it ideal for automotive intrusion detection, where timely attack detection on the CAN bus is crucial for vehicle security. By integrating this IDS within ECUs, vehicles can efficiently identify and mitigate attacks with minimal computational overhead, providing a practical and energy-efficient solution for modern automotive security systems. Future work will explore multi-core collaboration and in-memory computing to further enhance real-time detection capabilities.

Author Contributions: Conceptualisation, Q.L. and S.A.; methodology, Q.L. and S.A.; software, Q.L.; validation, Q.L.; writing—original draft preparation, Q.L.; writing—review and editing, Q.L. and S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work is part of a PhD research project conducted at the Wolfson School, Loughborough University, UK.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data supporting the conclusions of this article will be made available by the author upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. EIE: Efficient inference engine on compressed deep neural network. *Acm Sigarch Comput. Archit. News* **2016**, *44*, 243–254. [CrossRef]
2. Sze, V.; Chen, Y.H.; Yang, T.J.; Emer, J.S. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* **2017**, *105*, 2295–2329.
3. Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In Proceedings of the 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco, CA, USA, 9–13 February 2014; pp. 10–14.
4. Umuroglu, Y.; Fraser, N.J.; Gambardella, G.; Blott, M.; Leong, P.; Jahre, M.; Vissers, K. Finn: A framework for fast, scalable binarized neural network inference. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017; pp. 65–74.
5. Garofalo, A.; Tagliavini, G.; Conti, F.; Benini, L.; Rossi, D. Xpulpnn: Enabling energy efficient and flexible inference of quantized neural networks on risc-v based iot end nodes. *IEEE Trans. Emerg. Top. Comput.* **2021**, *9*, 1489–1505.
6. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or −1. *arXiv* **2016**, arXiv:1602.02830.
7. Liu, Q.; Amiri, S.; Ost, L. Exploring RISC-V Based DNN Accelerators. In Proceedings of the 2024 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), London, UK, 29–31 July 2024; pp. 1–6.
8. Wang, C.; Fang, C.; Wu, X.; Wang, Z.; Lin, J. A Scalable RISC-V Vector Processor Enabling Efficient Multi-Precision DNN Inference. *arXiv* **2024**, arXiv:2401.16872.
9. Reggiani, E.; Pappalardo, A.; Doblas, M.; Moreto, M.; Olivieri, M.; Unsal, O.S.; Cristal, A. Mix-gemm: An efficient hw-sw architecture for mixed-precision quantized deep neural networks inference on edge devices. In Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Montreal, QC, Canada, 25 February–1 March 2023; pp. 1085–1098.
10. Andri, R.; Cavigelli, L.; Rossi, D.; Benini, L. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights. In Proceedings of the 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, USA, 11–13 July 2016; pp. 236–241.
11. Rossi, D.; Conti, F.; Marongiu, A.; Pullini, A.; Loi, I.; Gautschi, M.; Tagliavini, G.; Capotondi, A.; Flatresse, P.; Benini, L. PULP: A parallel ultra low power platform for next generation IoT applications. In Proceedings of the 2015 IEEE Hot Chips 27 Symposium (HCS), IEEE Computer Society, Cupertino, CA, USA, 22–25 August 2015; pp. 1–39.
12. Khandelwal, S.; Shreejith, S. A Lightweight FPGA-based IDS-ECU Architecture for Automotive CAN. In Proceedings of the 2022 International Conference on Field-Programmable Technology (ICFPT), Hong Kong, 5–9 December 2022; pp. 1–9.
13. Kwon, H.; Samajdar, A.; Krishna, T. Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects. *Acm Sigarch Notices* **2018**, *53*, 461–475.
14. Garofalo, A.; Ottavi, G.; Conti, F.; Karunaratne, G.; Boybat, I.; Benini, L.; Rossi, D. A heterogeneous in-memory computing cluster for flexible end-to-end inference of real-world deep neural networks. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2022**, *12*, 422–435.
15. Zhang, H.; Liu, J.; Bai, J.; Li, S.; Luo, L.; Wei, S.; Wu, J.; Kang, W. HD-CIM: Hybrid-device computing-in-memory structure based on MRAM and SRAM to reduce weight loading energy of neural networks. *IEEE Trans. Circuits Syst. Regul. Pap.* **2022**, *69*, 4465–4474.
16. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 525–542.
17. Bulat, A.; Martinez, B.; Tzimiropoulos, G. High-capacity expert binary networks. *arXiv* **2020**, arXiv:2010.03558.
18. Qin, H.; Gong, R.; Liu, X.; Shen, M.; Wei, Z.; Yu, F.; Song, J. Forward and backward information retention for accurate binary neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 2250–2259.
19. Martinez, B.; Yang, J.; Bulat, A.; Tzimiropoulos, G. Training binary neural networks with real-to-binary convolutions. *arXiv* **2020**, arXiv:2003.11535.
20. Waterman, A.; Lee, Y.; Patterson, D.A.; Asanovic, K. *The Risc-v Instruction Set Manual, Volume I: Base User-Level Isa*; Technical Report No. UCB/EECS-2016-118; EECS Department, University of California at Berkeley: Berkeley, CA, USA, 2011; Volume 116, pp. 1–32.
21. Davide Schiavone, P.; Conti, F.; Rossi, D.; Gautschi, M.; Pullini, A.; Flamand, E.; Benini, L. Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications. In Proceedings of the 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Thessaloniki, Greece, 25–27 September 2017; pp. 1–8.
22. Pullini, A.; Rossi, D.; Loi, I.; Tagliavini, G.; Benini, L. Mr. Wolf: An energy-precision scalable parallel ultra low power SoC for IoT edge processing. *IEEE J.-Solid-State Circuits* **2019**, *54*, 1970–1981.

23. Elsadek, I.; Tawfik, E.Y. RISC-V resource-constrained cores: A survey and energy comparison. In Proceedings of the 2021 19th IEEE International New Circuits and Systems Conference (NEWCAS), Toulon, France, 13–16 June 2021; pp. 1–5.
24. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198.
25. Rangsikunpum, A.; Amiri, S.; Ost, L. A Reconfigurable Coarse-to-Fine Approach for the Execution of CNN Inference Models in Low-Power Edge Devices. *Iet Comput. Digit. Tech.* **2024**, *2024*, 6214436.
26. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324.
27. Xu, L.; Krzyzak, A.; Suen, C.Y. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. Syst. Man. Cybern.* **2002**, *22*, 418–435.
28. Cohen, G.; Afshar, S.; Tapson, J.; Van Schaik, A. EMNIST: Extending MNIST to handwritten letters. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AL, USA, 14–19 May 2017; pp. 2921–2926.
29. Xiao, H.; Li, K.; Zhu, M. FPGA-based scalable and highly concurrent convolutional neural network acceleration. In Proceedings of the 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), Shenyang, China, 22–24 January 2021; pp. 367–370.
30. Feng, G.; Hu, Z.; Chen, S.; Wu, F. Energy-efficient and high-throughput FPGA-based accelerator for Convolutional Neural Networks. In Proceedings of the 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Hangzhou, China, 25–28 October 2016; pp. 624–626.
31. Zhou, Y.; Jiang, J. An FPGA-based accelerator implementation for deep convolutional neural networks. In Proceedings of the 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), Harbin, China, 19–20 December 2015; Volume 1, pp. 829–832.
32. Liu, Q.; Amiri, S.; Ost, L. Low-Cost Intrusion Detection System for CAN Bus Networks Using RISC-V with Binarised Neural Networks. In Proceedings of the 2025 IEEE International Conference on Industrial Technology (ICIT), Yantai, China, 3–6 August 2025.
33. Al-Jarrah, O.Y.; Maple, C.; Dianati, M.; Oxtoby, D.; Mouzakitis, A. Intrusion detection systems for intra-vehicle networks: A review. *IEEE Access* **2019**, *7*, 21266–21289.
34. Pan, L.; Zheng, X.; Chen, H.; Luan, T.; Bootwala, H.; Batten, L. Cyber security attacks to modern vehicular systems. *J. Inf. Secur. Appl.* **2017**, *36*, 90–100.
35. Miller, C.; Valasek, C. Remote exploitation of an unaltered passenger vehicle. In Proceedings of the Black Hat USA, Las Vegas, NV, USA, 1–6 Augest 2015; Volume 2015, pp. 1–91.
36. Palaniswamy, B.; Camtepe, S.; Foo, E.; Pieprzyk, J. An efficient authentication scheme for intra-vehicular controller area network. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3107–3122.
37. Yorozu, T.; Hirano, M.; Oka, K.; Tagawa, Y. Electron spectroscopy studies on magneto-optical media and plastic substrate interface. *IEEE Transl. J. Magn. Jpn.* **1987**, *2*, 740–741.
38. Rangsikunpum, A.; Amiri, S.; Ost, L. An FPGA-Based Intrusion Detection System Using Binarised Neural Network for CAN Bus Systems. In Proceedings of the 2024 IEEE International Conference on Industrial Technology (ICIT), Bristol, UK, 25–27 March 2024; pp. 1–6.
39. Rangsikunpum, A.; Amiri, S.; Ost, L. BIDS: An efficient Intrusion Detection System for in-vehicle networks using a two-stage Binarised Neural Network on low-cost FPGA. *J. Syst. Archit.* **2024**, *156*, 103285.
40. Zhao, Q.; Chen, M.; Gu, Z.; Luan, S.; Zeng, H.; Chakrabory, S. CAN bus intrusion detection based on auxiliary classifier GAN and out-of-distribution detection. *Acm Trans. Embed. Comput. Syst.* **2022**, *21*, 1–30.
41. Hoang, T.N.; Kim, D. Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders. *Veh. Commun.* **2022**, *38*, 100520.
42. Tariq, S.; Lee, S.; Woo, S.S. CANTransfer: Transfer learning based intrusion detection on a controller area network using convolutional LSTM network. In Proceedings of the 35th Annual ACM Symposium on Applied Computing, Brno, Czech Republic, 30 March–3 April 2020; pp. 1048–1055.
43. Yang, L.; Moubayed, A.; Shami, A. MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet Things J.* **2021**, *9*, 616–632.

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.