FPGA Implementation of AI-Based Road Sign Detection for Autonomous Systems

Gabriel Eide
Department of Engineering
Glasgow Caledonian University
Glasgow, United Kingdom
eidegabe98@gmail.com

Godwin Enemali
Department of Engineering
Glasgow Caledonian University
Glasgow, United Kingdom
godwin.enemali@gcu.ac.uk

Ahmed Solyman
Department of Engineering
Glasgow Caledonian University
Glasgow, United Kingdom
ahmed.solyman@gcu.ac.uk

Prof. Ir. Dr. Zakaria Che Muda
Faculty of Engineering and Quantity Surveying
INTI International University (INTI-IU)
Nilai, Malaysia
zakaria.chemuda@newinti.edu.my

***Abstract*—This research investigates the deployment of a quantized convolutional neural network for traffic sign recognition on an embedded FPGA platform. A convolutional neural network was trained on a dataset conforming to the Vienna Convention on Road Signs and Signals to demonstrate the benefits of international standardization in improving classification performance. The trained model was quantized using Brevitas to reduce precision, exported through QONNX, and compiled with the FINN framework for deployment on a PYNQ-Z2 board. A live webcam was integrated to simulate real-time image acquisition for inference. The deployed system achieved an accuracy of 94.45%, demonstrating the feasibility of low bit-width neural networks for real-time, low-latency inference on resource-constrained hardware. This work highlights the critical role of quantization-aware training, model streamlining, and hardware-software co-design in enabling efficient edge AI deployment.SDG alignment—This research advances *SDG 3 (Good Health and Well-Being), target 3.6* and *SDG 11 (Sustainable Cities and Communities), target 11.2* by enabling robust driver-fatigue detection to improve road safety and reduce traffic injuries within safer, more sustainable transport systems.**

***Index Terms*—FPGA, Convolutional Neural Network, Quantization, Traffic Sign Recognition, Edge AI, PYNQ, FINN**

I. Introduction

The increasing integration of Artificial Intelligence (AI) into autonomous systems, particularly within the rapidly evolving automotive industry, fundamentally relies on robust and efficient solutions for environmental perception and decision-making. Traffic Sign Recognition (TSR) is a mission-critical component within the autonomous vehicle stack, allowing the accurate, real-time interpretation of road conditions, regulatory commands, and warnings. The performance and reliability of the TSR system are paramount, directly impacting passenger safety and the overall functionality of the self-driving platform. Consequently, reliance on Deep Learning (DL) models[5], specifically Convolutional Neural Networks (CNNs), has become the industry standard for achieving high recognition accuracy[1].

A. Challenge: Computational Constraints on the Edge

Despite the superior accuracy provided by state-of-the-art CNNs, the deployment of these complex models faces significant computational challenges when migrating from high-performance cloud environments to resource-constrained embedded platforms. Autonomous systems require inference to be performed locally, instantaneously, and with minimal power consumption requirements that often conflict with the sheer memory footprint and computational intensity of traditional 32-bit floating-point (FP32) CNN architectures. Field Programmable Gate Arrays (FPGAs) offer a compelling solution due to their inherent parallelism and high power-efficiency compared to general-purpose CPUs or GPUs. However, unlocking this potential necessitates specialized tool flows to map complex neural networks onto fixed-point, low-bit-width hardware. Furthermore, the variability in road sign designs across different regions poses an additional challenge, underscoring the necessity to investigate how international standardization, such as that provided by the Vienna Convention on Road Signs and Signals, can enhance model generalization and overall performance [3]. Prior works such as [4], though based on the Vienna Convention, lacked substantial hardware results and exhibited low accuracy.

B. Proposed Solution and Core Contribution

This research directly addresses these dual challenges by presenting a comprehensive end-to-end hardware-software co-design pipeline for developing and deploying a highly optimized, low-power Quantized Convolutional Neural Network (QNN) for TSR. The central technical contribution revolves around the use of the Quantization-Aware Training (QAT) paradigm using **PyTorch** and the **Brevitas** framework. QAT allows the model's precision to be aggressively reduced from the standard FP32 representation to a compact 8-bit signed Integer (INT) format while simultaneously mitigating the associated accuracy degradation. This quantized model is then

exported using a dedicated subset of the Open Neural Network Exchange (QONNX) format to preserve the low-bit quantization parameters. Crucially, the model is compiled using the Fast Interpretable Neural Network (FINN) framework, which is specifically designed to generate an optimized, fully synthesized bitstream for efficient deployment on the PYNQ-Z2 FPGA board. This systematic approach culminates in a real-time system, integrated with a live webcam, demonstrating low-latency inference suitable for Intelligent Transportation Systems (ITS) edge applications[4].

C. research Aims and Objectives

The research aims to validate a lightweight, low-power, and highly accurate AI deployment pipeline for edge-based intelligent transportation systems. Specifically, it involves designing a robust CNN for Traffic Sign Recognition (TSR) using PyTorch with $=>$**95%** accuracy [1], applying Quantization-Aware Training via Brevitas [6, 7] to achieve INT8 precision with $<$**4%** accuracy loss [2], exporting to QONNX for PYNQ-Z2 FPGA deployment using FINN [8][15][16]. Finally, the system's latency, power, resource use, and accuracy will be evaluated, highlighting the benefits of compliance with Vienna Convention standards for improved model generalizability [2].

D. Scope and Limitations

The scope of this research is intentionally focused on the development, quantization, and FPGA deployment of a deep learning model for the classification of already localized traffic signs[10], [14]. The work specifically excludes the development of object detection and localization algorithms (which precede classification) and the exploration of alternative FPGA boards or deployment frameworks outside of the established FINN toolflow[9]. The study operates under the assumption that all dataset images adhere to the Vienna Convention for Road Signs and Signals (VCRSS), ensuring a focused approach to solving specific, measurable problems within defined resource and time constraints[18], [19]. The subsequent sections of this paper are organized to detail the methodology, implementation, and rigorous evaluation of the proposed system[12].

II. Methods

This section details the comprehensive, end-to-end methodology employed for developing and deploying a high-performance, for Traffic Sign Recognition (TSR) on a resource-constrained embedded FPGA platform. The approach is defined by a synergistic integration of sophisticated software model development techniques, advanced low-bit quantization methods, and hardware-software co-design utilizing the specialized FINN framework[13].

A. AI Model Development and Preparation

1) Dataset Preparation and Standardization: The selection of a robust and representative dataset is paramount for training deep learning models intended for generalized deployment. The European Traffic Sign Dataset (ETSD) was chosen as the primary source for training and evaluating the TSR models. This dataset is distinguished by its size and diversity, comprising 81,042 images spanning 164 distinct classes of road signs collected across various European countries. Crucially, the dataset's composition aligns with the design principles stipulated by the Vienna Convention on Road Signs and Signals (VCRSS). This standardization alignment is central to the research 's goal of demonstrating enhanced model generalization capabilities.

The raw images, originally encoded in the Portable PixMap (PPM) format, underwent several critical pre-processing steps. To achieve a balance between preserving essential visual features and adhering to the tight memory and resource constraints of the target **PYNQ-Z2 FPGA**, all images were deterministically resized to a fixed resolution of 32×32 pixels. They were subsequently converted to the UINT8 (8-bit unsigned integer) format. To ensure statistical rigor and prevent class imbalance issues during training, the original ETSD split was re-stratified to a rigorous 70% training / 30% testing ratio, which forms the basis of the European Traffic Sign All Class dataset (ETSAC). The total number of input values processed per image is calculated as:

$$\text{Total Inputs per Image} = (\text{Channels} \times \text{Height} \times \text{Width}) = (3 \times 32 \times 32) = 3072 \text{ inputs} \quad (1)$$

2) Model Architectural Design and Iteration: The research adopted an iterative model development strategy, starting with a foundational architecture and progressively introducing complexity and sophistication to enhance both accuracy and efficiency for hardware mapping.

a) Baseline Model: A simple Baseline Model was implemented as a proof of concept. This shallow CNN, with one convolutional layer, a MaxPool layer, and two fully connected layers, was designed for easy hardware implementation and to establish a minimum performance benchmark, validating compatibility with the FINN toolflow.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

num_classes =164

class TrafficSignCNNV1(nn.Module):
    def __init__(self):
        super(TrafficSignCNNV1, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=2, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(32 * 16 * 16, 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.pool(x)
        x = x.view(x.size(0), -1)
        x = self.fc1(x)
        x = self.fc2(x)
        return x
```

Fig. 1. Architectural diagram of the Baseline Model, illustrating the foundational sequential flow of convolutional and pooling operations.

b) Improved Model (Model 3): To address the performance limitations of the Baseline Model, an Improved Model

(Model 3) was developed. This architecture introduced increased depth and complexity to enable more robust and nuanced feature extraction. The network employed two sequential convolutional and MaxPool layers, allowing the model to capture a wider hierarchy of visual features. This increased depth effectively compressed the image dimensionality from 32×32 pixels down to a concentrated feature map of 6×6 pixels before the critical flattening operation, significantly reducing the parameter count for the subsequent dense linear layers and improving computational efficiency[11].

```python
import torch.nn as nn
import torch.nn.functional as F

class CNNvM3(nn.Module):
    def __init__(self, input_channels, num_classes):
        super(CNNvM3, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3)
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(32 * 6 * 6, 128)
        self.relu3 = nn.ReLU()
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool2(self.relu2(self.conv2(x)))
        x = x.view(x.size(0), -1)
        x = self.relu3(self.fc1(x))
        x = self.fc2(x)
        return x
```

Fig. 2. Architectural diagram of the Improved Model (Model 3), highlighting the increased depth for enhanced feature learning.

c) Hierarchical Model (H4): The final, most advanced architecture, the Hierarchical Model (H4), was designed to explicitly leverage the inherent hierarchical structure of the VCRSS. This modular model processes input images through shared convolutional and MaxPool blocks, after which the feature maps branch out to category-specific linear layers. This multi-stage classification pipeline first predicts the top-level Category (e.g., regulatory, warning), then the Subcategory, and finally the definitive Class ID within that subcategory. This design not only optimizes for the varying complexity across different sign classes but also provides more granular diagnostic information for driver assistance systems, linking prediction errors directly to their correct hierarchical level.

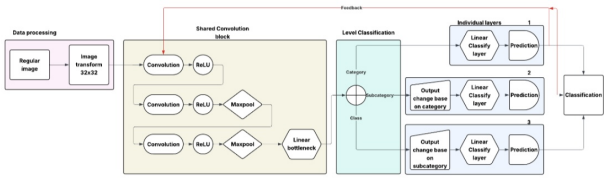

Fig. 3. Block diagram of the Hierarchical Model (H4), illustrating the VCRSS-compliant, multi-level classification structure.

B. Quantization and Hardware Toolchain Optimization

The transition of the floating-point models to the target FPGA requires aggressive reduction in arithmetic precision. This critical step was executed through a controlled methodology to ensure minimal accuracy loss while maximizing hardware efficiency.

1) Quantization-Aware Training (QAT): Instead of less effective Post-Training Quantization (PTQ), Quantization-Aware Training (QAT) was implemented using the Brevitas library within the PyTorch framework. Brevitas simulates the precise low bit-width behavior (specifically 8-bit signed integer, INT8) during the forward pass of training. This allows the network's weights and activations to adapt to the quantization errors, thereby significantly mitigating accuracy degradation. The core process of symmetric quantization maps a real-valued number x to an integer $q(x)$ using a scale factor s:

$$q(x) = \text{round}\left(\frac{x}{s}\right), \quad x \in R, q(x) \in Z \tag{2}$$

The corresponding dequantization (approximation) process, including the zero-point z for asymmetric quantization, is given by:

$$x' = s \times (q - z) \tag{3}$$

During backpropagation, the non-differentiable rounding operation is managed by the Straight-Through Estimator (STE), which approximates the gradient as unity:

$$\frac{\partial \text{round}(x)}{\partial x} \approx 1 \tag{4}$$

This technique is fundamental to allowing the model to learn weights and biases that are resilient to the INT8 constraint.

2) QONNX Export and FINN Compilation: The fully trained and quantized QNN was exported to the Quantized Open Neural Network Exchange (QONNX) format. QONNX is an extension of the ONNX standard designed specifically to preserve all quantization metadata (scales, zero-points, and bit-widths) required for hardware synthesis. The exported QONNX graph served as the input for the Fast Interpretable Neural Network (FINN) framework. FINN is a dedicated hardware compiler for QNNs that streamlines the deployment process through several stages:

1) **Graph Manipulation**: High-level optimization of the QONNX graph (e.g., node fusion, constant folding).
2) **High-Level Synthesis (HLS) Generation**: FINN converts the network description into VHDL/Verilog hardware description language by leveraging dataflow parallelism.
3) **Bitstream Synthesis**: The HLS output is fed into the Xilinx Vivado toolchain to generate a fully synthesizable, highly optimized bitstream (`.bit` file) specifically tailored for the PYNQ-Z2 FPGA fabric.

This flow transforms the software model into a custom hardware accelerator, maximizing throughput and minimizing latency by exploiting the FPGA's spatial computing capabilities.

C. System Deployment and Performance Evaluation

The final optimized bitstream was loaded onto the PYNQ-Z2 board, a cost-effective platform built around the Xilinx Zynq System-on-Chip (SoC). This deployment enables the

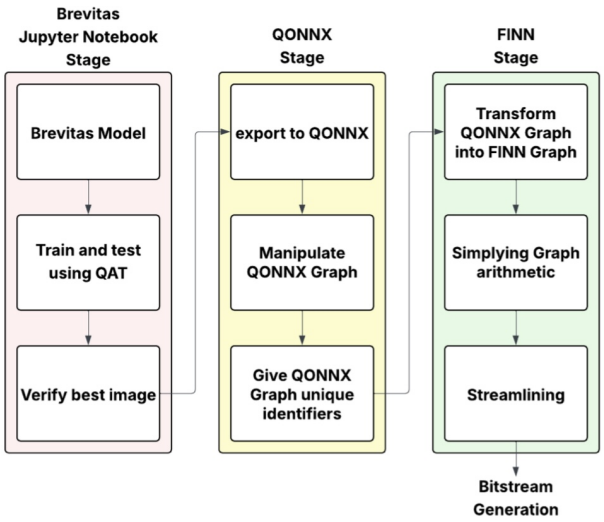

Fig. 4. The end-to-end FINN dataflow, from the quantized PyTorch model to the final FPGA bitstream synthesis.

execution of the QNN entirely on the FPGA's Programmable Logic (PL).

1) Real-Time Simulation: To validate the system's performance under operational conditions, a live webcam module was integrated with the PYNQ-Z2. This setup simulated the real-time image acquisition pipeline typical of an autonomous vehicle's front-facing camera. The video feed was processed dynamically by the deployed accelerator, allowing for robust testing of the model's inference speed and accuracy in a practical, dynamic environment[20].

2) Performance Metrics: The system's performance was rigorously evaluated against multiple dimensions to provide a comprehensive assessment of the embedded AI solution. Key performance indicators included: Inference Latency, FPGA Resource Utilization, Power Consumption, and Classification Accuracy. This multi-faceted evaluation strategy allowed for a direct comparison between the software and hardware implementations, highlighting the trade-offs and benefits achieved through the quantization and FPGA acceleration pipeline. z

III. Results

This section presents the key findings derived from the rigorous evaluation of the developed AI models and their subsequent hardware-accelerated deployment on the PYNQ-Z2 FPGA platform. The results are meticulously organized to cover data processing insights, comparative PyTorch model performance before quantization, the quantifiable effects of Quantization-Aware Training (QAT), and the final synthesized performance metrics on the embedded hardware.

A. Data Processing Insights: The ETSAC Dataset Analysis

Analysis of the **European Traffic Sign All Class dataset (ETSAC)** dataset revealed an imbalance in class distribution, a critical factor for model generalization. As visualized in Figure 5, Category 3 (Informative) was notably underrepresented, comprising 20% of the classes but only 4% of the images. This bias was further exemplified at the subcategory level; for instance, Subcategory 10 contained 26% of all images despite representing only 7% of the classes. This skewed distribution inherently risks skewing the model's predictive capabilities towards overrepresented classes and underscores the impact of dataset composition on final performance.

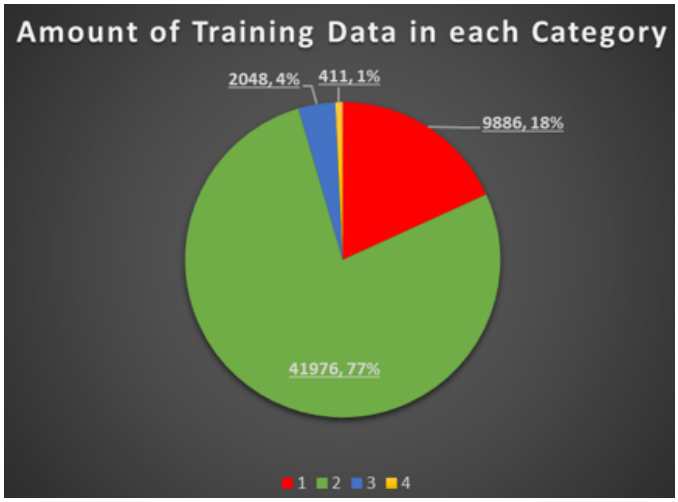

Fig. 5. Distribution of classes and training data images across categories in the ETSAC dataset. The plot highlights the clear statistical imbalance, particularly the underrepresentation of Category 3 (Informative) images relative to its class count.

B. PyTorch Model Performance Evaluation

Comparative analysis across the developed PyTorch models the Baseline Model, Model 3 (Improved), and the Hierarchical Model (H4) demonstrated an iterative improvement in classification accuracy and feature learning efficiency.

1) Model Accuracy Comparison: Model 3 consistently achieved high results, establishing itself as a stable and efficient design candidate for hardware conversion. However, the Hierarchical Model (H4), specifically engineered to align with the VCRSS structure, exhibited the highest accuracy. Figure 6 provides a clear visual summary of this comparative performance, essential for selecting the most suitable candidate for the subsequent hardware deployment stage.

2) Hierarchical Model (H4) Detailed Analysis: The H4 model was subjected to detailed evaluation, including the generation of confusion matrices and accuracy breakdowns at each hierarchical level (Category, Subcategory, and Class ID). The model demonstrated strong performance in classifying the main categories, with subsequent predictions for subcategory and class IDs substantially benefiting from the hierarchical approach Notably, Category 2 (Regulatory) showed consistently high accuracy, a result directly correlated with its larger representation in the training dataset. Figure 7 visually encapsulates this diagnostic detail, which is critical for identifying areas of model strength and pinpointing specific subcategories where misclassifications might occur.

3) Impact of Standardization on Generalization: The evaluation of models across country-specific data provided critical

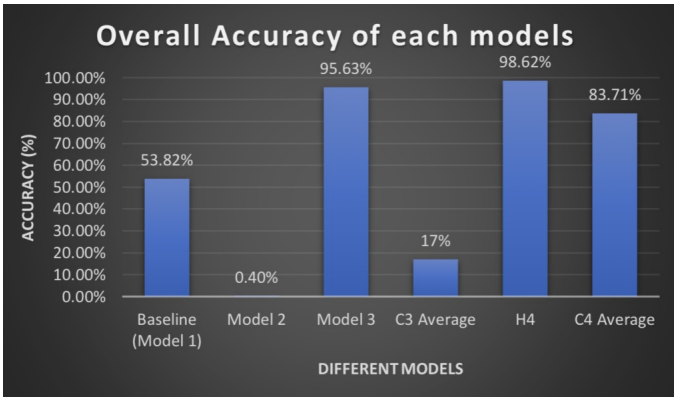

Fig. 6. Comparative performance of PyTorch models. The bar chart summarizes the overall classification accuracy of the Baseline, Model 3, and H4 architectures on the ETSAC dataset.

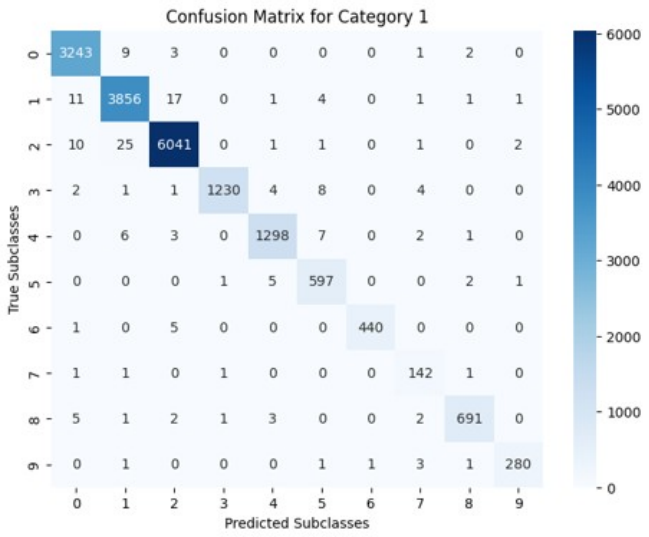

Fig. 7. H4 Model accuracy and confusion matrices for categories and subcategories. This visualization provides a multi-level performance breakdown critical for VCRSS-compliant systems.

insights into the impact of international standardization. Countries whose traffic signs adhere more closely to the Vienna Convention on Road Signs and Signals (VCRSS) generally exhibited higher classification accuracies. This suggests that the standardization of visual design significantly enhances the model's ability to generalize. Figure 8 illustrates this effect, allowing for a direct comparison of model performance against varying national interpretations of the VCRSS.

C. Quantization and FPGA Deployment Metrics

1) Quantization Performance: The Quantization-Aware Training (QAT) process, implemented using the Brevitas library,[7] successfully reduced the model's numerical precision to 8-bit signed integers (INT8) . Crucially, this reduction was achieved with minimal accuracy degradation, well within the targeted objective of maintaining degradation under 4%

2) FPGA Deployment and System Metrics: The FINN framework was successfully employed to convert the QONNX

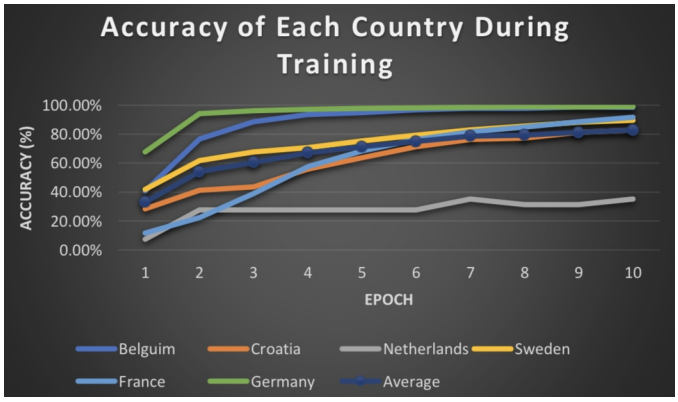

Fig. 8. Performance of models across different countries. The comparative chart highlights how well Model 3 and H4 generalize to traffic signs from countries with varying degrees of VCRSS adherence.

Fig. 9. Brevitas quantization testing results. The bar chart compares the accuracy of the full-precision model against the 8-bit quantized model, demonstrating minimal accuracy degradation.

model into a fully synthesizable and highly optimized bitstream tailored for the PYNQ-Z2 FPGA. This deployment demonstrated the feasibility of achieving real-time, ultra-low-latency inference on the resource-constrained hardware[17].

Key performance metrics measured on the PYNQ-Z2 include:

- **Overall Accuracy**: The deployed system achieved an overall classification accuracy of 94.45%. this represents a final degradation of only 1.18% compared to the software-only QNN, remaining comfortably within the acceptable tolerance.
- **Resource Utilization**: Quantification of the hardware resources (e.g., LUTs, FFs, BRAMs) confirmed the efficiency of the FINN-generated core, showcasing its ability to maximize throughput while minimizing resource footprint.
- **Inference Latency and Power Consumption**: Metrics for latency and power consumption were measured, validating the low-power and high-speed efficacy required for edge AI applications.

TABLE I
FPGA RESOURCE UTILIZATION AND PERFORMANCE METRICS.

Resource	Utilization	Available	Utilization (%)
LUT	38 614	53 200	72.58
LUTRAM	2205	17 400	12.67
FF	42 643	106 400	40.08
BRAM	126	140	89.64
DSP	5	220	2.27
BUFG	2	32	6.25

IV. Discussion

A. Discussion

The comprehensive performance analysis confirms the technical feasibility and robust efficacy of deploying a Quantized Neural Network (QNN) for Traffic Sign Recognition (TSR) on a resource-constrained embedded FPGA platform.

1) Model Performance and Trade-offs: The evaluation of the developed architectures revealed a critical trade-off between architectural complexity and hardware amenability. The Hierarchical Model (H4) achieved the highest classification accuracy at 98.62**%,** validating the hypothesis that aligning the network structure with the Vienna Convention on Road Signs and Signals (VCRSS) hierarchy enhances predictive power. However, this superior performance came at the cost of increased complexity, which complicates the FINN hardware synthesis process and resource utilization. Conversely, Model 3, with an accuracy of 95.63%, offered the most favorable balance. Its simpler, yet highly effective, design established it as the optimal candidate for hardware deployment, demonstrating that high accuracy can be maintained without excessive architectural overhead when targeting FPGA synthesis.

2) The Critical Impact of Dataset Imbalance: A notable and crucial correlation was empirically established between the number of images within specific categories and their corresponding classification accuracies. Categories and subcategories characterized by data scarcity consistently demonstrated the lowest performance, underscoring the critical impact of dataset balance on model generalization. For instance, the severely underrepresented Category 0, Subcategory 1, which comprised only 0.08% of the total dataset images, exhibited a poor classification accuracy of 60**%.** This finding conclusively highlights that data paucity is the most significant limiting factor to generalization, necessitating advanced strategies to mitigate this distributional bias in future work.

3) Validation of International Standardization: The research provided strong empirical evidence supporting the research 's hypothesis concerning the benefits of standardization. The **C3 model**, naively trained on heterogeneous country-specific datasets without VCRSS compliance, performed poorly with an average accuracy of only **17%**. This result confirms that deploying separate models for every country without standardization is computationally untenable. In stark contrast, the **C4 model**, which leveraged VCRSS principles, achieved a commendable average accuracy of **83%**, showcasing the dramatic robustness gained from adherence to a unified design standard. Furthermore, the comparison between Belgium (high class diversity) and Germany (high image volume in specific categories) suggests that diversity of images across classes is more impactful than sheer sample volume when training for robust generalization in heterogeneous traffic environments.

4) Hardware Acceleration and Quantization Fidelity: The deployment process successfully validated the effectiveness of the chosen hardware toolchain. Quantization-Aware Training (QAT) using Brevitas successfully constrained the model's precision to 8-bit signed integers, resulting in a minimal accuracy decrease of only 1.94% compared to the full-precision Model 3. This outcome was comfortably within the targeted 4% degradation objective. The final deployment on the PYNQ-Z2 board further affirmed the viability of real-time, low-latency inference. The deployed AI model experienced only an additional 1.18% accuracy loss compared to the Brevitas software emulation, confirming the high fidelity of the FINN framework in mapping the QNN to the FPGA with negligible computational error and successfully meeting the objective for efficient edge AI deployment.

V. Conclusion

This research successfully demonstrated the design, implementation, and evaluation of a machine learning-based Traffic Sign Recognition (TSR) system, highlighting a complete end-to-end pipeline. The feasibility of integrating quantized neural networks with Field-Programmable Gate Arrays (FPGAs) was established, achieving real-time, low-latency performance within resource-constrained environments. Through techniques such as model quantization, hardware acceleration, and FINN/QONNX optimizations, the system maintained high classification accuracy while maximizing efficiency.

Multiple Convolutional Neural Network (CNN) models were developed, with the most accurate achieving 98.62% classification accuracy. A key achievement was the conversion of the CNN into a Quantized Neural Network (QNN) using Quantization-Aware Training (QAT) tools, quantizing the model to 8-bit unsigned integers (UINT8). This quantized model was successfully exported and manipulated using the FINN/QONNX framework, retaining its high accuracy. The final deployment on a PYNQ-Z2 board yielded an accuracy of 94.45%, representing a degradation of only 1.18% from the software-only QNN, well within the research 's objective of less than 4% degradation. This work underscores the critical role of quantization-aware training, model streamlining, and hardware-software co-design in enabling efficient and accurate edge AI deployment for autonomous systems.

References

[1] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[2] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German traffic sign recognition benchmark: A multi-class classification competition," in *Proc. IJCNN*, pp. 1453–1460, 2011.

[3] E. Nurvitadhi et al., "Can FPGAs beat GPUs in accelerating next-generation deep neural networks," in *Proc. ACM/SIGDA Int. Symp. FPGA*, 2017.

[4] R. Hmida, A. Abdelali, and A. Mtibaa, "Hardware implementation and validation of a traffic road sign detection and identification system," *J Real-Time Image Proc* 15, 13–30, 2018.

[5] "PYNQ," Xilinx, 2025. [Online]. Available: https://www.pynq.io/board.html

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[7] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[8] "Brevitas: Quantization-aware training in PyTorch," Xilinx, 2025. [Online]. Available: https://github.com/Xilinx/brevitas

[9] "Open Neural Network Exchange (ONNX)," ONNX Community. [Online]. Available: https://onnx.ai/

[10] "FINN: A framework for fast, scalable binarized neural network inference," Xilinx Research Labs, 2025. [Online]. Available: https://xilinx.github.io/finn/

[11] Y. Umuroglu et al., "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. IEEE FPGA*, 2017.

[12] "Vitis AI: AI inference development platform," Xilinx, 2024. [Online]. Available: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html

[13] J. Duarte et al., "Fast inference of deep neural networks in FPGAs for particle physics," *J. Instrum.*, vol. 13, 2018.

[14] G. Sikander, S. Anwar, G. Husnain, R. Thinakaran, and S. Lim, "An Adaptive Snake Based Shadow Segmentation for Robust Driver Fatigue Detection: A 3D Facial Feature Based Photometric Stereo Perspective," *IEEE Access*, vol. 11, pp. 99178–99188, 2023, doi: 10.1109/ACCESS.2023.3312576.

[15] Z. Lin et al., "Benchmarking deep learning frameworks and investigating FPGA deployment for traffic sign classification," *IEEE Access*, vol. 4, pp. 385–395, 2019.

[16] W. Farhat et al., "Embedded system for road sign detection using MicroBlaze," in *Proc. IEEE Conf.*, pp. 1–5, 2015.

[17] H. H. M. et al., "FPGA-based system for road signs detection," in *Proc. IEEE Conf.*, pp. 1–6, 2017.

[18] G. Sikander, S. Anwar, G. Husnain, R. Thinakaran, and S. Lim, "An adaptive snake based shadow segmentation for robust driver fatigue detection: A 3D facial feature based photometric stereo perspective," *IEEE Access*, vol. 11, pp. 99178–99188, 2023.

[19] "Machine learning inference on FPGAs," Xilinx, White Paper, Jun. 2017.

[20] A. T. Abu–Jassar, H. Attar, A. Amer, V. Lyashenko, V. Yevsieiev, and A. Solyman, "Development and Investigation of Vision System for a Small-Sized Mobile Humanoid Robot in a Smart Environment," *International Journal of Crowd Science*, vol. 9, no. 1, pp. 29–43, 2025.