Hybrid Weightless Neural Networks for Efficient Edge Inference

Mugdha Jadhao
The University of Texas at Austin
Austin, USA
mugdhaj17@utexas.edu

Alan T. L. Bacellar
The University of Texas at Austin
Austin, USA
alanbacellar@utexas.edu

Shashank Nag
The University of Texas at Austin
Austin, USA
shashanknag@utexas.edu

Igor D. S. Miranda
Federal University of Recôncavo da Bahia
Cruz das Almas, Brazil
igordantas@ufrb.edu.br

Felipe M. G. França
Instituto de Telecomunicações
Porto, Portugal
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
felipe@ieee.org

Lizy K. John
The University of Texas at Austin
Austin, USA
ljohn@ece.utexas.edu

***Abstract*—Deploying fast, accurate, and efficient machine learning on edge devices remains a key research challenge. While deep neural networks (DNNs) excel in vision tasks, their computational and storage demands hinder deployment on resource-constrained hardware. Optimization strategies such as quantization, sparsity induction, and multiplication-free architectures have been explored to address these challenges. Weightless Neural Networks (WNNs), based on look-up tables, offer high energy efficiency and low-latency inference, making them attractive for edge applications. However, WNNs struggle with complex vision tasks due to their lack of support for positional invariance. To enable machine learning models that are both lightweight and accurate for edge inference, we propose a hybrid weightless neural network model (H-WNN) that integrates the efficiency of WNNs with the spatial feature extraction capabilities of quantized convolution. FPGAs serve as an ideal platform for exploring the hybrid approach, as they facilitate efficient design space exploration for quantized convolution, allow direct mapping of look-up tables in WNNs, and enable seamless integration of both models. We additionally present a workflow to explore hardware-accuracy tradeoffs for H-WNN models on hardware platforms. We evaluate H-WNN on multiple classification datasets relevant to edge applications and compare them against similar existing studies, such as FINN-R and DWN. Across the benchmarks, our results consistently demonstrate that H-WNN achieves lower resource usage and latency while maintaining competitive accuracy and throughput. For example, on CIFAR-10, H-WNN achieves 87.72% accuracy while being 2× smaller than competing approaches at the same throughput, resulting in improved energy efficiency.**

***Index Terms*—Light-weight inference, efficient edge computing, convolutional networks, weightless neural networks**

I. Introduction

Deep Neural Networks (DNNs) have achieved remarkable success in the field of computer vision, enabling a wide range of applications and use cases. However, their substantial computational and storage demands pose major challenges for deployment, particularly in resource-constrained edge environments where high energy efficiency, low latency, and high throughput are essential for real-time and safety-critical applications [3]. Various optimization techniques like pruning [4]–[6] and sparse neural networks [7]–[9] have been proposed that help reduce the model size and computational needs.

Among optimization strategies, quantization is a promising approach for reducing redundancy in neural networks by converting floating-point parameters to lower precision. This shift results in a smaller memory footprint, reduced power consumption, and lower hardware cost. However, it typically comes with a loss in accuracy, especially for complex tasks. Increasing the network layer size has been shown to help compensate for this accuracy loss [10], [11], but this approach can be counterproductive for edge deployment, where resource constraints are a primary concern.

Another approach to address the limitations in the computational efficiency of DNNs, which stem from the inherent cost of multiplication, is the use of multiplication-free architectures such as Binary Neural Networks (BNNs) [12], Weightless Neural Networks (WNNs) [13], DiffLogicNet [14], and TreeLUT [15]. Among these, WNNs stand out as a distinct category due to their impressive computational efficiency. WNNs eliminate weighted connections, instead utilizing lookup tables (LUTs) with binary values to drive neural activity. This approach enables WNNs to capture highly non-linear behaviors with minimal arithmetic.

Recently, researchers proposed Differentiable Weightless Neural Networks (DWNs) [2], enabling the development of multi-layer WNNs with interconnected LUTs. This approach outperforms fully-connected BNNs [16] and DiffLogicNet [14] in latency, throughput, energy efficiency, and model area across various benchmarks. Particularly, they demonstrate significant energy savings, achieving up to a 135× reduction in energy costs in FPGA implementations compared to both the prior works. Compared to other LUT-based NNs like LogicNets [17], PolyLUT [18], and NeuralLUT [19], DWN models achieve up to 43x less hardware usage and 1.5x reduced

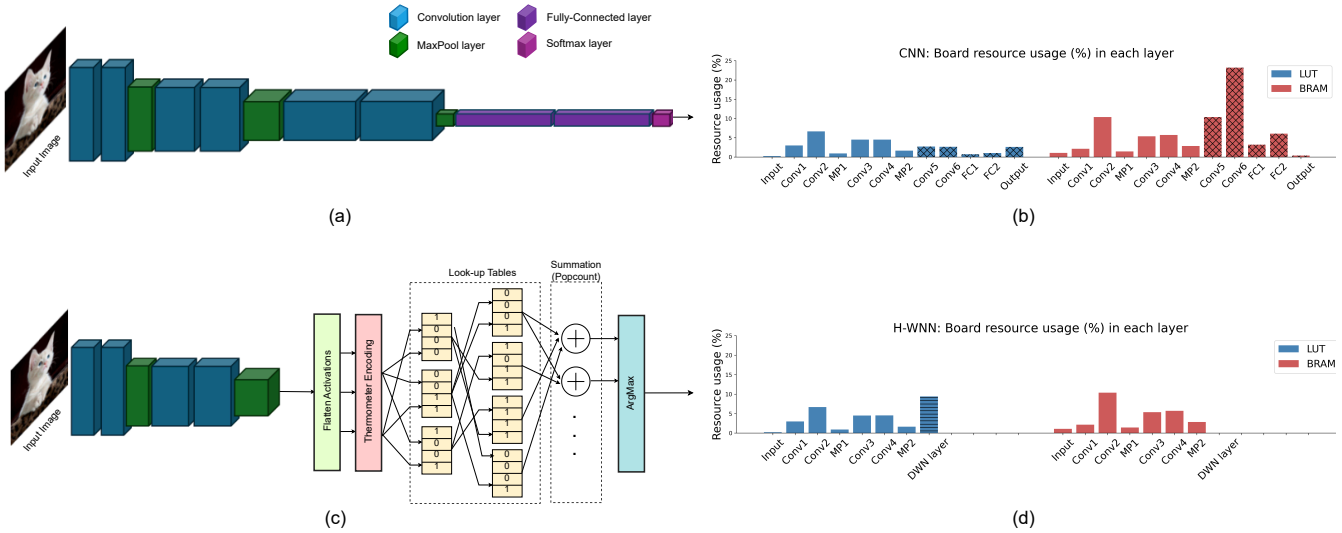

Fig. 1: **Hybrid Weightless Neural Network (H-WNN) Architecture** designed for lightweight and accurate edge inference. (a) A generic example CNN comprising three stages of convolutional layers with max-pooling, followed by fully connected layers and a softmax output. (b) Layer-wise hardware resource utilization (%) of the CNN using FINN-R [1], showing high resource usage in the later layers. (c) The proposed H-WNN integrates quantized convolution layers for global feature extraction with weightless DWN [2] layers for learning finer, localized features with minimal hardware overhead. (d) Resource usage of the H-WNN, demonstrating a 60% reduction compared to the CNN model, with the same accuracy and throughput.

latency for similar accuracies in FPGA implementations.

While DWNs show substantial lower hardware costs, they face limitations in handling complex vision tasks that often require convolutional operations to capture spatial hierarchies and patterns within images. The lack of native support for convolution in DWNs restricts their effectiveness in such tasks. As stated above, although quantized convolutional neural networks (QNNs) are well-suited for vision applications due to their spatial processing capabilities, they generally require larger models to minimize the accuracy gap compared to full-precision counterparts [20], [21].

This reveals a gap where neither DWNs nor QNNs alone fully meet the need for both good accuracy and hardware efficiency on edge platforms. Addressing this challenge requires an innovative approach that combines the hardware efficiency of DWNs with the spatial processing strengths of convolution, enabling models that are both accurate and lightweight for edge-based vision tasks. In this work, we introduce a hybrid weightless neural network (H-WNN) model illustrated in Figure 1 that adeptly addresses all the aforementioned challenges of inference under energy and resource constraints.

Figure 1(a) illustrates a generic Convolutional Neural Network (CNN) architecture consisting of convolutional, fully connected, and max-pooling layers. The model shown includes three stages, each with two convolutional layers followed by a max-pooling layer, with progressively increasing channel sizes. It concludes with two fully connected layers and a softmax output layer. The early convolutional stages help the network learn broader spatial features and establish positional invariance in a global context, while the deeper convolutional layers capture finer, localized patterns. Finally, the fully connected layers integrate these learned features to enable complex decision-making for classification [22].

Figure 1(b) presents the layer-wise hardware resource utilization (%) for the model depicted in Figure 1(a), evaluated using single-bit precision for both weights and activations. The model is deployed for maximum throughput using the FINN-R framework [1], which exhaustively maps the design onto available hardware resources. The results indicate that the later convolutional layers consume substantially more hardware than earlier layers, primarily due to the increased filter sizes needed to extract fine-grained image features. Additionally, the fully connected layers account for approximately 14% of the total hardware usage.

Previous works on WNNs have demonstrated their ability to efficiently learn input subpatterns, which are used as addresses to access each LUT [2], [23], [24]. However, conventional WNN architectures are limited in their ability to capture global pattern dependencies. Building on the earlier analysis, this limitation reveals a promising design opportunity: replacing the later layers of a QNN—which are responsible for learning fine-grained, localized features—with a WNN to achieve a more favorable trade-off between spatial expressiveness and hardware efficiency.

Figure 1(c) illustrates the hybrid architecture of the H-WNN model, which integrates quantized convolution operations in the initial layers for feature extraction in a global context, followed by weightless layers for learning finer, localized features with minimal hardware overhead. The quantized layers also handle input encoding, providing a strong foundation for the

weightless stages. As shown in Figure 1(d), this design reduces hardware resource usage by 60% compared to the FINN-R-based implementation shown in Figure 1(b) while maintaining comparable throughput. This leads to improved energy efficiency and brings WNNs closer to leveraging convolutional processing for vision tasks. The resulting resource savings can be leveraged to either unfold the QNN architecture for higher throughput or increase model precision to improve accuracy, offering flexibility based on application requirements.

Thus, we present H-WNN, a new class of hybrid neural architectures that strategically combines the strengths of quantized convolutions and weightless inference. This design achieves an effective balance between spatial feature extraction and hardware efficiency, specifically tailored for edge deployment. In this paper, we introduce the H-WNN architecture alongside a comprehensive training and deployment workflow and demonstrate its effectiveness across multiple edge-relevant vision benchmarks. Our evaluations show that H-WNN models consistently achieve significant reductions in latency and hardware resource usage while matching or surpassing the accuracy of existing quantized baselines. This positions H-WNN as a practical and scalable solution for energy-efficient inference in resource-constrained environments.

II. Background

A. Quantized Neural Networks

In recent years, the machine learning community has shifted from using high-precision floating-point operations to lower-precision floating-point, fixed-point, or integer values. This transition aims to exploit the redundancy in trained model parameters, thus reducing the computational cost of inference. This technique, known as quantization, decreases memory requirements by allowing models to fit into more compact formats. As a result, QNNs can be entirely stored in on-chip memory, minimizing reliance on off-chip memory and consequently reducing energy consumption [1], [25]–[27]. On-chip memory also provides high bandwidth, improving the efficiency of compute resource utilization and boosting overall performance. Replacing floating-point arithmetic with fixed-point or integer operations drastically reduces the processing power required. In addition, quantized operators are more compact than floating-point operators, which helps lower hardware costs and increases computational density within the same resource constraints.

An extreme form of quantization is Binary Neural Networks (BNNs) [12], where both weights and activations are constrained to binary values. This allows BNNs to replace traditional multiply-accumulate operations with simple XNOR and popcount operations, significantly lowering hardware costs and enabling multiplication-free computation. In a fully-connected BNN, each neuron performs a binary operation on input values with binarized weights, sums them with a bias, and applies a sign function to produce binary activations (+1 or -1). However, the reduced precision in QNNs—particularly in BNNs—often results in accuracy degradation, especially for complex tasks requiring high representational capacity. To mitigate this, quantized models may need to be scaled up or calibrated carefully, which can offset some of the resource efficiency gains achieved through quantization [10], [11], [21], [28].

B. Weightless Neural Networks

Weightless Neural Networks (WNNs) [2], [13], [23], [24] represent a class of neural models inspired by the dendritic structures of biological neurons. Unlike traditional neural networks that rely on weighted connections, WNNs adopt a multiplication-free approach. Instead of weights, they use an n-input lookup table (LUT) with 2^n learnable single-bit entries as the fundamental unit of computation. This architecture enables the LUT to represent 2^{2^n} possible functions, a significant increase compared to a BNN, where a single XNOR-and-popcount neuron can only represent $2^n + 2$ learnable functions. This difference illustrates the high learning capacity of LUT-based neurons compared to traditional DNN neurons. Moreover, the complete absence of multiply-accumulate (MAC) operations in WNNs allows for the development of high-throughput, energy-efficient models, making them particularly well-suited for edge computing environments where power and resource constraints are paramount.

C. Related Work

Several works in the past have explored the idea of QNNs and multiplication-free convolutional networks to develop efficient neural networks for edge inference.

FINN-R: FINN-R [1] is a continuously evolving exploratory tool from Xilinx (AMD) designed to explore the design space for resource footprint and throughput constraints for QNN inference, with an emphasis on dataflow-style architectures on FPGAs. These architectures are primarily aimed at creating fully rate-balanced systems that maximize throughput by utilizing hardware resources until they are exhausted. It has demonstrated excellent performance in resource-constrained environments, achieving high throughput and energy efficiency while maintaining competitive accuracy on datasets like MNIST [29], CIFAR-10 [30] and SVHN [31]. FINN-R has been widely utilized to deploy efficient applications and use cases in challenging edge environments [32]–[34].

Other Quantized Neural Network Frameworks: ReBNet [35] proposes a multi-level binarization technique for neural networks that offers an accuracy-throughput tradeoff without area overheads. However, this technique is not optimized for edge devices with resource constraints. Other frameworks like WRPN [11] and DoReFa-Net [36] also explore quantization strategies but typically rely on increased width or custom bit-precision that can lead to higher hardware costs.

Differentiable Weightless Networks: A notable recent advancement in the area of WNNs is the development of Differentiable Weightless Networks (DWNs) [2]. DWNs extend the concept of WNNs by enabling multi-layer interconnected lookup tables. DWNs are trained using a novel approach based on an extension of the Extended Finite Difference method, allowing for approximate differentiation of binary values. In

performance tests, DWNs have demonstrated superior performance over fully-connected BNNs implemented on the Xilinx FINN [16] platform under iso-accuracy conditions, showing improvements in latency, throughput, energy efficiency, and model area in FPGA implementations across datasets such as MNIST [29], FashionMNIST [37], KWS, and ToyADMOS [38], with the exception of CIFAR-10 [30], where a large accuracy drop is observed. A significant limitation of DWNs, and WNNs in general, is their lack of support for convolution operations. This limitation hinders their ability to effectively handle tasks that require positional invariance, such as those common in vision applications.

Other LUT-Based Neural Networks: Beyond DWNs, several efforts have explored logic-inspired architectures using lookup tables. TreeLUT [15] implements decision trees as trainable neural components mapped to logic gates, achieving impressive energy savings and interpretability. PolyLUT [18] and NeuralLUT [19] offer approaches that embed polynomial and neural transformations within LUTs to improve expressiveness, while LogicNets [17] use logic synthesis to replace arithmetic-heavy neural layers with Boolean functions. However, most of these approaches are limited in scalability or convolutional support. Truth Table Net [39] demonstrates a deep convolutional network built from Boolean truth tables but achieves only 70% accuracy on CIFAR-10 with substantial resource usage, limiting its practicality for real-world vision applications.

III. Methodology

We illustrate our methodology through representative use cases to better convey analysis and insights; however, the approach itself is broadly applicable across diverse scenarios. As a primary case study, we focus on a QNN model well-suited for edge deployment and representative of typical CNN architectures. Specifically, we adopt the CNV-6 topology, a model introduced in the FINN framework [16], inspired by BinaryNet [40] and VGG-16 [41]. CNV-6 consists of three convolutional stages, each consisting of two 3×3 convolutional layers followed by a 2×2 max-pooling layer, with channel sizes progressively increasing from 64 to 128 to 256. The network concludes with two fully connected layers, each containing 512 neurons.

In this work, we implement the quantized convolutional layers of the CNV-6 model using the FINN-R framework [1]. Beyond the advantages outlined in Section II, FINN-R provides detailed layer-wise hardware utilization through its dataflow-style architecture, enabling precise and fair comparisons. Its fully rate-balanced design also facilitates accurate estimation of per-layer latency. While we adopt FINN-R in this study for its profiling capabilities and throughput-optimized deployment, the proposed methodology is framework-agnostic and can be implemented using any quantization platform.

A. Profiling Analysis and Insights

We deploy the QNN model for maximum throughput by utilizing all available hardware resources and characterize it under various precision settings. The results for the CNV-6 model, evaluated at single-bit (1w1a) and two-bit (2w2a) weight and activation precisions at a clock period of 100 MHz, are summarized in Figure 2. The "_w_a" notation denotes the bit-width of the weights and activations in the QNN.

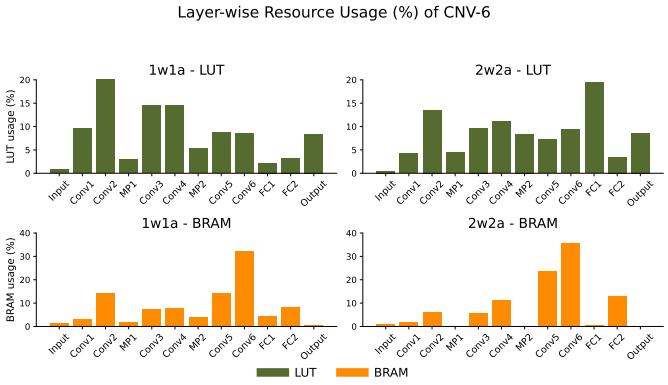

Fig. 2: Layer-wise hardware resource usage (LUT and BRAM18s) (%) for the CNV-6 model in 1w1a and 2w2a configurations. The final convolutional layer (`Conv6`) contributes to the majority of resource usage, particularly in higher-precision settings. The "_w_a" notation refers to the bit-width of the weights and activations in a QNN.

As shown in Figure 2, the later convolutional layers consume significantly more hardware resources than the earlier layers, primarily due to the increasing number of output channels (64, 128, 256) required to capture finer-grained image features. At 1w1a precision, *Conv4* accounts for approximately 14.5% of total LUT usage, while *Conv6* consumes around 32% of the total BRAM18s utilized by the model on the FPGA board. Under the higher-precision 2w2a setting, *Conv6* becomes even more resource-intensive, contributing 9.4% of total LUT usage and 36% of total BRAM consumption due to the increased memory requirements for storing higher bit-width weights.

The profiling analysis not only reveals the disproportionate hardware demands of deeper convolutional and fully connected layers, but also motivates the need to rethink how representational learning depth and hardware efficiency can be balanced in neural architectures. The initial convolutional layers in a neural network contribute to learning larger spatial features and establishing positional invariance in a global context. In contrast, the later convolutional layers focus on learning finer details in smaller regions of the image. Meanwhile, the fully connected layers are responsible for learning patterns necessary for final classification [22]. Previous work on WNNs has demonstrated their effectiveness at learning sub-patterns from smaller input data [2], [23], [24].

B. Hybrid Weightless Neural Network (H-WNN) Design

The above analysis highlights a natural opportunity to combine the later layers of QNNs (convolutional and fully connected) with a WNN model. In the proposed hybrid architecture, we retain quantized convolution operations in

TABLE I: Maximum resource savings (LUT & BRAM18) (%) achievable by replacing all layers from the design point onward in the CNV-6 QNN model for different precisions. Notably, all BRAM usage is eliminated since WNNs do not require BRAMs.

Layer Replacement	1w1a (%)		2w2a (%)	
	LUT	BRAM	LUT	BRAM
FC1 + FC2 + Output	13.83	13.37	31.49	14.51
Conv6 + ... + Output	22.35	45.54	40.86	49.82
Conv5 + ... + Output	31.10	59.90	48.06	73.63
Conv4 + ... + Output	50.91	71.78	67.61	84.99
Conv3 + ... + Output	65.39	79.21	77.29	90.85

the early layers to enable effective feature extraction and positional invariance in the global context while replacing the resource-intensive later layers with a smaller, energy-efficient, and low-latency WNN that captures fine-grained features. Table I presents the maximum achievable resource savings by replacing all layers from the selected design point onward in the CNV-6 QNN. A notable expected outcome is complete BRAM18 savings, as WNNs are LUT-based and do not require BRAMs. We use DWNs as the weightless component of our hybrid architecture throughout this study.

Figure 1 illustrates the proposed hybrid weightless neural network (H-WNN) model, designed for inference under energy and resource constraints. Figure 1(a) shows a typical CNN architecture. For each design point where a layer and all subsequent quantized layers are selected for replacement, as shown in Figure 1(b), intermediate activations from the layer preceding the selected layer are flattened and passed through thermometer encoding [42], where they are binarized and fed into the DWN model. As illustrated in Figure 1(c), the encoded output is grouped into tuples and concatenated to form addresses for the first layer of lookup tables (LUTs). The binary LUT outputs are then used to generate addresses for subsequent layers. Finally, the outputs from the last LUT layer are summed to derive the scores for each class, followed by an argmax layer to determine the output class.

C. Design Workflow

Training the H-WNN model begins with training a quantized CNN using a QNN training strategy of choice. In this study, we adopt the method described by Courbariaux et al. [40]. To train the H-WNN, we use a progressive training strategy inspired by transfer learning paradigms [43]. For each selected design point, we first freeze the weights of the QNN layers preceding the design point and train the DWN independently. This is done by passing intermediate activations from the residual QNN through a flattening and encoding process (e.g., thermometer encoding for higher-precision models), with hyperparameters explored across multiple configurations to identify the optimal setup. In the second stage of training, we unfreeze the QNN and jointly fine-tune both the QNN and DWN components in an end-to-end manner. Finally, the trained hybrid model is evaluated on the test dataset.

The maximum achievable throughput of H-WNN models is constrained by the convolutional layers in the QNN component, as WNNs typically exhibit significantly higher throughput in practical implementations. We implement the quantized convolutional layers using the FINN-R framework [1], targeting maximum throughput by fully utilizing the available hardware resources on the FPGA board. The DWN layers are implemented based on the original DWN design [2], with their lookup tables aligned to the native LUT size of the FPGA to maximize hardware efficiency. This workflow enables the exploration of hardware-accuracy trade-offs for H-WNN models. It is generic, scalable, and can be extended to other DNN topologies and precision levels.

IV. Evaluation

A. Evaluation Methodology

We compare H-WNN against the FINN-R [1] framework, developed by Xilinx Research Labs (AMD), which targets QNN inference on FPGAs. Although FINN-R does not introduce a novel QNN algorithm, it offers a comprehensive toolchain for generating hardware accelerators from pretrained QNNs. As such, evaluating H-WNN against FINN-R provides both hardware-level insights and a representative benchmark within the broader quantized neural network literature. The rationale behind selecting FINN-R as our baseline is discussed in more detail in Section III.

To ensure a fair comparison, we re-generate both software and hardware results for FINN-R using its latest publicly available release [44]. All implementations are configured for maximum throughput, aligning with the design goals of H-WNN, which also targets high-throughput operation in edge deployments. QNN models for FINN-R are trained using the Brevitas [45] low-precision machine learning library, following training pipelines recommended by the framework. For a balanced comparison, we report hardware usage only for the IP core generated for the QNN, excluding the full FPGA accelerator wrapper typically included in FINN-R's end-to-end deployments.

We evaluate the effectiveness of H-WNNs on the following datasets:

(1) *Keyword spotting (KWS)*: This dataset is extracted from Speech Commands v2 [38]. It includes 105,829 utterances across 2,618 speakers, targeting the classification of ten spoken keywords and one "unknown" category. Inputs are converted into spectrograms.

(2) *CIFAR-10*: An image classification dataset consisting of 32×32 RGB images in 10 classes [30].

(3) *SVHN*: Street View House Numbers dataset comprises 32 x 32 RGB images of house numbers derived from Google Street View images. classes [31].

For our hardware implementation and comparison, we target FPGAs. In addition to enabling rapid prototyping without the fabrication overhead of custom ICs, FPGAs are particularly well-suited for exploring the proposed hybrid approach. They

TABLE II: Comparison between H-WNN and Prior Work (FINN-R) on a Pynq-Z1 board at a clock speed of 100 MHz. H-WNN achieves competitive accuracy while offering smaller parameter sizes, significantly reduced hardware resource utilization, and lower latency at the same throughput as the FINN-R models.

Dataset	Model	Test Accuracy (%)	Parameter Size (KiB)	LUT6s (1000s)	BRAM18s	Latency (ms)
KWS	FINN-R (1w1a)	80.23	188.34	16.58	202	2.56
	H-WNN (L)	81.52	207.69	22.87	110	1.60
	H-WNN (M)	80.76	162.05	21.42	81	1.28
	H-WNN (S)	78.22	139.98	17.74	42	0.64
CIFAR-10	FINN-R (1w1a)	84.19	188.34	16.58	202	2.56
	FINN-R (2w2a)	88.63	376.67	25.87	273	9.12
	H-WNN (L)	88.23	306.52	20.72	235	6.84
	H-WNN (M)	87.72	207.69	23.30	137	5.70
	H-WNN (S2)	85.71	162.05	29.44	72	4.56
	H-WNN (S1)	84.62	139.98	22.87	110	1.60
SVHN	FINN-R (1w1a)	93.24	188.34	16.58	202	2.56
	H-WNN (L)	94.54	166.81	17.28	175	1.92
	H-WNN (M)	93.72	158.04	22.87	110	1.60
	H-WNN (S)	88.44	117.23	20.14	57	0.96

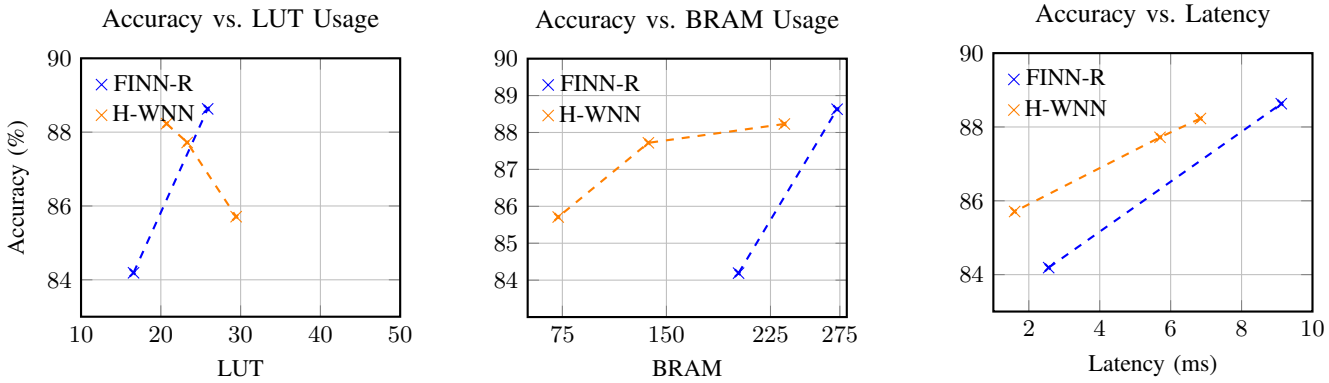

Fig. 3: Accuracy versus hardware usage (LUTs and BRAMs) and latency for FINN-R and H-WNN models. H-WNN achieves higher accuracy than FINN-R while using significantly fewer BRAMs. FINN-R models exhibit higher latency, whereas H-WNN models provide a better trade-off between accuracy and latency across configurations. No consistent correlation is observed between LUT usage and accuracy.

support efficient design space exploration for quantized convolutional architectures, offer the flexibility to directly map look-up tables used in WNNs, and allow seamless integration of both QNN and WNN components within a unified pipeline. We use the Pynq-Z1 board [46], which was also employed in FINN-R [1] evaluations. This board is a compact FPGA platform designed for edge computing and resource-constrained environments, offering 53.2k LUTs and 280 BRAM18 blocks. Consistent with prior work, all designs are synthesized for a 100 MHz clock frequency. Hardware synthesis is performed using Vivado 2022.1 in out-of-context (OOC) mode.

B. Results and Performance Evaluation

Table II presents the accuracy and hardware evaluation results for H-WNN in comparison with FINN-R models. As observed, H-WNN achieves competitive accuracy while offering smaller parameter sizes, significantly reduced hardware resource utilization, and lower latency at the same throughput as the FINN-R models. We evaluate four configurations of H-WNNs on the CIFAR-10 dataset—Large (L), Medium (M), and two Small variants (S1 and S2). Compared to the FINN-R (1w1a) model, our smallest model (S2) achieves 1.5% higher accuracy while using 2.8× fewer BRAM18s and 1.7× more LUT6s. However, LUT6s are considerably more abundant and less expensive than BRAM18s on FPGAs, making this trade-off highly favorable for edge deployment.

The medium configuration (M) maintains the same LUT6 usage as the FINN-R CNV-6 (2w2a) model while using 2× fewer BRAM18s, with only a modest 0.9% drop in accuracy. Scaling up, our largest configuration (L) achieves accuracy comparable to the highest-performing FINN-R baseline while reducing resource usage by 1.25× in LUT6s and 1.16× in BRAM18s. All four H-WNN models deliver superior hardware efficiency compared to their FINN-R counterparts, with comparable or improved accuracy. A further thing to note is the hardware accuracy tradeoff in Table II. Comparing our model (S2) to the FINN-R 1w1a model, we observe that we achieve a higher accuracy by 1.5% by using 2.8x less BRAM18s and

TABLE III: Ablation study for the benchmarks, showcasing the impact of replacing later layers in the QNN with H-WNN architecture. Results include accuracy, hardware cost, and latency for two configurations: FINN (1w1a) + DWN and FINN (2w2a) + DWN. The study highlights the trade-offs between accuracy and hardware cost as progressively later layers in the QNN are replaced with WNN layers.

Method	**Accuracy (%)**			**LUT6s (1000s)**	**BRAM18s**	**Latency (ms)**
H-WNN: FINN (1w1a) + DWN	**KWS**	**CIFAR-10**	**SVHN**			
FC1 + FC2 + Output	80.61	84.04	94.54	17.28	175	1.92
Conv6 + ... + Output	81.52	84.62	93.72	22.87	110	1.60
Conv5 + ... + Output	80.76	83.22	92.40	21.42	81	1.28
Conv4 + ... + Output	79.94	74.00	88.44	20.14	57	0.96
Conv3 + ... + Output	78.22	65.00	87.17	17.74	42	0.64

Method	**CIFAR-10 Accuracy (%)**	**LUT6s (1000s)**	**BRAM18s**	**Latency (ms)**
H-WNN: FINN (2w2a) + DWN				
FC1 + FC2 + Output	88.23	20.72	235	6.84
Conv6 + ... + Output	87.72	23.30	137	5.7
Conv5 + ... + Output	85.71	29.43	72	4.56
Conv4 + ... + Output	74.10	20.38	41	3.42
Conv3 + ... + Output	67.93	17.87	25	2.28

with (S1) we achieve a higher accuracy by 0.41% with 1.6× less latency.

In terms of inference speed, we evaluate latency per sample for iso-accurate models. Our H-WNN (L) is 1.33× faster than the iso-accurate FINN-R CNV-6 (2w2a) model, and H-WNN (S1) achieves 1.6× lower latency compared to the FINN-R CNV-6 (1w1a), while still maintaining slightly higher accuracy in both cases. Figure 3 further highlights the benefits of H-WNN models in terms of accuracy versus LUT usage, BRAM18 consumption, and latency. Notably, for iso-accurate comparisons, H-WNN models consistently consume significantly fewer BRAM18s and exhibit lower latency compared to their FINN-R counterparts. Interestingly, no clear or consistent correlation is observed between LUT usage and accuracy.

The throughput of H-WNN models is primarily constrained by the QNN component, as the WNN layers inherently offer significantly higher throughput—requiring only 10 ns per sample. As previously observed, our H-WNN model achieves 2× lower hardware resource usage and is 1.6× faster in latency per sample, all while maintaining the same overall throughput and similar accuracy to the baseline. Minimizing energy consumption per classification task is critical in edge deployments, as it directly correlates with maximizing throughput per watt during batch processing. Since reduced hardware usage leads to proportionally lower power consumption, we estimate that the H-WNN design can reach up to approximately 2× more energy-efficient than the equivalent FINN-R implementation.

These resource savings create flexibility in deployment. The freed-up resources can either be used to unfold the QNN architecture to further improve throughput or to increase model precision and thereby enhance accuracy—depending on application-specific requirements. Furthermore, with nearly half of the hardware left unused, H-WNN allows sufficient headroom to build a complete end-to-end hardware accelerator pipeline on the Pynq-Z1 FPGA platform.

C. Ablation Study

In this section, we explore the hardware-accuracy trade-offs of selectively replacing layers of the CNV-6 QNN model with DWN layers in both 1w1a and 2w2a precision configurations. The goal of this ablation study is to identify which layers can be replaced to retain optimal accuracy while minimizing hardware resource usage. Table III summarizes the results reporting accuracy, resource usage, and latency for each configuration. To evaluate the resource benefits of DWNs, we progressively replace the later layers of the CNV-6 model with DWN components—beginning with the fully connected layers and moving backward through the convolutional layers.

The results show that substantial hardware savings can be achieved with only moderate reductions in accuracy. For instance, in the 1w1a configuration on the SVHN dataset, replacing layers from *Conv6* onward achieves accuracy comparable to the original FINN-R model, while reducing BRAM usage by 1.6×, despite a 1.3× increase in LUT consumption. As more layers are replaced, the trade-offs become more significant, highlighting the importance of balancing DWN integration with accuracy preservation. In the 2w2a configuration, the benefits of DWN replacement are even more pronounced. Starting from *Conv5* onward, H-WNN reduces BRAM usage by approximately 74% while maintaining 92.40% accuracy. This demonstrates the flexibility of our hybrid model design, enabling deployment on platforms where hardware resources are limited.

Table III shows that H-WNN models maintain high accuracy while significantly reducing hardware usage, making them a compelling choice for edge deployments under resource constraints. Moreover, our approach enables exploration across a broad design space—from high-accuracy models that require

moderate resources to highly compact models that still deliver acceptable performance. These results demonstrate that H-WNNs can be tailored to the needs of specific applications by strategically selecting layer replacement points. By leveraging DWN replacements selectively, our method paves the way for deploying compact, efficient DNNs in edge environments that cannot support the computational demands of full-scale QNN or BNN implementations.

V. Conclusion

We introduced the Hybrid Weightless Neural Network (H-WNN), a novel architecture that combines the spatial processing and feature extraction strengths of quantized convolution layers with the hardware efficiency of Weightless Neural Networks (WNNs). By analyzing the trade-off between representational depth and hardware efficiency in modern neural architectures, we proposed replacing the computationally intensive later layers with WNNs. To demonstrate this hybrid approach, we integrate the quantized model FINN-R with the state-of-the-art WNN model, Differentiable Weightless Neural Network (DWN). H-WNN achieves significant resource savings while maintaining competitive accuracy on complex image classification tasks at similar throughput. This hybrid design provides a flexible and adaptable framework that balances accuracy and efficiency, making it well-suited for energy- and resource-constrained edge environments. Further, it brings WNNs a step closer to utilizing the benefits of convolution. This work also opens avenues for extending H-WNNs to other architectures and domains beyond image classification.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback. This research was supported in part by Semiconductor Research Corporation (SRC) Task 3148.001, NSF Grants #2326894, #2425655, and NVIDIA Applied Research Accelerator Program Grant. This research has been supported by computing support on the Texas Advanced Computing Center (TACC) at the University of Texas at Austin. Any opinions, findings, conclusions, or recommendations are those of the authors and not of the funding agencies.

References

[1] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, Dec. 2018. [Online]. Available: https://doi.org/10.1145/3242897

[2] A. T. L. Bacellar, Z. Susskind, M. Breternitz Jr, E. John, L. K. John, P. M. V. Lima, and F. M. França, "Differentiable weightless neural networks," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Eds., vol. 235. PMLR, 21–27 Jul 2024, pp. 2277–2295. [Online]. Available: https://proceedings.mlr.press/v235/bacellar24a.html

[3] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," *arXiv preprint arXiv:2109.05472*, 2021.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[5] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," 2017. [Online]. Available: https://arxiv.org/abs/1705.07565

[6] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, "Accelerating convolutional networks via global & dynamic filter pruning." in *IJCAI*, vol. 2, no. 7. Stockholm, 2018, p. 8.

[7] Y.-L. Sung, V. Nair, and C. A. Raffel, "Training neural networks with fixed sparse masks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 193–24 205, 2021.

[8] W. Sun, A. Zhou, S. Stuijk, A. Nelson, R. Wijnhoven, H. Li, and H. Corporaal, "Dominosearch: find layer-wise fine-grained n:m sparse schemes from dense neural networks," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2024.

[9] R. Ma and L. Niu, "A survey of sparse-learning methods for deep neural networks," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2018, pp. 647–650.

[10] N. J. Fraser, Y. Umuroglu, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Scaling binarized neural networks on reconfigurable logic," 2017. [Online]. Available: https://arxiv.org/abs/1701.03400

[11] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: Wide reduced-precision networks," 2017. [Online]. Available: https://arxiv.org/abs/1709.01134

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[13] I. Aleksander, M. De Gregorio, F. França, P. Lima, and H. Morton, "A brief introduction to weightless neural systems," in *17th European Symposium on Artificial Neural Networks (ESANN)*, 04 2009, pp. 299–305.

[14] F. Petersen, C. Borgelt, H. Kuehne, and O. Deussen, "Deep differentiable logic gate networks," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 2006–2018.

[15] A. Khataei and K. Bazargan, "Treelut: An efficient alternative to deep neural networks for inference acceleration using gradient boosted decision trees," in *Proceedings of the 2025 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 14–24. [Online]. Available: https://doi.org/10.1145/3706628.3708877

[16] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 65–74. [Online]. Available: https://doi.org/10.1145/3020078.3021744

[17] Y. Umuroglu, Y. Akhauri, N. J. Fraser, and M. Blott, "Logicnets: Co-designed neural networks and circuits for extreme-throughput applications," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2020, pp. 291–297.

[18] M. Andronic et al., "Polylut: learning piecewise polynomials for ultra-low latency fpga lut-based inference," in *2023 International Conference on Field Programmable Technology (ICFPT)*. IEEE, 2023, pp. 60–68.

[19] ——, "Neuralut: Hiding neural network density in boolean synthesizable functions," *arXiv preprint arXiv:2403.00849*, 2024.

[20] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[21] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2017. [Online]. Available: https://arxiv.org/abs/1605.07146

[22] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[23] Z. Susskind, A. Arora, I. D. Miranda, L. A. Villon, R. F. Katopodis, L. S. De Araújo, D. L. Dutra, P. M. Lima, F. M. França, M. Breternitz, and L. K. John, "Weightless neural networks for efficient edge inference," in *31st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2022.

[24] Z. Susskind, A. Arora, I. D. S. Miranda, A. T. L. Bacellar, L. A. Q. Villon, R. F. Katopodis, L. S. de Araújo, D. L. C. Dutra, P. M. V. Lima, F. M. G. França, M. Breternitz Jr., and L. K. John, "Uleen: A novel

architecture for ultra-low-energy edge neural networks," *ACM Trans. Archit. Code Optim.*, vol. 20, no. 4, dec 2023. [Online]. Available: https://doi.org/10.1145/3629522

[25] R. Banner, I. Hubara, E. Hoffer, and D. Soudry, "Scalable methods for 8-bit training of neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1805.11046

[26] B. Chmiel, L. Ben-Uri, M. Shkolnik, E. Hoffer, R. Banner, and D. Soudry, "Neural gradients are near-lognormal: improved quantized and sparse training," 2020. [Online]. Available: https://arxiv.org/abs/2006.08173

[27] F. Faghri, I. Tabrizian, I. Markov, D. Alistarh, D. Roy, and A. Ramezani-Kebrya, "Adaptive gradient quantization for data-parallel sgd," 2020. [Online]. Available: https://arxiv.org/abs/2010.12460

[28] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," 2016. [Online]. Available: https://arxiv.org/abs/1511.06488

[29] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[31] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng *et al.*, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, no. 2. Granada, 2011, p. 4.

[32] N. Fasfous, M.-R. Vemparala, A. Frickenstein, L. Frickenstein, and W. Stechele, "Binarycop: Binary neural network-based covid-19 face-mask wear and positioning predictor on edge devices," 2021. [Online]. Available: https://arxiv.org/abs/2102.03456

[33] F. Jentzsch, Y. Umuroglu, A. Pappalardo, M. Blott, and M. Platzner, "Radioml meets finn: Enabling future rf applications with fpga streaming architectures," *IEEE Micro*, vol. 42, no. 6, pp. 125–133, 2022.

[34] J. Posso, G. Bois, and Y. Savaria, "Real-time spacecraft pose estimation using mixed-precision quantized neural network on cots reconfigurable mpsoc," 2024. [Online]. Available: https://arxiv.org/abs/2407.06170

[35] M. Ghasemzadeh, M. Samragh, and F. Koushanfar, "Resbinnet: Residual binary neural network," *CoRR*, vol. abs/1711.01243, 2017. [Online]. Available: http://arxiv.org/abs/1711.01243

[36] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[37] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *ArXiv*, vol. abs/1708.07747, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:702279

[38] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly, P. Montino, D. Kanter, S. Ahmed, D. Pau *et al.*, "Mlperf tiny benchmark," *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[39] A. Benamira, T. Guérand, T. Peyrin, T. Yap, and B. Hooi, "A scalable, interpretable, verifiable differentiable logic gate convolutional neural network architecture from truth tables," 2023. [Online]. Available: https://arxiv.org/abs/2208.08609

[40] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016. [Online]. Available: https://arxiv.org/abs/1602.02830

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[42] H. Carneiro, F. França, and P. Lima, "Multilingual part-of-speech tagging with weightless neural networks," *Neural Networks*, vol. 66, 03 2015.

[43] S. Niu, Y. Liu, J. Wang, and H. Song, "A decade survey of transfer learning (2010–2020)," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 2, pp. 151–166, 2020.

[44] Xilinx, "Finn: A framework for fast and efficient quantized neural networks," n.d., accessed: 2024-11-14. [Online]. Available: https://xilinx.github.io/finn/

[45] ——, "Brevitas: Quantization-aware training in pytorch," https://github.com/Xilinx/brevitas, 2021, accessed: 2025-03-31.

[46] X. Inc., "Zynq-7000 all programmable soc data sheet: Overview," 2017. [Online]. Available: https://docs.amd.com/v/u/en-US/ds190-Zynq-7000-Overview