FPGA-Powered Environment Awareness via Quantized Neural Networks for LiDAR-Aided mm-Wave Beam Prediction

Arish Sateesan, Ljiljana Simić
Institute for Networked Systems, RWTH Aachen University, Aachen, Germany
{asa, lsi}@inets.rwth-aachen.de

***Abstract*—Environment awareness can be highly beneficial for robust and agile beam prediction, particularly for beyond-5G millimeter-wave (mm-wave) networks. While machine learning (ML) algorithms have shown potential in leveraging external sensor data, such as LiDAR, radar, and cameras, to enhance beam prediction, existing solutions often rely on simulations or offline data processing, limiting their applicability in real-world mm-wave deployments. Real-world deployment of such solutions requires low-latency ML inference tailored to resource-constrained hardware platforms but remains challenging due to the computational complexity and latency demands of ML models. This paper addresses these challenges by implementing real-time ML inference in hardware focusing on environment awareness using LiDAR data. We present an FPGA-based implementation of Quantized Neural Networks (QNNs) optimized for real-time LiDAR-aided beam prediction in beyond-5G mmWave networks. Evaluations on the ZCU104 FPGA platform using real-world datasets demonstrate inference latencies in the tens to hundreds of microseconds, achieving comparable accuracy to state-of-the-art methods with only 2-bit weights and activations. Our results underline the effectiveness of the QNNs in achieving high accuracy, low latency, and hardware resource efficiency for real-world mmWave applications.**

I. Introduction

Millimeter-wave (mm-wave) bands offer wide channels supporting high data rates and low latency, making them attractive for next-generation wireless communications. However, mm-wave signals face significant challenges like susceptibility to blockage, high propagation losses, and highly site-specific coverage. The use of directional beams with high beamforming gain can counteract these challenges but requires efficient and complex dynamic beam management to maintain stable connectivity under mobility, especially in dense urban environments. Traditional beam selection techniques, such as the exhaustive search beam sweep employed in standards like IEEE 802.11ad and 5G-NR, suffer from resource and time overheads, limiting their real-time performance. For instance, 5G-NR employs exhaustive beam search with synchronization signal (SS) bursts of 5 ms, repeating every 20 ms, resulting in beamforming delays of 300–900 ms [1].

In the context of directional mm-wave networks, effective beam management directly depends on environmental factors such as base station (BS)/user equipment (UE) positions relative to the geometry of the urban building layout and moving obstacles such as cars or pedestrians that may block or reflect the mm-wave signal. Explicit environment awareness is thus valuable for achieving robust, flexible, and efficient radio resource management (RRM) for beyond-5G networks. Machine learning (ML) techniques have shown significant potential in enhancing beam prediction and beamforming by leveraging real-time and historical environmental data from non-RF sensors such as LiDAR (Light Detection and Ranging), GPS (Global Positioning System), and cameras [2]–[4]. The predictive capability of ML algorithms enables real-time perception of the surrounding environment. This data-driven approach reduces system complexity and beam training overhead while greatly reducing direct radio measurement signalling overhead by utilizing side information from environmental sensing [5].

Despite the advances in ML-based environment awareness, most studies demonstrate performance primarily through simulations using synthetic data or offline processing of real-world measurements. Deploying ML techniques in real-world beyond-5G networks requires low-latency, real-time inference on hardware, which is often challenged by the computational complexity, latency sensitivity, and energy constraints inherent in mobile network systems. Our work addresses these challenges by targeting efficient real-time QNN inference for environment-aware beam prediction in beyond-5G networks.

Beam prediction in mm-wave networks: Recent research has explored combining ML with out-of-band beam prediction approaches [6] using non-RF sensors such as cameras [7], LiDAR [8]–[10], and multi-modal non-RF sensing [3]. LiDAR, in particular, offers better depth perception and object localization than cameras and superior spatial resolution compared to radar, making it highly suitable for environment mapping. Multi-modal approaches enhance accuracy, but with increased computational and hardware demands. Existing LiDAR-based studies have addressed the scenarios in both outdoor [2], [8], [9] and indoor beam prediction [10]–[12]. However, most of the existing, limited, LiDAR-based approaches for beam prediction are evaluated with simulated datasets [11], [12] or offline post-processing of measurement data [2], [3], with limited hardware implementation, making it difficult to assess their real-world efficiency and real-time performance.

Quantized Neural Networks: In practical cellular networks, both the BS and especially the UE typically have memory and computational constraints, demanding efficient processing solutions. Quantization techniques in neural networks address these challenges by reducing the computational and memory requirements [13]. Binarized Neural Networks (BNNs), for instance, constrain weights and activations to {-1, 1}, thereby minimizing memory and computational requirements with minimal accuracy trade-offs [14]. Even low-bit-width quantization reduces computational demands while preserving accuracy [15]. Lower bit-width representations enable models to fit on fast, on-chip memory, reducing off-chip memory accesses and power consumption.

In this paper, we bridge the gap between theoretical advancements in ML-based beam prediction and practical implementation by focusing on real-time ML inference for environment awareness in mm-wave networks. We present an efficient FPGA-based implementation of QNNs, leveraging low-bit-width models optimized for real-time beam prediction. We note that, although GPUs generally excel in ML tasks, their high power consumption and dependence on auxiliary components limit their applicability. In contrast, FPGAs are energy-efficient and standalone, making them ideal for deploying ML algorithms in resource-constrained environments [16]. Moreover, existing next-generation WiFi and beyond-5G experimental testbeds are based on RFSoC FPGAs [17] and FPGA-SDR platforms [18], enabling seamless integration of FPGA-based implementation of QNNs into wireless state-of-the-art experimental platforms.

To tackle the challenges in emerging ML-based environment awareness for beyond-5G networks, we propose a LiDAR-based beam prediction model using QNN, trained on real-world datasets. Our model achieves a Top-1 accuracy of 85% on the FLASH [19] dataset, surpassing the full-precision model with a 2.6× reduction in model size. We also analyze the effects of quantization on accuracy by varying the spatial characteristics of the datasets. We deploy the QNN model on the Zynq Ultrascale+ ZCU104 FPGA using the Xilinx FINN framework [13], which achieves inference speeds 100-1000 times faster than the data capture rate of LiDAR, showing its applicability in real-time, real-world scenarios. Our results show that combining FPGAs with QNNs lowers computational and memory demands while maintaining predictive accuracy.

II. LiDAR and ML-based Beam Prediction

We propose and implement an ML-based beam prediction model on FPGA leveraging LiDAR data for environment awareness to predict the optimal beam in mm-wave networks. The subsequent sections detail the input data preprocessing, model architecture, and the system model.

A. System Model for ML-based Beam Prediction

The system model for the ML-based beam prediction is shown in Fig. 1. It consists of a LiDAR preprocessing module and QNN, where QNN performs the feature extraction and classification. The input to the beam prediction model is the LiDAR point cloud P. Based on this input, the model predicts the best beam $b_o \in \mathcal{B}$ at the BS, where $\mathcal{B} = \{b_1, b_2, ..., b_M\}$ is the set of all possible beams, with M denoting the total number of beams. Each beam b_j has a specific RSS value, where $j \in \{1, 2, \ldots, M\}$, and the optimal beam b^* is the one with the maximum RSS value. In the ideal scenario, the predicted beam

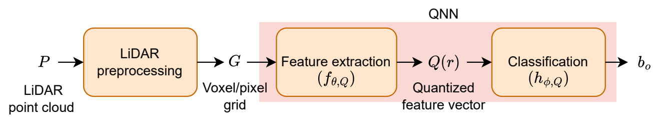


Fig. 1. System model for ML-based beam prediction.

b_o is the optimal beam b^*. Each beam b_j also corresponds to a specific antenna beam steering direction and respective coverage area. The preprocessing module converts the LiDAR point cloud into a feature map, represented as a grid $G \in \mathbb{R}^{H\times W\times D}$, as elaborated in Section II-B. We use our proposed SVQNN model (cf. Section II-C) for feature extraction, represented as $f_{\theta,Q}$, configured for quantized operations, with parameters θ. The SVQNN processes the quantized LiDAR feature map G to generate a quantized feature vector $Q(\mathbf{r}) \in \mathbb{R}^d_{2^k}$, where d is the feature vector size and k is the quantization bit-width if the grid r is populated on value-based method and is quantized to k-bit integer values. For an occupancy-based grid (cf. Section II-B), the value of k is 1. The quantized feature vector is expressed as:

$$Q(\mathbf{r}) = f_{\theta,Q}(G)$$

Here, all weights, activations, and outputs in $f_{\theta,Q}$ are quantized. The quantized feature vector $Q(\mathbf{z})$ is input to a quantized classifier (dense layer), $h_{\phi,Q}$, with parameters ϕ. The predicted optimal beam b_o is selected using the classifier, as it outputs a quantized probability distribution over the M beams. During training, cross-entropy loss is used between the predicted quantized beam probabilities $h_{\phi,Q}(Q(\mathbf{z}))$ and the true beam label. The predicted beam is determined as:

$$b_o = \arg\max_{b_j \in \mathcal{B}} \; h_{\phi,Q}(Q(\mathbf{r}))[j]$$

We evaluate the system model based on the Top-k accuracy metric. It measures the proportion of instances where the optimal beam b^* is among the top k predicted beams.

B. LiDAR Data & Preprocessing

The LiDAR sensor generates a spatial point cloud, $P = \{(x_i, y_i, z_i)\}_{i=1}^{|N|}$, where (x_i, y_i, z_i) are the 3D coordinates of the i_{th} point mapping to the detected object points around the sensor and N is the total number of points. Given the complexity of these point clouds, preprocessing is essential to reduce data dimensionality and noise, ensuring structuring and scaling of relevant features are suitable for the deep learning model. The point cloud is transformed into either a 3D voxel or a 2D pixel grid, denoted by G. A voxel grid is represented as $G \in \mathbb{R}^{H\times W\times D}$ and a pixel grid is represented as $G \in \mathbb{R}^{H\times W}$, where H and W represents the 2D spatial plane and D represents the number of layers.

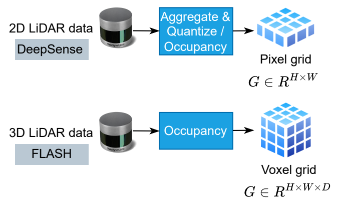


Fig. 2. LiDAR data preprocessing.

A voxel grid represents spatial information, dividing space into discrete units called voxels, while a pixel grid is its 2D counterpart. Grids are generated either by occupancy or value-based methods. An occupancy grid is populated with binary values and is marked as '1' (occupied) if it contains any data points, while value-based grids aggregate and quantize data points. In this paper, we use two datasets, DeepSense [20] and FLASH [19]. The FLASH dataset is preprocessed into a 3D occupancy voxel grid, while DeepSense data is transformed into a 2D pixel grid, based on either occupancy or value-based methods, as shown in Fig. 2. The details of these datasets are elaborated in Section III-A.

C. Deep Learning Model

The neural network model we adopt is a quantized adaptation of multi-view convolutional neural network (MVCNN), originally designed for 3D shape recognition [21]. The MVCNN architecture, illustrated in Fig. 3(a), comprises multiple view-CNNs for feature extraction from individual sensors, a view pooling layer for feature aggregation, and a final CNN for classification. Each view-CNN includes three 2D convolutional layers with a kernel size of 3×3, followed by a fully connected (FC) layer. In this work, we modified the model to adapt to using only a single view with LiDAR as the descriptor. The MVCNN framework can also integrate multi-sensor data (e.g. camera views) for multi-modal sensing. Since we only use one view, we omit the view pooling and concatenation and replace the final CNN with an FC layer with 64 output channels as classifier, followed by quantization of layers. Each quantized convolutional layer is followed by batch normalization, quantized max-pooling (kernel size 2×2, stride 2), and a quantized ReLU/HardTanh activation. The quantized model for a single descriptor is called single-view QNN (SVQNN), and is shown in Fig. 3(b). The input to the model is the LiDAR voxel or pixel grid, where the depth of the grid D serves as the input channel dimension for the first 2D convolution layer.

D. Training & Inference on FPGA

Training: We use quantization-aware training (QAT) to train the neural network, a technique that simulates the effects of reduced numerical precision during the training process. Unlike post-training quantization, which applies quantization to trained parameters after the model is fully trained, QAT allows the model to learn to adapt to quantization effects, such as reduced precision in weights and activations, while being trained. This approach helps mitigate any potential loss in accuracy. The input to the model, a voxel/pixel grid of a LiDAR point cloud, tends to be sparse, with each value carrying critical information. Applying binarization to such a sparse input risks loss in information and accuracy. This prompts us to employ quantized weights and activations instead of pure binarization during training. The model is trained with Brevitas [22], a library designed for QAT, built on top of PyTorch. In Brevitas, *fake quantization* layers (e.g., convolution, max-pooling, ReLU, HardTanh) are applied during training to simulate quantization effects. These layers apply quantization only during forward passes in training and retain higher precision weights in the model graph. During inference, weights, activations, and outputs are quantized to their specified bit-widths, ensuring consistent performance in a quantized environment.

Inference on FPGA: We employ the Xilinx FINN-R framework [13] to map the trained model to hardware. The hardware architecture of each layer in the FINN-based QNN inference model is shown in Fig. 4. The fundamental computational components of the hardware architecture are the matrix-vector threshold unit (MVU) and the sliding window unit (SWU). An array of parallel processing elements (PE) with each PE performing parallel computations based on the number of single-instruction-multiple-data (SIMD) lanes forms the MVU. The number of PEs and SIMD lanes determines the computational parallelism of the hardware architecture. The PE is responsible for multiplication, accumulation, and threshold comparison operations in convolutional and FC layers. Both convolution and max pooling layers employ the sliding window unit (SWU) to organise input data into overlapping or non-overlapping patches, enabling the MVU to execute matrix-matrix multiplications and comparison operations for convolution and pooling operations, respectively. FINN has limited support for non-linear activations like Tanh and Sigmoid, as it is optimized for linear functions or simpler threshold-based activations such as ReLU or HardTanh. Thus, we adopt the quantized HardTanh activation with the value range $\{-1, +1\}$ as the activation function for approximating non-linear behaviour. HardTanh works well with low-bit activations due to its bounded output range and is hardware-friendly for binarized parameters, compared to ReLU [23].

III. Performance Evaluation

A. Experimental Setup

Dataset: Training a neural network for environment-aware beam prediction requires a dataset that features the environmental conditions relevant to beamforming, such as user location, channel state information in terms of RSS per beam, mobility patterns, and obstacles that affect signal propagation. We use two real-world datasets, *DeepSense* [20] (scenario 8)

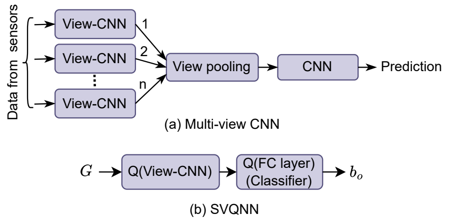


Fig. 3. Multi-View CNN and SVQNN models.

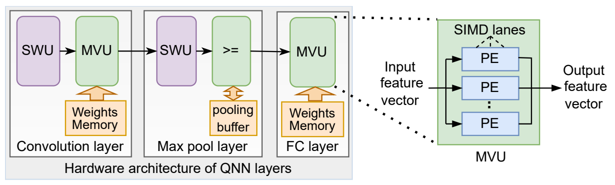


Fig. 4. Hardware architecture of QNN layers.

and *FLASH* [19], to train and evaluate the proposed beam prediction model. In the FLASH data collection setup, the moving UE is equipped with LiDAR and GPS connected to an onboard computer, with data capture rates of 10 Hz for LiDAR and 1-1.5 Hz for RSS beam sweep measurements. IEEE 802.11ad Wi-Fi radios, equipped with antenna arrays operating at a 60 GHz[1]frequency, are used as both BS and UE, with the UE being omnidirectional. The default codebook includes 34 defined sectors (M=34), sector IDs 1-31 and 61-63, while IDs 32-60 remain undefined. In the DeepSense setup, the BS features a LiDAR, GPS receiver, and 16-element phased array receiver. The moving UE has GPS and mm-wave omni-directional transmitter. Both BS and UE operate at 60 GHz. The signal is transmitted using a codebook of 64 pre-defined beams (M=64). The data capture rate of LiDAR is 10 Hz and of the RSS beam sweep measurements is 8-9 Hz. The collected LiDAR, GPS, and RF data are synchronized during preprocessing. FLASH and DeepSense datasets contains $\sim$32K and $\sim$4K samples, respectively. The dataset is split into 80% for training, and 10% each for validation and testing. Notably, the FLASH dataset contains 3D LiDAR point clouds, while the DeepSense dataset provides 2D LiDAR measurements, making them distinct in dimensionality and information richness. Each dataset is labelled with the optimal beam index b^* based on the received signal strength (RSS).

Hardware platform: Our work aims towards the integration of ML-aided beam prediction into FPGA and SDR-based experimental platforms and testbeds [17], [18]. For wireless network applications, the preferred FPGA platforms are RF-SoC FPGAs, such as the Zynq Ultrascale+ ZCU208 evaluation platform [24]. However, since FINN does not directly support deploying neural networks into RFSoC FPGAs, we manually integrate the hardware-mapped QNN generated on a FINN-supported platform into the RFSoC programmable logic. We choose the FINN-supported Zynq UltraScale+ ZCU104 MP-SoC (xczu7ev-ffvc1156-2-e) evaluation board [25] for this work due to its architectural similarity with Zynq UltraScale+ RFSoC devices. The QNN is mapped to this platform generating a hardware IP (*stitched IP*), that integrates into the ZCU208 RFSoC via AXI stream FIFO interfaces.

Input and output: For the FLASH dataset, the LiDAR point clouds are preprocessed into an occupancy voxel grid of size $(20 \times 20 \times 20)$. For the DeepSense dataset, the 2D LiDAR data is transformed into an occupancy or value-based pixel grid of size (20×20). These representations serve as inputs, with only the input channels of the first convolutional layer of the QNN differ between the datasets. The model outputs a single predicted beam index from the M available classes.

B. Architectural Exploration

For evaluation, we analyze two variations of the SVQNN (cf. Section II-C), namely SVQNN$_S$ and SVQNN$_L$, which differ in the number of channels per layer. SVQNN$_S$ is the standard model with 128 channels in the convolution layers and 512 channels in the FC layer. SVQNN$_L$ is the lightweight version with reduced hardware requirements by scaling down the number of channels in each layer, featuring 16 channels in the convolution layers and 128 channels in the FC layer. For weights and activation bit-widths $\leq$2, the activation function used is HardTanh. Both networks are trained with quantized weights and activations, with bit-widths varied between $w \in \{2, 8\}$ for weights and $a \in \{1, 8\}$ for activations to assess performance under quantization. Training spans 75 epochs using the cross-entropy loss function, the ADAM optimizer, and a learning rate of 0.001. The out-of-context synthesis on hardware is performed for a target clock frequency of 100 MHz. The effects of parallelization on hardware resource usage, throughput, and latency are evaluated by varying folding parameters such as the number of PEs, SIMD lanes, and MVU width. The MVU width determines the degree of parallelism in vector operations, defined by the number of PEs, SIMD lanes, and multi-vector parallelization. Folding reduces the parallelism by distributing computations over multiple clock cycles, enabling neural network models to fit within FPGA resource constraints. In FINN, folding can be automated by adjusting the MVU width or manually tuned by modifying PE and SIMD values for each layer.

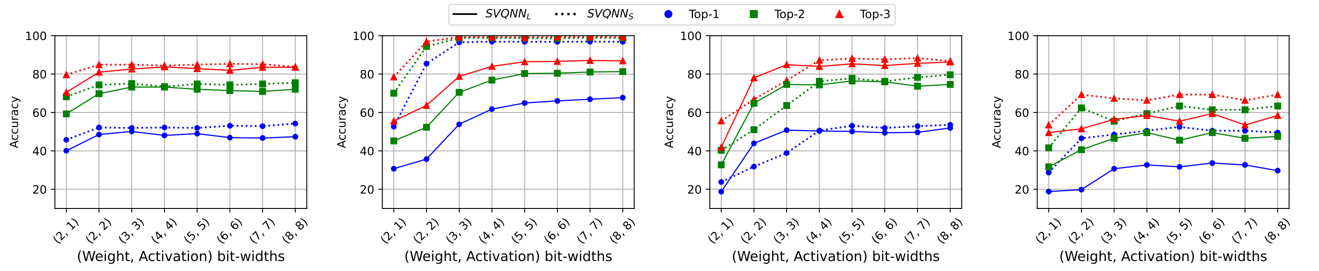
C. Results

We present the results evaluating the proposed model using the DeepSense and FLASH datasets, focusing on key metrics, including accuracy, resource usage, and latency.

Accuracy and the impact of dataset characteristics: We evaluate the effect of quantization on accuracy by varying w and a bit-widths, using Top-k beam prediction accuracy as the primary metric. Fig. 5 shows the accuracy vs. $\{w, a\}$ bit-width configurations for the DeepSense and FLASH datasets with different input quantizations and SVQNN configurations. Fig. 5(a) shows that the DeepSense dataset exhibits minimal sensitivity to different QNN models and varying quantization values, while the FLASH dataset, as shown in Fig. 5(b), shows greater dependence to the QNN variations. This is due to the differences in size, dimensionality, and complexity of the datasets: the FLASH dataset, with a large sample size and 3D LiDAR data, contains richer spatial information and higher redundancy, while the DeepSense dataset has lower complexity and fewer features, which may lead to overfitting, thereby masking the effects of quantization. This difference highlights the greater sensitivity of the FLASH dataset to quantization errors compared to the simpler DeepSense dataset.

To further analyze the impact of dataset characteristics, we tested the models on the quantized version of the DeepSense dataset and the category-1 (Cat-1) sub-dataset of FLASH. The Cat-1 FLASH dataset includes line-of-sight (LOS) passing scenario and has a sample size of $\sim$9.7K. The quantized DeepSense dataset, containing 16-bit quantized data points, carries more detailed information and is more sensitive to quantization compared to its occupancy-based counterpart as evident from Figs. 5(a) and 5(c), respectively. The Cat-1

[1]We note that we use these mm-wave datasets at 60 GHz without loss of generality due to the lack of datasets at 28 GHz corresponding to 5G and beyond FR2 deployments.



(a) DeepSense: Occupancy-based dataset (b) FLASH: Occupancy-based dataset (c) DeepSense: Quantized dataset (d) FLASH: Cat-1 occupancy-based dataset

Fig. 5. Top-k beam prediction vs. weight and activation bit-widths for the SVQNN$_S$ and SVQNN$_L$ models and different datasets.

FLASH dataset remains highly sensitive to quantization due to its data complexity as shown in Fig. 5(d), but with a significant reduction in accuracy due to the reduced sample size compared to the full FLASH dataset in Fig. 5 (b).

Table I compares the Top-1 accuracy, number of parameters, and inference latency as measured in our FPGA implementation across SVQNN models and benchmarks like Omni-CNN [3] and Jiang et al. [2]. The results of SVQNN models are generated with an MVU width of 32 for the DeepSense dataset, and MVU widths of 128 and 32 for SVQNN$_L$ and SVQNN$_S$, respectively, on the FLASH dataset. Although a direct comparison with the existing models from the literature is not possible as the implementation platform and the CNN models differ, Table I shows that our quantized SVQNN versions demonstrate comparable accuracy to the existing full-precision models for LiDAR-based beam prediction while achieving a significantly smaller memory footprint and excellent inference latency of tens to hundreds of microseconds. The SVQNN$_L$ model for the DeepSense dataset has $\sim$29$\times$ smaller model size and $\sim$49$\times$ lower inference latency than [2], with only 0.14$\times$ loss in accuracy. SVQNN$_S$ attains top-3 accuracies of 79% with only 2-bit weights and 1-bit activations, and 97% with 2-bit weights and activations on the FLASH dataset. It also achieves a top-1 accuracy of 85.43% with 2-bit weights and activations, outperforming Omni-CNN in accuracy while maintaining a smaller model size, but with increased inference latency. For reference, Omni-CNN is reported in [3] to achieve 0.008 ms latency, but presumably on a GPU. We note that the BS and UE radio sweeping delay of 5 ms and 20 ms for a single beam in 5G-NR [1], a beam prediction inference latency of under 1 ms likely meets system-level latency requirements. Importantly, our SVQNN models on FPGA offer better trade-offs in latency, energy efficiency, and resource utilization in contrast to GPU-based implementations which have lower energy efficiency and limited customizability.

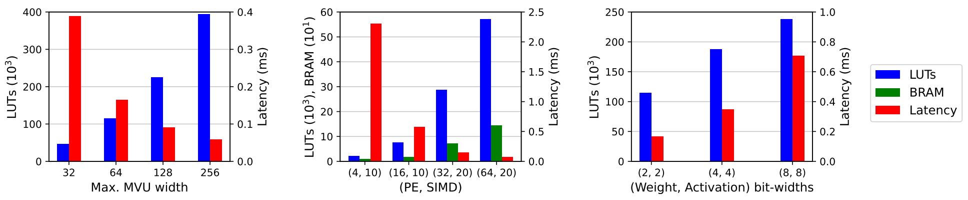
Resource usage, latency, and throughput: The synthesis results on the ZCU104 MPSoC FPGA platform for the SVQNN models are presented in Table I, showing good trade-offs between resource utilization, latency, and throughput. The SVQNN$_S$ model for the FLASH dataset outperforms Omni-CNN in accuracy, requiring only $\sim$20% of the LUTs and $\sim$27% of the on-chip memory available. The lightweight models require only $<$1% of the on-chip block RAM (BRAM) while achieving a latency of less than 0.1 ms, showcasing the benefits of quantization. Although ZCU104 MPSoC platform's constrained resources limit the parallelization potential of complex models like SVQNN$_S$, thus capping its throughput, it far exceeds the sample rate of LiDAR in the order of 10-30 fps [19], [20], ensuring real-time capability for beam prediction and LiDAR data processing. Table I shows that the lightweight SVQNN models achieve high throughput with low resource usage, demonstrating a clear trade-off between hardware efficiency and accuracy. The bit-width of QNN weights and activations also influences the resource usage and latency due to the increased complexity of the multiply-accumulate (MAC) operations, as shown in Fig. 6(c).

Impact of parallelization: The interrelation between parallelization and resource utilization is shown in Figs. 6(a) and 6(b), based on the FINN estimation results on FPGA of the FLASH SVQNN$_S$ model. The resource-constrained ZCU104 platform imposes limits on the degree of parallelism that can be applied, leading us to rely on FINN estimation results. Increasing the MVU width or the number of PEs and SIMD lanes reduces latency but increases resource requirements as shown in Figs. 6(a) and 6(b), underscoring the trade-off between speed and resource efficiency. Fig. 6(b) shows the effect of manual folding only for the first convolutional layer. Furthermore, FINN's architectural approach associates preloaded weight memories with each PE, elevating the BRAM usage, as shown in 6(b). Larger memory blocks would also introduce increased routing delays, thereby reducing throughput.

TABLE I
Performance, hardware results, and comparison of the QNN-LiDAR model.

Model	Top-1 accuracy (%)	Number of parameters	Inference latency(ms)	Model size (KB)	LUT utilization	BRAM utilization	Throughput fps	Hardware platform
DeepSense Dataset								
SVQNN$_L$ (2, 2)[1]	48.51	21,136	0.028	5.3	4,804 (2.1%)	1 (0.3%)	305761	Zynq Ultrascale+
SVQNN$_S$ (2, 2)	52.17	590,976	0.151	144.3	21412 (9.3%)	69 (22.1%)	22347	Zynq Ultrascale+
Jiang et al. [2]	57.51	$\sim$40,000	1.38	156.3	–	–	–	Intel Xeon Silver 4216
FLASH Dataset								
SVQNN$_L$ (4, 4)	61.76	23,872	0.09	11.7	19710 (8.5%)	2 (0.6%)	55246	Zynq Ultrascale+
SVQNN$_S$ (2, 2)	85.43	612,864	0.179	149.6	45806 (19.9%)	84 (26.9%)	22904	Zynq Ultrascale+
Omni-CNN [3]	82.68	$\sim$100,000	$\sim$0.008	390.6*	–	–	–	Not disclosed
FLASH [26]	68.17	$\sim$690,000	0.6	2695.3*	–	–	–	Not disclosed

[1](Weight, activation) bit-widths; *Assuming 32-bit precision



(a) LUT, Latency vs. MVU width (b) LUT, Latency, BRAM vs. PE, SIMD - CONV_1 (c) LUT, Latency vs. W, A

Fig. 6. Effect of parallelization and bit-width on resource usage and latency for the FLASH dataset and SVQNN$_S$ model.

Therefore, careful tuning of folding parameters is essential to balance the performance and hardware efficiency. Higher-end platforms like the ZCU208 RFSoC [24] support deeper parallelism than the ZCU104, offering better throughput and lower latency for larger models. Our ongoing work focuses on porting our QNN models to the ZCU208 platform.

IV. Conclusions

This paper addressed the critical need for real-time, environment-aware beam prediction in beyond-5G networks by leveraging the synergy of LiDAR data and QNNs. Our FPGA-based implementation of LiDAR-QNN model demonstrates the feasibility of deploying low bit-width QNNs for beam prediction in mm-wave networks, balancing accuracy, latency, and hardware efficiency. Using the DeepSense and FLASH datasets, we validated our model's performance in dynamic real-world environments, achieving inference latencies in the range of tens to hundreds of microseconds, 100 to 1000 times faster than the LiDAR data rate. This showcases the model's real-time inference capabilities and its applicability to beyond-5G scenarios.

Acknowledgment

This work has received funding by the German BMBF in the course of the 6GEM research hub, grant number 16KISK038.

References

[1] C. N. Barati, S. Dutta, S. Rangan, and A. Sabharwal, "Energy and latency of beamforming architectures for initial access in mmWave wireless networks," *Journal of the Indian Institute of Science*, vol. 100, pp. 281–302, 2020.

[2] S. Jiang, G. Charan, and A. Alkhateeb, "LiDAR aided future beam prediction in real-world millimeter wave V2I communications," *IEEE Wireless Communications Letters*, vol. 12, no. 2, pp. 212–216, 2022.

[3] B. Salehi, D. Roy, T. Jian, C. Dick, S. Ioannidis, and K. Chowdhury, "Omni-cnn: A modality-agnostic neural network for mmwave beam selection," *IEEE Transactions on Vehicular Technology*, 2024.

[4] D. d. S. Brilhante, J. C. Manjarres, R. Moreira, L. de Oliveira Veiga, J. F. de Rezende, F. Müller, A. Klautau, L. Leonel Mendes, and F. A. P. de Figueiredo, "A literature survey on AI-aided beamforming and beam management for 5G and 6G systems," *Sensors*, vol. 23, 2023.

[5] Q. Xue, C. Ji, S. Ma, J. Guo, Y. Xu, Q. Chen, and W. Zhang, "A survey of beam management for mmWave and THz communications towards 6G," *IEEE Communications Surveys & Tutorials*, 2024.

[6] D. Roy, B. Salehi, S. Banou, S. Mohanti, G. Reus-Muns, M. Belgiovine, P. Ganesh, C. Dick, and K. Chowdhury, "Going beyond RF: A survey on how AI-enabled multimodal beamforming will shape the NextG standard," *Computer Networks*, vol. 228, p. 109729, 2023.

[7] M. Alrabeiah, A. Hredzak, and A. Alkhateeb, "Millimeter wave base stations with cameras: Vision-aided beam and blockage prediction," in *proceedings of the 91st vehicular technology conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–5.

[8] A. Klautau, N. González-Prelcic, and R. W. Heath, "LIDAR data for deep learning-based mmWave beam-selection," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 909–912, 2019.

[9] M. Zecchin, M. B. Mashhadi, M. Jankowski, D. Gündüz, M. Kountouris, and D. Gesbert, "LIDAR and position-aided mmWave beam selection with non-local CNNs and curriculum training," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 2979–2990, 2022.

[10] T. Woodford, X. Zhang, E. Chai, K. Sundaresan, and A. Khojastepour, "Spacebeam: Lidar-driven one-shot mmwave beam management," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 389–401.

[11] D. Marasinghe, N. Jayaweera, N. Rajatheva, S. Hakola, T. Koskela, O. Tervo, J. Karjalainen, E. Tiirola, and J. Hulkkonen, "LiDAR aided Wireless Networks - Beam Prediction for 5G," in *proceedings of the 96th Vehicular Technology Conference (VTC2022-Fall)*, 2022, pp. 1–7.

[12] N. Jayaweera, D. Marasinghe, N. Rajatheva, S. Hakola, T. Koskela, O. Tervo, J. Karjalainen, E. Tiirola, and J. Hulkkonen, "LiDAR aided Wireless Networks-LoS Detection and Prediction based on Static Maps," in *proceedings of the 96th Vehicular Technology Conference (VTC2022-Fall)*. IEEE, 2022, pp. 1–6.

[13] 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 Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.

[14] 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, 2016.

[15] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *arXiv preprint arXiv:1511.06488*, 2015.

[16] A. Sateesan, S. Sinha, S. KG, and A. Vinod, "A survey of algorithmic and hardware optimization techniques for vision convolutional neural networks on FPGAs," *Neural Processing Letters*, vol. 53, no. 3, pp. 2331–2377, 2021.

[17] J. O. Lacruz, R. R. Ortiz, and J. Widmer, "A real-time experimentation platform for sub-6 GHz and millimeter-wave MIMO systems," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 427–439.

[18] Schott, Aron and Ichkov, Aleksandar and Acikgöz, Berk and Beckmann, Niklas and Simić, Ljiljana, "A Multi-Band mm-Wave Experimental Platform Towards Environment-Aware Beam Management in the Beyond-5G Era," in *ACM WiNTECH*. ACM, 2024.

[19] J. Gu, B. Salehi, D. Roy, and K. Chowdhury, "FLASH Dataset," https://repository.library.northeastern.edu/files/neu:k930bx12m, accessed: 2024.

[20] A. Alkhateeb, G. Charan, T. Osman, A. Hredzak, J. Morais, U. Demirhan, and N. Srinivas, "DeepSense 6G: A large-scale real-world multi-modal sensing and communication dataset," *IEEE Communications Magazine*, vol. 61, no. 9, pp. 122–128, 2023.

[21] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[22] A. Pappalardo, "Xilinx/brevitas," https://doi.org/10.5281/zenodo.3333552, 2023.

[23] K. Abdelouahab, M. Pelcat, and F. Berry, "Phd forum: Why tanh can be a hardware friendly activation function for cnns," in *Proceedings of the 11th International Conference on Distributed Smart Cameras*, 2017.

[24] AMD, "Zynq UltraScale+ RFSoC ZCU208 Evaluation Kit," https://www.xilinx.com/products/boards-and-kits/zcu208.html, accessed: 2024.

[25] AMD, "Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit," https://www.xilinx.com/products/boards-and-kits/zcu104.html, accessed: 2024.

[26] B. Salehi, J. Gu, D. Roy, and K. Chowdhury, "Flash: Federated learning for automated selection of high-band mmwave sectors," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, 2022.