

Quantized CNV Equalization for MIMO Channels with Memory on a Zynq FPGA

Jeison Magallanes Poma, Jonathan Aguiar Soares, Icaro Freitas, Christian Esteve Rothenberg, Dalton S. Arantes, and Marco Roberto Cavallari

Abstract—Embedded machine learning enables latency-sensitive signal processing at the network edge, but neural receivers on small FPGA platforms require careful quantization and hardware co-design. This paper presents a quantized convolutional neural network (CNN) equalizer for a real-valued 2×2 BPSK MIMO channel with finite memory, implemented on a Digilent Zybo-Z7-20 Zynq-7020 board using Brevitas, QONNX, FINN, Vivado and PYNQ. In this work, CNV denotes a FINN-compatible quantized CNN topology used for FPGA-oriented dataflow synthesis. We evaluate input, weight and activation bit widths, compare the selected hardware design against memory-aware ZF and LMMSE references, and analyze the effect of increasing the channel-memory span. The selected compact design reaches 96.19% hardware vector accuracy at 12 dB and approximately 1.32 Msamples/s. The experiments use pre-generated data and a fixed channel realization; therefore, the results represent an embedded FPGA proof of concept rather than validation of cross-channel or over-the-air robustness.

Keywords—Edge machine learning, MIMO equalization, quantized neural networks, FINN, FPGA, PYNQ.

I. INTRODUCTION

Machine learning workloads have traditionally scaled through centralized servers and cloud data centers. More recently, programmable logic, embedded processors and neural-network toolchains at the network edge have made it possible to move computation closer to the signal source. This is relevant for latency-sensitive digital signal processing (DSP), where sending raw samples to remote servers increases delay, bandwidth use and dependence on backhaul connectivity. Edge acceleration also supports distributed computing, allowing local nodes to reduce latency and contribute to aggregate network compute capacity [1], [2].

Wireless receivers are a representative case for this trend. In multiple-input multiple-output (MIMO) systems, the receiver must recover transmitted symbols from antenna-domain mixtures affected by noise, coupling and inter-symbol interference. Classical equalizers and detectors, such as zero-forcing, MMSE and sphere decoding, provide strong analytical

baselines [3]. However, neural receivers are attractive when the channel is difficult to invert explicitly or when the detector must be mapped to a fixed-throughput hardware datapath.

This work presents an end-to-end hardware co-design of a quantized CNV equalizer for a finite-memory MIMO channel, rather than a new detection algorithm. Labeled MIMO/BPSK sequences are generated offline, while Brevitas, QONNX and FINN are used to train, compile and deploy the accelerator on a Zybo-Z7-20. The contributions are: (i) a complete dataset-to-hardware flow; (ii) separate input, weight and activation quantization analysis; (iii) physical-board accuracy, throughput and post-synthesis resource measurements; (iv) comparison with memory-aware ZF and LMMSE; and (v) evaluation across increasing channel-memory spans.

II. RELATED WORK

Learning-based MIMO detection has evolved from fully data-driven neural detectors to model-driven and hybrid receivers. DetNet showed that deep networks can be trained to approximate MIMO detection maps directly [4]. Later work argued that physical-layer receivers benefit from embedding signal-processing structure into the network, leading to model-driven approaches such as OAMP-Net and related unfolded detectors [5], [6]. DeepSIC represents another branch, where neural components are inserted into an iterative soft interference-cancellation receiver [7]. Prior complex-valued neural-receiver work has also addressed joint channel estimation and decoding in massive MIMO with semi-supervised tracking for dynamic channels [8]. Recent surveys and analyses show that learned MIMO detection is now a mature research direction, while also emphasizing complexity, robustness and interpretability as key open issues [9], [10].

The present work is closest to the intersection of three lines of research: learned MIMO receivers, sequence detection in channels with memory and quantized neural-network deployment on FPGAs. Sliding recurrent receivers have shown the value of temporal context for communication channels with memory [11]. At the same time, recent low-complexity MIMO detectors, such as partial-MAP neural approaches, reinforce the importance of performance/complexity trade-offs [12]. On the hardware side, FINN-R and related n-bit quantized neural-network studies provide the academic basis for low-precision co-design on reconfigurable logic [13], [14]. Unlike works focused mainly on detection algorithms or software accuracy, this study combines finite-memory MIMO equalization, input/internal quantization, FINN synthesis, and physical-board accuracy, throughput and resource measurements on a

Jeison Magallanes Poma, Jonathan Aguiar Soares, Icaro Freitas, Marco Roberto Cavallari, Christian Esteve Rothenberg and Dalton S. Arantes are with the School of Electrical and Computer Engineering, University of Campinas (UNICAMP), Campinas, Brazil. E-mails: j300873@dac.unicamp.br, jonathan@decom.fee.unicamp.br, i278455@dac.unicamp.br, mrcavall@unicamp.br, chesteve@unicamp.br, dalson@unicamp.br. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — Brasil (CAPES) — Finance Code 001, and the São Paulo Research Foundation (FAPESP) under grants 2021/00199-8, 2025/05448-7, and 2025/11548-4, INCT NAMITEC (CNPq 406193/2022-3), and NanoQuantSS (CNPq 408596/2024-4). J. A. Magallanes thanks CNPq for the scholarship (130218/2026-0). *Corresponding author: J. A. Soares.*

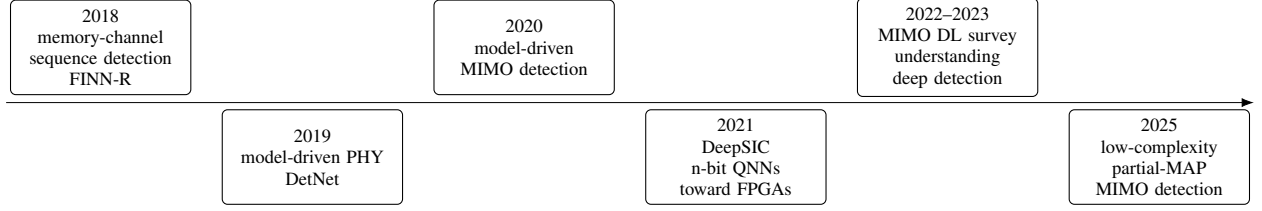


Fig. 1. Timeline of selected works related to learning-based MIMO detection, sequence detection for channels with memory, and quantized neural-network deployment on FPGAs.

III. MIMO CHANNEL WITH MEMORY

The dataset models a real-valued BPSK MIMO link with finite channel memory. The adopted model follows the tapped-delay-line interpretation commonly used for multipath wireless channels, where delayed replicas of the transmitted signal are represented by channel taps with delay-dependent average powers [15]. This formulation introduces inter-symbol interference because the current received vector depends on both current and past transmitted vectors [16]. At time t , the transmitted vector is

$$\mathbf{y}[t] \in \{-1, +1\}^{N_t}, \quad (1)$$

and the received vector is generated as

$$\mathbf{x}[t] = \sum_{\ell=0}^{L-1} \mathbf{H}_\ell \mathbf{y}[t-\ell] + \mathbf{n}[t], \quad (2)$$

where $\mathbf{H}_\ell$ is the ℓ th channel tap, L is the number of taps and $\mathbf{n}[t]$ is additive white Gaussian noise. In the implemented experiments, $N_t = N_r = 2$, $L = 3$, and the main memory window has depth $D = 4$. The choice is motivated by reduced-tap channel modeling studies. Therefore, the adopted three-tap model provides a controlled and hardware-friendly benchmark for evaluating the proposed neural equalizer and its FPGA deployment flow.

The channel taps are fixed during each dataset generation. Training and test samples use independent transmitted-symbol and noise realizations for the same channel configuration. Consequently, the test samples contain unseen symbol and noise realizations, but they do not represent previously unseen channel taps. The current evaluation is therefore channel-specific and does not establish generalization to independently generated or time-varying channels. The average tap power follows an exponentially decaying power-delay profile, which is a common simplified model for multipath tapped-delay-line channels [15]. The tap powers are normalized as

$$p_\ell = \frac{\exp(-\ell)}{\sum_{k=0}^{L-1} \exp(-k)}, \quad \ell = 0, \dots, L-1, \quad (3)$$

where each entry of $\mathbf{H}_\ell$ is drawn from a zero-mean Gaussian distribution with variance p_ℓ/N_t , i.e., the random coefficients are scaled by $\sqrt{p_\ell/N_t}$.

This normalization keeps the average received power controlled across the received dimensions.

Let

$$\mathbf{x}_{\text{clean}}[t] = \sum_{\ell=0}^{L-1} \mathbf{H}_\ell \mathbf{y}[t-\ell]. \quad (4)$$

The additive noise is generated independently at each receive antenna [3] as

$$\mathbf{n}[t] \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbf{I}_{N_r}). \quad (5)$$

The per-antenna noise variance is set from the empirical clean received-signal power as:

$$\sigma_n^2 = \frac{\mathbb{E} \{ \|\mathbf{x}_{\text{clean}}[t]\|_2^2 \}}{N_r 10^{\text{SNR}_{\text{dB}}/10}}. \quad (6)$$

With this definition, the average vector SNR is

$$\text{SNR} = \frac{\mathbb{E} \{ \|\mathbf{x}_{\text{clean}}[t]\|_2^2 \}}{\mathbb{E} \{ \|\mathbf{n}[t]\|_2^2 \}}. \quad (7)$$

The equalizer does not observe a single received vector. Instead, it receives a time window

$$\mathbf{X}_D[t] = [\mathbf{x}[t-D+1], \dots, \mathbf{x}[t]], \quad (8)$$

stored in the implementation as a tensor with shape $[1, D, N_r]$. In the main experiments, $D = 4$ and $L = 3$, so the input window covers the full finite-memory span of the channel and provides one additional received sample of temporal context. This explicit temporal context is consistent with prior learning-based sequence detectors for communication channels with memory [11]. The target is the class index associated with the BPSK vector $\mathbf{y}[t]$, resulting in $2^{N_t} = 4$ classes for the 2×2 case. The training and test sets use 60,000 and 10,000 samples, respectively.

Memory-aware ZF and LMMSE are used as low-complexity references with the same received window, exact channel taps and noise variance [17]. The CNV receives no explicit channel-state input, but is trained for that fixed channel realization. Therefore, the comparison is channel-specific and the linear references do not establish the gap to the optimal memory-aware MAP detector.

Let the received window be stacked as

$$\bar{\mathbf{x}}[t] = [\mathbf{x}^T[t-D+1], \dots, \mathbf{x}^T[t]]^T \in \mathbb{R}^{DN_r}. \quad (9)$$

The transmitted symbol vectors involved in this received window are stacked as

$$\bar{\mathbf{y}}[t] = [\mathbf{y}^T[t-D-L+2], \dots, \mathbf{y}^T[t]]^T \in \mathbb{R}^{T_{\text{seq}} N_t}, \quad (10)$$

where $T_{\text{seq}} = D + L - 1$. Therefore, the windowed channel can be written as

$$\bar{\mathbf{x}}[t] = \mathbf{G} \bar{\mathbf{y}}[t] + \bar{\mathbf{n}}[t], \quad (11)$$

where $\mathbf{G} \in \mathbb{R}^{DN_r \times T_{\text{seq}}N_t}$ is the extended convolutional channel matrix built from the channel taps $\{\mathbf{H}_\ell\}_{\ell=0}^{L-1}$.

A selection matrix $\mathbf{S} \in \mathbb{R}^{N_t \times T_{\text{seq}}N_t}$ is used to extract the current symbol vector $\mathbf{y}[t]$ from $\bar{\mathbf{y}}[t]$. The memory-aware ZF detector is then defined as

$$\tilde{\mathbf{y}}_{\text{ZF}}[t] = \mathbf{S}\mathbf{G}^\dagger \bar{\mathbf{x}}[t], \quad (12)$$

where $(\cdot)^\dagger$ denotes the Moore–Penrose pseudoinverse. Since the extended system may be underdetermined, this formulation gives the minimum-norm least-squares estimate before the final hard decision.

The memory-aware LMMSE detector assumes $\mathbb{E}\{\bar{\mathbf{y}}\bar{\mathbf{y}}^T\} = \mathbf{I}$ and $\mathbb{E}\{\bar{\mathbf{n}}\bar{\mathbf{n}}^T\} = \sigma_n^2\mathbf{I}$. The corresponding estimate of the current symbol vector is

$$\tilde{\mathbf{y}}_{\text{LMMSE}}[t] = \mathbf{S}\mathbf{G}^T (\mathbf{G}\mathbf{G}^T + \sigma_n^2\mathbf{I})^{-1} \bar{\mathbf{x}}[t]. \quad (13)$$

Finally, both linear estimates are mapped to the nearest BPSK symbol vector using the same decision rule used for BER and accuracy evaluation:

$$\hat{\mathbf{y}}[t] = \mathcal{Q}_{\text{BPSK}}(\tilde{\mathbf{y}}[t]). \quad (14)$$

IV. QUANTIZED CNV EQUALIZER

The proposed equalizer is a compact quantized convolutional neural network (CNN) implemented as a FINN-compatible CNV topology for quantized dataflow synthesis [13], [18]. The network begins with an input quantizer with bit width b_{in} . It then applies two temporal convolutional layers with kernels of size 2×1 , each followed by batch normalization and an activation quantizer with bit width b_a . The classifier head flattens the convolutional features, applies a quantized fully connected hidden layer, batch normalization, another activation quantizer and a final quantized linear layer with four outputs. The predicted MIMO symbol is obtained by the largest output score.

Weights are quantized with bit width b_w . Models were trained for 50 epochs with Brevitas [19] using Adam, squared-hinge loss and weight clipping to $[-1, 1]$. The initial learning rate was 0.02, the training batch size was 100, the random seed was 1, and the best-performing checkpoint was retained; the learning rate was halved after epoch 40.

The selected compact model uses $(C_1, C_2, F) = (12, 12, 12)$, while a $(64, 64, 64)$ variant evaluates whether additional width compensates for lower precision. All hardware builds used a FINN target throughput of 3000 frames/s with automatic folding. Hardware validation processed the 10,000-window test set, and throughput was obtained from the FINN/PYNQ driver.

The quantization sweep confirms that input precision is the dominant bottleneck. With $b_{\text{in}} = 2$, increasing (b_w, b_a) from $(2, 2)$ to $(4, 4)$ changes software accuracy only from 40.98% to 41.06%. With $b_{\text{in}} = 3$, the corresponding results are 76.98% and 84.24%. Thus, moderate low precision is effective, but information lost through extreme input quantization cannot be recovered solely by increasing internal precision or network width.

V. FPGA PLATFORM AND FLOW

Fig. 2 summarizes the experimental flow. The PyTorch/Brevitas model is trained on the host machine, exported to QONNX, and compiled by FINN into a dataflow accelerator [13], [18]. FINN produces resource reports, stitched IP, a bitstream, a PYNQ driver and a deployment package. The package and test set are copied to the board, where the PYNQ driver executes the accelerator and reports accuracy and throughput.

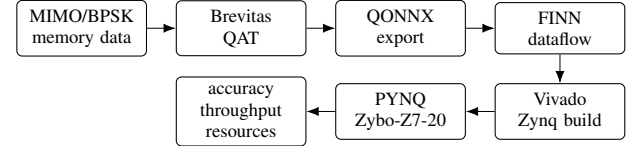


Fig. 2. End-to-end emulation and hardware implementation flow used in the experiments.

The target board is the Digilent Zybo-Z7-20, which contains the Xilinx Zynq-7020 SoC. The device combines a dual-core ARM Cortex-A9 processing system with Artix-7 programmable logic. The relevant programmable-logic resources are 53,200 LUTs, 106,400 flip-flops, 140 36Kb block RAMs and 220 DSP slices [20]. The board was booted using a PYNQ SD-card image adapted for Digilent Zybo boards, which provides the PYNQ runtime required to load the generated bitstream, access the hardware overlay and execute the FINN-generated Python driver on the embedded ARM processor [21]. The FPGA processes stored test windows; thus, the measured throughput excludes ADC/DAC conversion, RF processing, synchronization, online channel estimation and channel coding. The implementation is an embedded proof of concept, while over-the-air operation requires an RFSoc or external RF hardware.

VI. EXPERIMENTAL RESULTS

The experiments report vector accuracy, BER, PYNQ-measured throughput and post-synthesis resources. Software and hardware accuracy use the same test set. Accuracy counts correctly classified two-bit vectors, whereas BER counts individual bit errors; therefore, they are related but not complementary metrics.

A. Quantization and Resource Use

Table I reports the main hardware-validated configurations at SNR = 12 dB. The entry “in/w/a” denotes input, weight and activation bit widths, and C^3 denotes (C_1, C_2, F) .

The compact 8/6/6 model reaches 96.19% hardware vector accuracy at 1.323 Msamples/s. Increasing internal precision to 8/7/7 provides only a 0.05 percentage-point hardware gain, while throughput falls to 0.771 Msamples/s and DSP occupation decreases from 220 to 37. Although both builds used the same target-throughput setting, their post-synthesis resources and measured rates indicate substantially different FINN parallelization and serialization mappings. Therefore, Table I compares complete precision–parallelism operating

TABLE I
HARDWARE-VALIDATED QUANTIZATION RESULTS AT SNR = 12 dB;
EACH ROW USES ITS GENERATED FINN MAPPING.

Config in/w/a, C^3	Acc. (%)	Thr. (M/s)	LUT (k)	FF (k)	BRAM 36K	DSP
4/3/3, 64^3	94.86	0.471	19.4	24.4	24	220
6/4/4, 12^3	95.34	1.322	14.0	17.3	18	220
8/4/4, 12^3	95.46	1.321	14.9	17.3	18	220
8/6/6, 12^3	96.19	1.323	33.6	21.9	26	220
8/7/7, 12^3	96.24	0.771	24.2	13.5	5 [†]	37

[†]The 8/7/7 case also reports one BRAM_18K block.

points rather than precision alone. The wider 64^3 , 4/3/3 model reaches 94.86% at only 0.471 Msamples/s. Across the hardware-validated configurations, software and FPGA vector accuracies differ by at most 0.07 percentage points, confirming the numerical consistency of the exported QONNX/FINN implementation.

The 8/6/6 build uses 63.1% of LUTs, 20.6% of flip-flops, 18.6% of BRAMs and all 220 DSPs; it is therefore DSP-bound under this FINN mapping. Higher folding can release DSPs but lowers throughput. Scaling width and precision is also LUT-limited: the 64^3 , 8/7/7 model reaches 97.17% software accuracy but does not fit the device, while the 64^3 , 8/8/8 and 12^3 , 10/10/10 configurations also exceed its LUT capacity. Moreover, joint classification scales as M^{N_t} , so a 4×4 16-QAM extension would require 65,536 outputs. Larger systems therefore require stronger folding, mixed precision, pruning, scalable per-stream or bit-wise heads, or a larger FPGA.

B. SNR Sweep

After selecting the compact 8/6/6 configuration, the SNR was swept from 0 to 16 dB. Since the trained weights, bit widths, topology and folding are unchanged, the FPGA resource use is unchanged across the sweep; only the received samples and noise level vary.

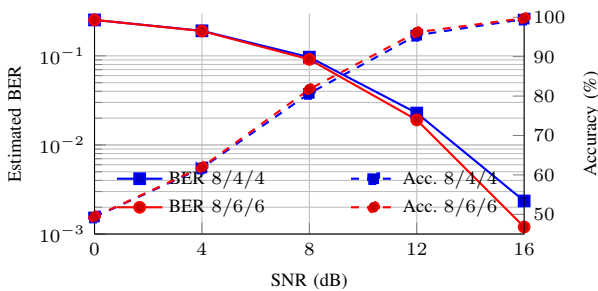


Fig. 3. SNR sweep for the compact 12^3 CNV. BER is shown on the logarithmic left axis and accuracy is shown on the right axis; zero-valued BER points are omitted from the logarithmic plot.

As SNR increases, reduced noise makes $\mathbf{X}_D[t]$ more informative, increasing vector accuracy and lowering BER [22]. Table II reports 49.42%, 81.74%, 96.19% and 99.76% accuracy at 0, 8, 12 and 16 dB, respectively. At 8 dB, 81.74% vector accuracy implies an 18.26% vector-error rate, whereas BER is 9.13%, indicating that most wrong vectors contain one

bit error. These are uncoded detector-level metrics; throughput remains near 1.32 Msamples/s because the accelerator is unchanged.

TABLE II
SNR SWEEP FOR THE SELECTED 8/6/6, 12^3 CONFIGURATION.

SNR (dB)	Acc. (%)	Est. BER	HW thr. (M/s)
0	49.42	0.2530	1.322
4	61.97	0.1900	1.314
8	81.74	0.0913	1.320
12	96.19	0.0191	1.323
16	99.76	0.0012	1.318

The 1.323 Msamples/s throughput corresponds to $0.756 \mu\text{s}$ per stored input window, not to complete radio-link latency.

C. Channel-Memory Sweep

The memory sweep evaluates $(L, D) = (2, 3)$ to $(6, 7)$, keeping channel counts, hidden width and quantization fixed; input-dependent tensor dimensions and the independently generated FINN mapping may change with D .

TABLE III
EFFECT OF CHANNEL-MEMORY SWEEP FOR THE SELECTED 8/6/6, 12^3
CNV CONFIGURATION AT SNR = 12 dB.

L	D	HW Acc. (%)	HW Thr. (M/s)	LUT (k)	FF (k)	BRAM 36K	DSP
2	3	98.98	1.323	34.6	21.8	25 [†]	220
3	4	96.19	1.323	33.6	21.9	26	220
4	5	92.89	1.317	32.4	21.6	25 [†]	220
5	6	93.62	1.324	35.2	21.7	25 [†]	220
6	7	88.31	1.323	35.8	21.9	25 [†]	220

[†]These cases also report one BRAM_18K block.

The longest span gives the lowest accuracy (88.31%), but the trend is not monotonic: $(L, D) = (5, 6)$ reaches 93.62%, above 92.89% for $(4, 5)$. Because each point uses a separately generated channel, accuracy also depends on tap realization and conditioning. The nearly constant PYNQ throughput does not imply equal computational cost, since FINN compiles each input size with its own folding and runtime overhead.

D. Memory-Aware Linear Reference Detectors

Fig. 4 compares the hardware-validated 8/6/6, 12^3 CNV equalizer with the linear baselines defined in Section III. These baselines follow the classical MIMO detection framework reviewed in [17], but are applied here to the extended convolutional channel matrix to account for finite memory.

For the evaluated fixed channel, the CNV has lower empirical BER than ZF/LMMSE at moderate and high SNR. At 12 dB, BER decreases from 9.55×10^{-2} (LMMSE) to 1.91×10^{-2} (CNV), about $5.0\times$. At 16 dB, the point estimates are 5.57×10^{-2} and 1.2×10^{-3} ; however, the CNV value is estimated from the 10,000-vector test set and contains few observed errors, making the exact multiplicative gain statistically fragile. These channel-specific results do not demonstrate unseen-channel robustness.

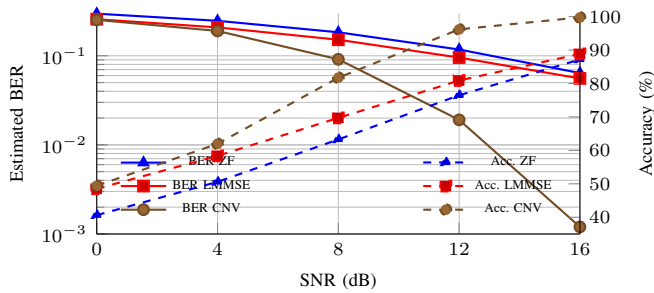


Fig. 4. SNR sweep comparing memory-aware ZF and LMMSE references with the selected hardware-validated 8/6/6, 12^3 CNV equalizer. BER is shown on the logarithmic left axis and accuracy is shown on the right axis.

VII. CONCLUSION

This work demonstrated an end-to-end Brevitas–QONNX–FINN–PYNQ implementation of a quantized CNV equalizer for a real-valued 2×2 BPSK MIMO channel with finite memory. The 8/6/6, 12^3 design reaches 96.19% hardware vector accuracy and approximately 1.32 Msamples/s at 12 dB on the Zybo-Z7-20. Software and FPGA accuracies differ by at most 0.07 percentage points across the hardware-validated configurations, supporting the numerical consistency of the exported implementation. The quantization sweep also shows that input precision is the main low-bit bottleneck: increasing internal precision cannot recover information lost by an excessively quantized received window.

The SNR sweep confirms that the same accelerator improves from 49.42% accuracy at 0 dB to 99.76% at 16 dB while maintaining nearly constant throughput. The channel-memory study indicates that longer spans are generally more challenging for the fixed compact architecture, although the trend is not strictly monotonic because each point also depends on the individual channel realization. For the evaluated fixed channel, the CNV achieves lower empirical BER than the memory-aware ZF and LMMSE references, but the results do not establish unseen-channel robustness or proximity to the optimal MAP detector.

The selected implementation is DSP-bound, while wider and higher-precision configurations can become LUT-limited. Moreover, the joint-classification output does not scale directly to larger MIMO/QAM systems. Future work includes training over multiple and unseen channels, pilot-based channel conditioning, comparison with MAP detection, dilated or recurrent models for longer temporal memory, scalable per-stream or bit-wise output heads, and RFSoc-based over-the-air validation [23].

REFERENCES

- [1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
- [2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, “Edge intelligence: Paving the last mile of artificial intelligence with edge computing,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.
- [3] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge University Press, 2005.
- [4] N. Samuel, T. Diskin, and A. Wiesel, “Learning to detect,” *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, May 2019. [Online]. Available: <https://doi.org/10.1109/TSP.2019.2899805>

- [5] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, “Model-driven deep learning for physical layer communications,” *IEEE Wireless Communications*, vol. 26, no. 5, pp. 77–83, Oct. 2019. [Online]. Available: <https://doi.org/10.1109/MWC.2019.1800447>
- [6] H. He, C.-K. Wen, S. Jin, and G. Y. Li, “Model-driven deep learning for MIMO detection,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 1702–1715, Mar. 2020. [Online]. Available: <https://doi.org/10.1109/TSP.2020.2976585>
- [7] N. Shlezinger, R. Fu, and Y. C. Eldar, “DeepSIC: Deep soft interference cancellation for multiuser MIMO detection,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1349–1362, Feb. 2021. [Online]. Available: <https://doi.org/10.1109/TWC.2020.3032663>
- [8] J. A. Soares, K. S. Mayer, and D. S. Arantes, “Semi-supervised ML-based joint channel estimation and decoding for m-MIMO with gaussian inference learning,” *IEEE Wireless Communications Letters*, vol. 12, no. 12, pp. 2123–2127, 2023. [Online]. Available: <https://doi.org/10.1109/LWC.2023.3309479>
- [9] M. A. M. Albreem, A. H. Alhabbasha, S. Shahabuddin, and M. Juntti, “Deep learning for massive MIMO uplink detectors,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 741–766, 2022. [Online]. Available: <https://doi.org/10.1109/COMST.2021.3135542>
- [10] Q. Hu, F. Gao, H. Zhang, G. Y. Li, and Z. Xu, “Understanding deep MIMO detection,” *IEEE Transactions on Wireless Communications*, vol. 22, no. 12, pp. 9626–9639, Dec. 2023. [Online]. Available: <https://doi.org/10.1109/TWC.2023.3272525>
- [11] N. Farsad and A. Goldsmith, “Sliding bidirectional recurrent neural networks for sequence detection in communication systems,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2331–2335. [Online]. Available: <https://doi.org/10.1109/ICASSP.2018.8462140>
- [12] L. Bai, Q. Zeng, R. Han, J.-H. Choi, and W. Zhang, “Deep learning-based low complexity MIMO detection via partial MAP,” *IEEE Transactions on Wireless Communications*, vol. 24, no. 3, Mar. 2025. [Online]. Available: <https://doi.org/10.1109/TWC.2024.3516738>
- [13] M. Blott, T. B. Preusser, N. J. Fraser, G. Gambardella, K. O’Brien, Y. Umuroglu, M. Leeser, and K. A. Vissers, “FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks,” *ACM Transactions on Reconfigurable Technology and Systems*, vol. 11, no. 3, Dec. 2018. [Online]. Available: <https://doi.org/10.1145/3242897>
- [14] J. Chen, L. Liu, Y. Liu, and X. Zeng, “A learning framework for n-bit quantized neural networks toward FPGAs,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, Mar. 2021. [Online]. Available: <https://doi.org/10.1109/TNNLS.2020.2980041>
- [15] S. Sagari, W. Trappe, and L. Greenstein, “Equivalent tapped delay line channel responses with reduced taps,” in *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, 2013, pp. 1–5.
- [16] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. New York, NY, USA: McGraw-Hill, 2008.
- [17] S. Yang and L. Hanzo, “Fifty years of MIMO detection: The road to large-scale MIMOs,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 1941–1988, 2015.
- [18] 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, pp. 65–74. [Online]. Available: <https://doi.org/10.1145/3020078.3021744>
- [19] A. Pappalardo, “Brevitas: A PyTorch library for neural network quantization and quantization-aware training,” <https://github.com/Xilinx/brevitas>, accessed: 2026-05-24.
- [20] AMD Xilinx, *Zynq-7000 SoC Product Selection Guide*, product selection guide for the XC7Z020 device family.
- [21] N. Petrovsky, “PYNQ-ZYBO: Pynq 3.0.1 for diligent zybo boards,” <https://github.com/nick-petrovsky/PYNQ-ZYBO>, 2023, gitHub repository, accessed: 2026-04-18.
- [22] Y. Sun, C. Wang, H. Cai, C. Zhao, Y. Wu, and Y. Chen, “Deep learning based equalizer for mimo-ofdm systems with insufficient cyclic prefix,” in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020, pp. 1–5.
- [23] F. D. L. Coutinho, J. A. Soares, K. S. Mayer, H. S. Silva, A. S. R. Oliveira, and D. S. Arantes, “Over-the-air performance evaluation of real and complex neural networks for joint channel estimation and equalization in QAM systems,” in *2025 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*, 2025.