A Compute&Memory Efficient Model-Driven Neural 5G Receiver for Edge AI-assisted RAN

Mahdi Abdollahpour[§] Marco Bertuletti[*] Yichao Zhang[*] Yawei Li[*]
Luca Benini[§*] Alessandro Vanelli-Coralli[§*]
[§]DEI, University of Bologna [*]IIS, ETH Zürich
[§]{*mahdi.abdollahpour,luca.benini,alessandro.vanelli*}*@unibo.it*, [*]{*mbertuletti,yiczhang,yawli,lbenini,avanelli*}*@iis.ee.ethz.ch*

***Abstract*—Artificial intelligence approaches for base-band processing for radio receivers have demonstrated significant performance gains. Most of the proposed methods are characterized by high compute and memory requirements, hindering their deployment at the edge of the Radio Access Networks (RAN) and limiting their scalability to large bandwidths and many antenna 6G systems. In this paper, we propose a low-complexity, model-driven neural network-based receiver, designed for multi-user multiple-input multiple-output (MU-MIMO) systems and suitable for implementation at the RAN edge. The proposed solution is compliant with the 5G New Radio (5G NR), and supports different modulation schemes, bandwidths, number of users, and number of base-station antennas with a single trained model without the need for further training. Numerical simulations of the Physical Uplink Shared Channel (PUSCH) processing show that the proposed solution outperforms the state-of-the-art methods in terms of achievable Transport Block Error Rate (TBLER), while reducing the Floating Point Operations (FLOPs) by 66×, and the learnable parameters by 396×.**

***Index Terms*—5G, channel estimation, convolutional neural networks, neural receiver**

I. Introduction

The evolution of Beyond-5G (B5G) and 6G Radio Access Networks (RAN), enabled by diverse edge functions, is rapidly advancing the network's service quality, capabilities, and user densities in complex deployment scenarios [1]. With the evolution of functional disaggregation in 3rd Generation Partnership Project (3GPP) and Open-RAN (O-RAN), Artificial Intelligence (AI)-based RAN processing is emerging as a key trend, promoted by open interfaces [2]. Processing the uplink Physical (PHY)-layer at the RAN edge is crucial for improving latency, performance, and system flexibility, but it stands out as one of the most computational and memory demanding RAN functions [3].

Recent research [4] has focused on AI-for-RAN to enhance PHY-layer performance. Neural Network (NN)-based Orthogonal Frequency Division Multiple Access (OFDMA) receivers have demonstrated improved Bit-Error-Rate (BER) performance compared to conventional Linear Minimum Mean Squared Error (LMMSE)-receivers [5]–[8]. However, these models incur high computational complexity and large memory footprint compared to classical approaches, exacerbating the computational bottleneck of the PHY processing.

As a consequence of growing computational requirements for mixed AI&wireless workloads, base-station edge-processors are evolving from RAN-specialized Application Specific Integrated Circuits (ASICs) to high-performance many-core programmable processors [9]–[13].

The top three rows of table I provide an insight into the computing capability, on-chip memory of these devices relative to the performance and memory footprint required by a runtime-compliant state-of-the-art (SoA) neural receiver (NRX) [7], addressing a 4×2 Multiple-User (MU)-Multiple-Input, Multiple-Output (MIMO) problem (4 receive antennas, 2 transmit data streams) in a 1-ms Transmission Time Interval (TTI) allocating 273 Physical Resource Blocks (PRBs) per stream. None of the devices intended for edge deployment can provide the required performance (as indicated by the required-to-processor performance ratios). High performance CPUs and GPUs have significantly higher capabilities, but they vastly exceed the power budget of a base station edge-processor. Hence, they need to be accessed via the fronthaul link of the Next Generation Node B (gNB), causing a significant increase of end-to-end latency.

TABLE I
Increase in computational complexity and memory capacity required to SoA RAN processors for Edge deployment of NRX

	Edge	16b-TFLOPs	RAM (MiB)	Power (W)	Req.-Perf.	Req.-Mem.[a]
Marvell OCTEON10 [9]	yes	1	24	50	75.86×	1.09×
TeraPool [10]	yes	3.7	4	7	20.86×	6.54×
Qualcomm X100 [11]	yes	-	-	18	-	-
NVDIA H100 [12]	no	1979	50	510	0.04×	0.52×
Intel Sapphire Rapids [13]	no	75.3	112.5	-	1.04×	0.23×

[a]Size of model parameters plus inputs.

This status quo stresses the need for lightweight models tailored for edge deployment that curtail compute and memory resources, preserving critical RAN performance metrics.

The SoA neural receivers use a fully data-driven approach which leads to large models with high computational costs [5], [7], [8]. In contrast, in our solution we follow a model-driven approach: we augment the conventional LMMSE receiver with learnable parameters, and specialized ResNet blocks (ResBlocks) processing to suppress noise and interference, which results in significant complexity reduction.

In this paper, we propose the design of a small, efficient Model Driven Neural Receiver (MDX), fully compatible with the 5th Generation (5G) New Radio (NR) and suitable for edge-RAN deployment. The proposed design is available open-source [14]. The main contributions of the proposed design are:

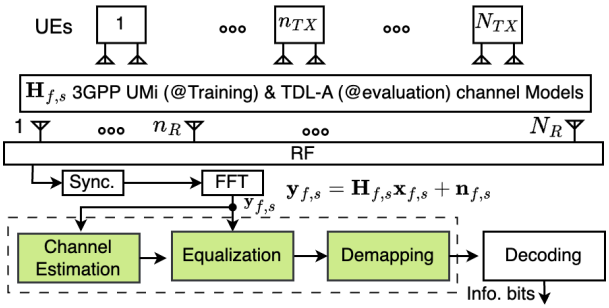

Fig. 1. The Mu-MIMO uplink system model. The colored conventional processing blocks are replaced with the proposed MDX model.

- The combined design of conventional baseband processing blocks with learnable parameters and specialized ResBlocks processing to suppress noise and interference, approaching Transport Block Error Rate (TBLER) performance near the limit with perfect channel knowledge.
- The evaluation of the receiver performance for the Physical Uplink Shared Channel (PUSCH) in terms of TBLER in 3GPP Tapped Delay Line (TDL)-A channel model.
- The assessment of the computational complexity, and the memory footprint of our model against SoA AI-receivers.

Our proposed MDX delivers superior TBLER performance and also achieves a 66× reduction in Floating Point Operations (FLOPs) and a 396× reduction in model parameters compared to the purely data-driven SoA, making it well suited for edge deployment. Furthermore, despite being trained only on a small 4×2 MIMO problem, we show that MDX effectively generalizes to support larger MIMO sizes and diverse configurations (e.g., modulation schemes, bandwidths), facilitating practical online training on edge devices.

II. System Model

This section describes the telecommunication system (Fig. 1) used for our model evaluation. We consider the 5G NR MU-MIMO Orthogonal Frequency Division Multiplexing (OFDM) uplink transmission, where N_{TX} layers (data streams) from N_U User Equipments (UEs) are transmitted to a gNB with N_R receiving antennas. Each UE may have multiple antennas and multiple layers: we generalize to multiple antennas by assuming 2 transmitting antennas and 1 layer per UE.

Each layer's transmission is divided into time slots. Within each slot the data is mapped on a Resource Grid (RG): the set $\mathcal{RG} = \{1,\dots,F\}\times\{1,\dots,S\}$, where F is the number of subcarriers, and S is the number of OFDM symbols in a slot. The smallest RG unit is a Resource Element (RE), identified by a subcarrier index f and an OFDM symbol index s, $(f,s) \in \mathcal{RG}$. Each RE, associated with the layer n_{TX}, where $1 \le n_{TX} \le N_{TX}$, carries a complex symbol $x_{f,s,n_{TX}}$. The symbol $x_{f,s,n_{TX}}$ is defined by a $2^{B_{n_{TX}}}$ Quadrature Amplitude Modulation (QAM) constellation, and encodes a vector of bits $\mathbf{b}_{f,s,n_{TX}} \in \{0,1\}^{B_{n_{TX}}}$, where $B_{n_{TX}}$ denotes QAM order of layer n_{TX}. Slots typically contain 14 OFDM symbols, and the frequency domain is organized into PRBs, each PRB comprising 12 subcarriers, which serve as the fundamental units for resource allocation [15].

After the FFT, the received signal on RE (f,s) is:

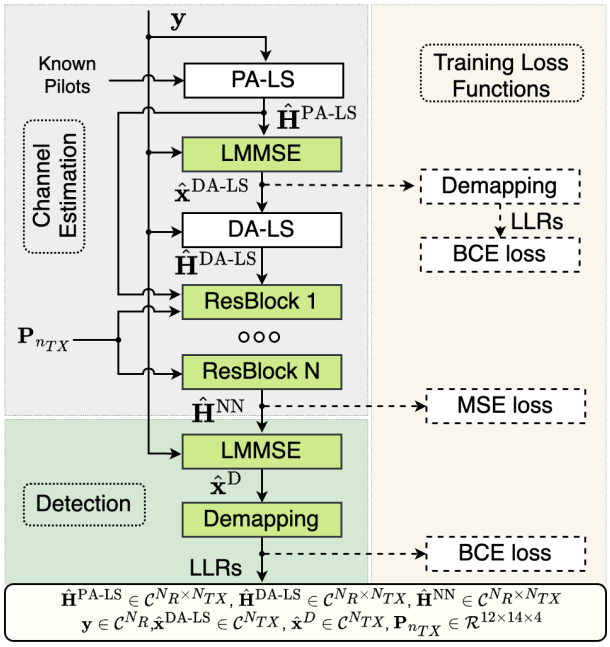

Fig. 2. Model driven neural receiver block diagram. The colored blocks include trainable weights. The dashed blocks are used only in training.

$$\mathbf{y}_{f,s} = \mathbf{H}_{f,s}\mathbf{x}_{f,s} + \mathbf{n}_{f,s}, \tag{1}$$

where $\mathbf{y}_{f,s} \in \mathcal{C}^{N_R}$, and $\mathbf{x}_{f,s} \in \mathcal{C}^{N_{TX}}$ are the received and transmitted symbols, $\mathbf{H}_{f,s} \in \mathcal{C}^{N_R \times N_{TX}}$ is the MIMO channel matrix, and $\mathbf{n}_{f,s} \in \mathcal{C}^{N_R}$ is the complex additive Gaussian noise with power spectral density N_0, distributed as $\mathcal{CN}(\mathbf{0}, N_0\mathbf{I})$.

The majority of REs carry data symbols. A subset within each RG transmits the Demodulation Reference Signals (DMRSs), also known as pilots. The data-carrying positions can be defined as the set $\mathcal{D}$, including all RE indices (f,s) corresponding to data symbols. DMRSs and their positions within an RG are layer-specific, known at the receiver, and used for channel estimation. The DMRS positions for layer n_{TX} are defined by the set $\mathcal{P}_{n_{TX}}$ of all the (f,s) indices of REs carrying pilots. In our configuration, the DMRS positions are located at OFDM symbols 2 and 11 and are configured to use a Code-Division Multiplexing (CDM) group size of 2 in the frequency domain.

III. Model Driven Neural Receiver (MDX)

In this section, we describe the architecture of our lightweight model: a mix of AI-enhanced classical signal processing blocks and AI blocks. The overall architecture of MDX is shown in Fig. 2. The model applies first Pilot Aided Least Squares (PA-LS) channel estimation, then LMMSE equalization with learnable parameters. The core of the MDX network consists of Data Aided Least Squares (DA-LS) channel estimation, specialized ResBlocks, and a Demapper with learnable parameters, which outputs the final bit Log-Likelihood Ratios (LLRs). Finally, the LLRs undergo Low Density Parity Check Code (LDPC) decoding. We detail these steps in the following.

A. PA-LS Channel Estimation

In 5G MU-MIMO, pilot sequences assigned to different layers are orthogonal in time, frequency, or code domains.

These predefined signals are used to obtain an initial estimate of channel vectors at the RE positions carrying pilots as

$$\hat{\mathbf{h}}^{\text{PA-LS}}_{f,s,n_{TX}} = \frac{p^*_{f,s,n_{TX}} \mathbf{y}_{f,s}}{|p_{f,s,n_{TX}}|^2} \in C^{N_R}, \quad \forall [f,s] \in \mathcal{P}_{n_{TX}}, \quad (2)$$

where $p_{f,s,n_{TX}}$ denotes the complex-valued pilot symbol transmitted by layer n_{TX} at RE (f,s), $\mathbf{y}_{f,s}$ represents the corresponding received signal, and the notation $(\cdot)^*$ denotes the complex conjugate. When CDM groups are used, the pilots of layers in the same CDM group are not orthogonal in time and in frequency. In such cases, the PA-LS channel estimate in (2) is averaged over the subcarriers in the same group. Then, linear interpolation is used to obtain the channel estimates at data-carrying REs. The PA-LS channel vectors can be arranged in matrix form as

$$\hat{\mathbf{H}}^{\text{PA-LS}}_{f,s} = \left[\hat{\mathbf{h}}^{\text{PA-LS}}_{f,s,1}, \ldots, \hat{\mathbf{h}}^{\text{PA-LS}}_{f,s,N_{TX}}\right] \in C^{N_R \times N_{TX}}. \quad (3)$$

B. Equalization and Demapping

We use the LMMSE equalization to obtain an estimate of the transmitted symbols. The LMMSE matrix is defined as

$$\mathbf{G}_{f,s} = \left(\hat{\mathbf{H}}^H_{f,s}\hat{\mathbf{H}}_{f,s} + \hat{\sigma}^2_{\text{adj},f,s} I\right)^{-1} \hat{\mathbf{H}}^H_{f,s} \in \mathbf{C}^{N_{TX} \times N_R}, \quad (4)$$

where $\hat{\mathbf{H}}_{f,s} \in C^{N_R \times N_{TX}}$ is an estimate of the MIMO channel. The superscript H denotes Hermitian operation, I is a $N_{TX} \times N_{TX}$ identity matrix, and

$$\hat{\sigma}^2_{\text{adj},f,s} = \mathbf{\Psi}(f',s) \cdot \hat{\sigma}_{f,s}, \quad (5)$$

where $\hat{\sigma}_{f,s}$ is the estimated noise variance, $f' = ((f-1) \bmod 12) + 1$, 'mod' being modulo operation, $\mathbf{\Psi} \in \mathcal{R}^{12\times 14}$ is a learnable matrix. Due to the sparse placement of pilot symbols within an RG, which provide only localized channel information, the channel estimation error varies across REs near the pilot locations and those farther away. The learnable matrix $\mathbf{\Psi}$ adjusts the LMMSE equalizer's input error variances on a per-PRB basis, enabling the receiver to adapt to these variations. The equalized symbols can be computed as

$$\hat{\mathbf{x}}_{f,s} = \text{Diag}(\mathbf{G}_{f,s}\hat{\mathbf{H}}_{f,s})^{-1}\mathbf{G}_{f,s}\mathbf{y}_{f,s} \in C^{N_{TX}}. \quad (6)$$

where the operator Diag$(\cdot)$ constructs a diagonal matrix using the diagonal elements of the input matrix. The effective post-equalization residual noise variances are

$$\hat{\boldsymbol{\sigma}}_{\text{res},f,s} = \text{diag}\left(\text{Diag}(\mathbf{G}_{f,s}\hat{\mathbf{H}}_{f,s})^{-1} - \mathbf{I}\right). \quad (7)$$

where, the operator diag$(\cdot)$ returns the diagonal elements of the input matrix as a vector, and $\hat{\boldsymbol{\sigma}}_{\text{res},f,s} = \{\hat{\sigma}^2_{\text{res},f,s,1}, \ldots, \hat{\sigma}^2_{\text{res},f,s,N_{TX}}\}$. The equalization process can be written as the function

$$\hat{\mathbf{x}}_{f,s}, \hat{\boldsymbol{\sigma}}_{\text{res},f,s} = \text{LMMSE}_{\mathbf{\Psi}}\left(\hat{\mathbf{H}}_{f,s}, \mathbf{y}_{f,s}, \hat{\sigma}^2_{f,s}\right). \quad (8)$$

Lets define LLR for a bit $\mathscr{b}$ as $\ell = \log(p/(1-p))$, where p denotes the probability of $\mathscr{b}=1$. Then the estimated symbols can be demapped onto LLRs as

$$\ell_b = \frac{1}{\hat{\sigma}^2_{\text{dem}}}\left(\arg\min_{x \in C^0_b} ||\hat{x} - x||_2^2 - \arg\min_{x \in C^1_b} ||\hat{x} - x||_2^2\right), \quad (9)$$

where the sets C^0_b and C^1_b represent the constellation points for which the b-th bit is 0 and 1, respectively, and $b = 1, \ldots, B$ with B denoting the number of bits per symbol. The noise variance is adjusted as

$$\hat{\sigma}^2_{\text{dem},f,s} = \gamma_m \mathbf{\Phi}(f',s) \cdot \hat{\sigma}^2_{\text{res},f,s}, \quad (10)$$

where $\mathbf{\Phi} \in \mathcal{R}^{12\times 14}$ is a learnable matrix, and γ_m for $m = 1, \ldots, M$ is a modulation-specific learnable scaler where M indicates the number of supported modulation orders. The demapping function can be written as

$$\ell_{b,f,s,n_{TX}} = \text{Demapper}_{\gamma_m,\mathbf{\Phi}}\left(\hat{\mathbf{x}}_{f,s}(n_{TX}), \hat{\boldsymbol{\sigma}}_{\text{res},f,s}(n_{TX})\right) \quad (11)$$

C. DA-LS Channel Estimation

For all $(f,s) \in \mathcal{D}$, the data symbols can be estimated using PA-LS channel matrix as

$$\hat{\mathbf{x}}^{DA\text{-}LS}_{f,s}, \hat{\boldsymbol{\sigma}}^{DA\text{-}LS}_{res,f,s} = \text{LMMSE}_{\mathbf{\Psi}^{DA\text{-}LS}}\left(\hat{\mathbf{H}}^{PA\text{-}LS}_{f,s}, \mathbf{y}_{f,s}, \hat{\sigma}^2_{f,s}\right). \quad (12)$$

Then the estimated symbols in (12) can be used to obtain a data-aided estimate of channel as

$$\hat{\mathbf{h}}^{\text{DA-LS}}_{f,s,n_{TX}} = \left(\mathbf{y}_{f,s} - \hat{\mathbf{H}}^{\text{PA-LS}}_{f,s,\setminus n_{TX}}\hat{\mathbf{x}}^{\text{DA-LS}}_{f,s,\setminus n_{TX}}\right) \cdot (\hat{\mathbf{x}}^{\text{DA-LS}}_{f,s}(n_{TX}))^*, \quad (13)$$

where, $\hat{\mathbf{h}}^{\text{DA-LS}}_{f,s,n_{TX}} \in C^{N_R}$, the subscript $(\cdot)_{\setminus n_{TX}}$ excludes the n_{TX}-th column or element from a matrix or a vector respectively. The resulting DA-LS channel vectors in matrix form are

$$\hat{\mathbf{H}}^{\text{DA-LS}}_{f,s} = \left[\hat{\mathbf{h}}^{\text{DA-LS}}_{f,s,1}, \ldots, \hat{\mathbf{h}}^{\text{DA-LS}}_{f,s,N_{TX}}\right] \in C^{N_R \times N_{TX}}. \quad (14)$$

Note that, the resulting channel estimate in (13) has different scaling from the actual channel. Additionally, it is noisy, and retains post-equalization residual interferences. However, the following neural processing, using our specialized ResNet blocks, can learn to adjust the scale and suppress noise and interference.

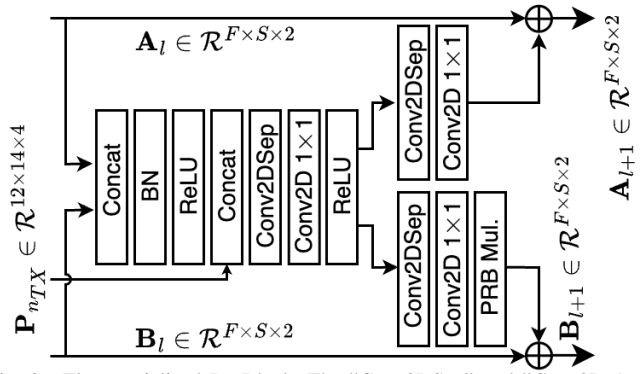

Fig. 3. The specialized ResBlock. The "Conv2DSep" and "Conv2D 1×1" form a depthwise separable convolution [16], and "BN" indicates batch normalization.

D. ResBlocks Processing

The so far estimated MIMO channel is enhanced by further processing through a series of N ResBlocks. The block diagram of our specialized ResBlock is illustrated in Fig. 3. In the figure, A_l and B_l are the outputs of previous ResBlock, and $P_{n_{TX}}$ is the positional encoding of layer n_{TX}. The block "PRB Mul." indicates a per-PRB weighting multiplier. The details of ResBlocks processing is explained in the following.

Defining the set of MIMO links as $\mathcal{L} = \{1,\ldots,N_R\} \times \{1,\ldots,N_{TX}\}$, the ResBlocks process every $(n_R, n_{TX}) \in \mathcal{L}$, in parallel using shared weights. The inputs of the network are:

1. The PA-LS channel estimates in (3) reshaped as a 3-Dimensional (3D) tensor $\mathbf{A}_1 \in \mathcal{R}^{F\times S\times 2}$, with its (f,s,k)-th element defined as

$$\mathbf{A}_1(f,s,k) = \begin{cases} \Re\left(\hat{H}^{\text{PA-LS}}_{f,s}(n_R,n_{TX})\right) & \text{if } k=1, \\ \Im\left(\hat{H}^{\text{PA-LS}}_{f,s}(n_R,n_{TX})\right) & \text{if } k=2, \end{cases} \tag{15}$$

2. The DA-LS channel estimates in (14) reshaped as a 3D tensor $\mathbf{B}_1 \in \mathcal{R}^{F\times S\times 2}$ with its (f,s,k)-th element defined as

$$\mathbf{B}_1(f,s,k) = \begin{cases} \Re\left(\hat{H}^{\text{DA-LS}}_{f,s}(n_R,n_{TX})\right) & \text{if } k=1, \\ \Im\left(\hat{H}^{\text{DA-LS}}_{f,s}(n_R,n_{TX})\right) & \text{if } k=2, \end{cases} \tag{16}$$

3. The layer-specific Positional Encoding (PE), $\mathbf{P}_{n_{TX}} \in \mathcal{R}^{12\times 14\times 4}$, which is provided to all ResBlocks. We incorporate a per-PRB positional encoding to capture the channel's structure across time and frequency domains. The encoding consists of two components. The first component $\dot{\mathbf{P}}$, inspired by [8], consists of the normalized relative vertical (in frequency) and horizontal (in time) distance of each RE inside a PRB from the nearest DMRS. It is a real-valued 3D tensor with a shape of $12\times 14\times 2$. Since the DMRS positions are layer-specific, this component of the PE is also layer-dependent. The second component, $\ddot{\mathbf{P}}$, is a real-valued 3D tensor of shape $12\times 14\times 2$, consisting of the normalized absolute cartesian coordinates of each RE inside a PRB, and is defined as

$$\ddot{\mathbf{P}}(f_p,s_p,k) = \begin{cases} \frac{f_p}{12} & \text{if } k=1, \\ \frac{s_p}{14} & \text{if } k=2, \end{cases} \quad \forall 1\le f_p\le 12, 1\le s_p\le 14. \tag{17}$$

Then, the overall PE is created by concatenating the two components, $\dot{\mathbf{P}}$ and $\ddot{\mathbf{P}}$, along the last dimension.

Algorithm 1 summarizes the ResBlocks processing. We use $N=4$ ResBlocks, where each ResBlock produces two outputs, A and B, except for the final block, which generates only A. The A outputs are weighted on a per-PRB basis before performing the residual summation, using a learnable multiplier $\mathbf{\Gamma}_l \in \mathbb{R}^{12\times 14\times 1}$, $l=1,\ldots,N$. The tensors are repeated and broadcast to the appropriate shapes when necessary. The network output for the link (n_R,n_{TX}) is denoted by $\mathbf{A}_{N+1} \in \mathbb{R}^{F\times S\times 2}$. This output can be interpreted as a complex-valued matrix $\mathbf{C}_{n_R,n_{TX}} \in \mathbb{C}^{F\times S}$.

E. Detection

We use the channel estimates enhanced by ResBlocks processing for the MIMO detection. The output of Alg. 1 can be written as $\hat{\mathbf{H}}^{\text{NN}}_{f,s} \in C^{N_R\times N_{TX}}$ where its (n_R,n_{TX})-th element defined as

$$\hat{\mathbf{H}}^{\text{NN}}_{f,s}(n_R,n_{TX}) = \mathbf{C}_{n_R,n_{TX}}(f,s). \tag{18}$$

The equalized symbols and the processed noise variance are

$$\hat{\mathbf{x}}^{\text{D}}_{f,s}, \hat{\boldsymbol{\sigma}}^{\text{D}}_{\text{res},f,s} = \text{LMMSE}_{\boldsymbol{\Psi}^{\text{D}}}\left(\hat{\mathbf{H}}^{\text{NN}}_{f,s}, \mathbf{y}_{f,s}, \hat{\sigma}_{f,s}\right). \tag{19}$$

Then the LLRs can be computed as

$$\ell^{\text{D}}_{b,f,s,n_{TX}} = \text{Demapper}_{\gamma_m,\boldsymbol{\Phi}}\left(\hat{\mathbf{x}}^{\text{D}}_{f,s}(n_{TX}), \hat{\boldsymbol{\sigma}}^{\text{D}}_{f,s}(n_{TX})\right) \tag{20}$$

F. Training MDX

The proposed MDX architecture is fully differentiable. It is trained by defining a loss function and back-propagating the gradients. We use Binary Cross-Entropy (BCE) loss for the LLRs and Mean Squared Error (MSE) loss for the MIMO channel estimates. For a given estimate of LLRs ℓ, and for every training data sample n_{TTI} processed by MDX (every TTI), the BCE loss function is defined as

$$\mathcal{J}^{\text{BCE}}_{n_{\text{TTI}}}(\ell) = -\frac{1}{|\mathcal{D}|N_{TX}\prod_{n_{TX}=1}^{N_{TX}} B_{n_{TX}}} \sum_{n_{TX}=1}^{N_{TX}} \sum_{(f,s)\in\mathcal{D}} \sum_{b=1}^{B_{n_{TX}}} \Big(\ell_{b,f,s,n_{TX}} \log_2(\hat{p}_{b,f,s,n_{TX}}) + (1-\ell_{b,f,s,n_{TX}}) \log_2(1-\hat{p}_{b,f,s,n_{TX}})\Big), \tag{21}$$

where $\hat{p}_{b,f,s,n_{TX}} = 1/(1+\exp(-\ell_{b,f,s,n_{TX}}))$, $B_{n_{TX}}$ is the number of bits per symbol for the n_{TX}-th MIMO layer, and $|\mathcal{D}|$ represents the cardinality of the set $\mathcal{D}$. Similarly, a MSE loss is defined as

$$\mathcal{J}^{\text{MSE}}_{n_{\text{TTI}}}(\hat{\mathbf{H}}_{f,s}) = \frac{1}{|\mathcal{D}||\mathcal{L}|} \sum_{(f,s)\in\mathcal{D}} \|\hat{\mathbf{H}}_{f,s} - \mathbf{H}_{f,s}\|_F^2, \tag{22}$$

where $\hat{\mathbf{H}}_{f,s}$ is an estimate of the MIMO channel, and $\|\cdot\|_F$ is the Frobenius norm. In addition to using BCE loss on the final estimate of LLRs in (20), we also put a BCE loss on an intermediate estimate of the LLRs. The intermediate estimate of LLRs are

$$\ell^{\text{DA-LS}}_{b,f,s,n_{TX}} = \text{Demapper}_{1,\mathbf{1}}\left(\hat{\mathbf{x}}^{\text{DA-LS}}_{f,s}(n_{TX}), \hat{\boldsymbol{\sigma}}^{\text{DA-LS}}_{f,s}(n_{TX})\right), \tag{23}$$

where the subscripts "$1,\mathbf{1}$" indicate that all learnable parameters of the demapping function are fixed to 1, making them non-trainable. Then, the overall loss function can be defined over a training batch of size N_{TTI} as

$$\mathcal{J} = \frac{1}{N_{\text{TTI}}} \sum_{n_{\text{TTI}}=1}^{N_{\text{TTI}}} \log_2(1+\text{snr}_{n_{\text{TTI}}}) \Big(\mathcal{J}^{\text{BCE}}_{n_{\text{TTI}}}(\ell^{\text{D}}) + \mathcal{J}^{\text{BCE}}_{n_{\text{TTI}}}(\ell^{\text{DA-LS}}) + \lambda\cdot\mathcal{J}^{\text{MSE}}_{n_{\text{TTI}}}(\hat{\mathbf{H}}^{\text{NN}}_{f,s})\Big), \tag{24}$$

where $\text{snr}_{n_{\text{TTI}}}$ is the linear SNR of the n_{TTI}-th training sample, and the multiplier λ is a hyperparameter that weights the MSE loss. Note that the intermediate LLRs in (23) are computed only in the training phase.

Algorithm 1 ResBlocks Processing

Inputs: $\hat{H}^{\text{PA-LS}}_{f,s}, \hat{H}^{\text{DA-LS}}_{f,s}$ $\forall(f,s)\in\mathcal{RG}$, $\mathbf{P}_{n_{TX}}$ $\forall 1\le n_{TX}\le N_{TX}$, N.
for $(n_R,n_{TX}) \in \mathcal{L}$ **do**
  Create $\mathbf{A}_1$, $\mathbf{B}_1$ ▷ (15), (16)
  $\mathbf{P} \leftarrow$ Repeat $\mathbf{P}_{n_{TX}}$ on first dimension ▷ $F\times S\times 4$
  for $l \leftarrow 1$ to N **do**
    $\mathbf{X} \leftarrow \text{Concat}(\mathbf{A}_l, \mathbf{B}_l)$ ▷ $F\times S\times 4$
    $\mathbf{X} \leftarrow \text{ReLU}(\text{BN}(\mathbf{X}))$ ▷ $F\times S\times 4$
    $\mathbf{X} \leftarrow \text{Concat}(\mathbf{X}, \mathbf{P})$ ▷ $F\times S\times 8$
    $\mathbf{X} \leftarrow \text{Conv2D}(\text{Conv2DSep}(\mathbf{X}))$ ▷ $F\times S\times 8$
    $\mathbf{X} \leftarrow \text{ReLU}(\mathbf{X})$ ▷ $F\times S\times 8$
    $\mathbf{A}_{l+1} \leftarrow \text{Conv2D}(\text{Conv2DSep}(\mathbf{X}))$ ▷ $F\times S\times 2$
    $\tilde{\mathbf{\Gamma}}_l \leftarrow$ Repeat and Broadcast $\mathbf{\Gamma}_l$ ▷ $F\times S\times 2$
    $\mathbf{A}_{l+1} \leftarrow \mathbf{A}_l + \tilde{\mathbf{\Gamma}}_l \cdot \mathbf{A}_{l+1}$ ▷ $F\times S\times 2$
    if $l < N$ **then**
      $\mathbf{B}_{l+1} \leftarrow \text{Conv2D}(\text{Conv2DSep}(\mathbf{B}_{l+1}))$ ▷ $F\times S\times 2$
      $\mathbf{B}_{l+1} \leftarrow \mathbf{B}_l + \mathbf{B}_{l+1}$ ▷ $F\times S\times 2$
    end if
  end for
  $\mathbf{C}_{n_R,n_{TX}} \leftarrow \text{Complex}(\mathbf{A}_{N+1})$ ▷ $\mathbf{C}_{n_R,n_{TX}} \in \mathbb{C}^{F\times S}$
end for
Output: $\mathbf{C}_{n_R,n_{TX}}$, $\forall(n_R,n_{TX}) \in \mathcal{L}$

IV. Simulation Results

We compare the performance of MDX with the SoA model from [7], [8]. In addition, three baseline methods are used for comparison: Least Squares (LS) channel estimation at pilot locations with linear interpolation to data symbols followed by LMMSE equalization, LMMSE channel estimation with K-best detection, and perfect channel knowledge with K-best detection. For more details on these baselines see [7], [8], [17].

A. Simulation Setup

We use the Adaptive Moment Estimation (ADAM) optimization with a learning rate of lr = 0.001, and $\lambda = 0.01$ to train our MDX. The end to end MIMO transmission is simulated in Sionna [17]. As in [8], the training is done on the 3GPP Urban Microcell (UMi) channel model, with random drops of users for each training data sample n_{TTI}, ensuring randomized power delay profiles, angle of arrival and departure. Also, the number of users, and their speeds are randomized: the number of active users is drawn from a triangular distribution, and the user speeds from a uniform distribution in $[0, 56]$ m/s. The model was trained on 3 different modulation orders namely 4-QAM, 16-QAM, and 64-QAM with different LDPC code rates corresponding to 5G Modulation Coding Schemes (MCSs) indices $I_{\text{MCS}} = \{9, 14, 19\}$ described in [18, Table 5.1.3.1-1].

All benchmarked methods employed an identical PUSCH configuration: 4 PRBs for training and 273 for evaluation, with 2.14 GHz carrier frequency, 30 kHz subcarrier spacing. The DMRS configurations consist of 2 CDM groups, with pilot symbols allocating OFDM symbols 2, and 11. The evaluations are performed on MCS indices I_{MCS} and a TDL-A channel model [19], with Doppler shifts and RMS delay spreads uniformly drawn from the intervals $[0, 325]$ Hz and $[10, 300]$ ns, respectively. The number of filters for ResBlocks of MDX is set to 8. All results in this paper were obtained using a single pre-trained MDX model without additional training or fine-tuning. The MDX model was trained on an NVIDIA GeForce GTX 1080 Ti GPU for 500,000 iterations with a batch size of $N_{\text{TTI}} = 128$.

B. MU-MIMO 4×2

We evaluated the receivers on a MU-MIMO PUSCH scenario with 2 active layers ($N_{TX} = 2$), and 4 receiver antenna elements ($N_R = 4$). Fig. 4 shows the TBLER performance vs Signal-to-Noise Ratio (SNR) values, for varying MCS indices in I_{MCS}. For this evaluation we use the NRX model that supports varying MCS by a masking method. We use the weights open sourced in [7]. Our model outperforms the LS channel estimation baseline across all three modulation orders and achieves performance comparable to LMMSE channel estimation baseline at 16-QAM and 64-QAM. While MDX surpasses NRX at 64-QAM, NRX performs better at QPSK, albeit at the cost of significantly higher computational complexity.

C. MU-MIMO 16×4

Fig. 5 shows the evaluation results of the receivers on a MU-MIMO PUSCH scenario with a varying number of active layers (maximum $N_{TX} = 4$), 16 receiver antenna elements ($N_R = 16$), and for MCS indices in I_{MCS}. Here we use the same MDX model from previous evaluation (trained on MU-MIMO 4×2), without further training.

Using the source code from [7], [8], we trained a separate NRX model for each of the three MCS indices in I_{MCS}. We adopted the parameterization (e.g., filter counts, iterations) from [8], and set the unspecified $d_s = 64$. Each model was trained on Swiss National Supercomputing Centre (CSCS) resources (NVIDIA GH200 GPUs) for several million iterations with a batch size of 128.

Our model outperforms the baseline with LS channel estimation as well as NRX across all tested modulation orders and number of active layers. It performs close to the LMMSE channel estimation baseline at 16-QAM modulation, and approaches perfect channel knowledge at 64-QAM.

D. Complexity Analysis

The complexity of MDX is dominated by two LMMSE blocks, and ResBlocks processing (mostly depth-wise separable convolutional neural networks). The NRX model uses compute-expensive depthwise separable convolutional layers, and fully connected layers. We count one complex-valued multiplication as four real-valued multiplications. The number of real valued multiplications for our LMMSE block is $2N_{TX}^3 + 6N_R N_{TX}^2 + 6N_R N_{TX} - 2N_{TX} + 2$ per RE. For the pair of a depthwise separable convolution followed by a point-wise convolution (with stride set to 1) we count $k^2 N_0 FS + N_0 NFS$ multiplications (k, N_0, and N denote the filter size, number of input feature maps, and number of output feature maps). The complexities, in FLOPs, along with the number of parameters for the MDX and NRX models, configured as for the TBLER evaluation, are in Table II. In the 4×2 MU-MIMO configuration, our MDX model requires 106× fewer FLOPs and 157× fewer parameters than the NRX model. For the 16×4 MU-MIMO configuration, these reductions are 66× and 396×, respectively.

TABLE II
FLOPs and Parameters

MIMO	Model	FLOPs(G)	Params(k)	NRX/MDX
4 × 2	MDX	0.7	2.7	**106**× (FLOPs)
	NRX	78.6	431.2	**157**× (Params)
16 × 4	MDX	6	2.7	**66**× (FLOPs)
	NRX	397.6	1,088.4	**396**× (Params)

V. Conclusion

This paper presents a 5G NR PUSCH receiver that integrates conventional PHY blocks with learnable parameters and ResBlocks processing. By employing a model-driven approach, we substantially lowered the receiver's computational complexity (by 66×) and memory requirements (by 396×), while surpassing the performance of both classical baselines and SoA solutions. Additionally, we simplified the training phase, which is essential for online learning using site-specific data captured from the actual radio environment at the edge base stations. On the other hand, our receiver processes each MU-MIMO link independently using shared parameters

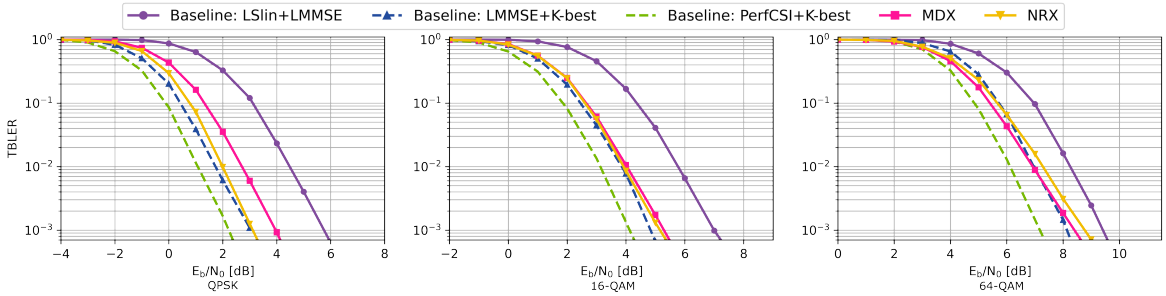

Fig. 4. TBLER vs. SNR performance with $N_{TX} = 2$ active layers, and $N_R = 4$ receiving antenna elements for a 3GPP TDL-A channel.

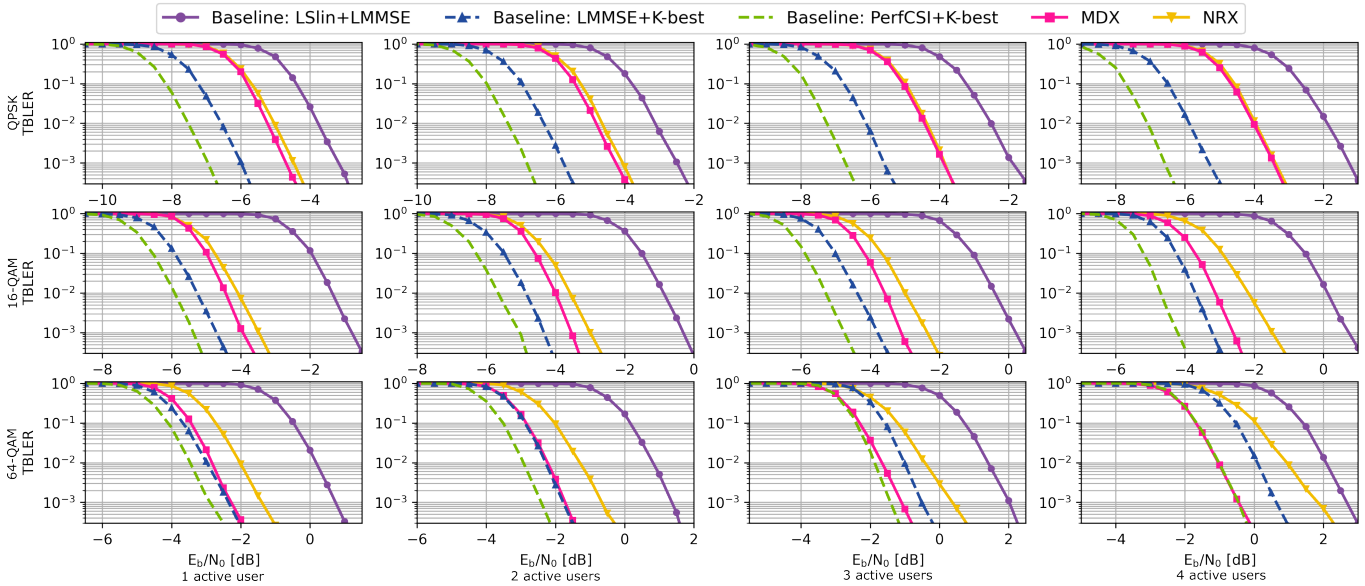

Fig. 5. TBLER vs. SNR for varying MCS indices and number of active layers in a 3GPP TDL-A channel. Columns represent the number of active layers, ranging from 1 (left) to 4 (right). Rows correspond to MCS indices: 9 (top), 14 (middle), and 19 (bottom).

and depthwise separable convolutions, unlocking significant potential for parallel processing. Furthermore, it demonstrates greater flexibility than existing SoA solutions, supporting a wide range of PUSCH MU-MIMO configurations—including variations in MIMO layers, receiver antennas, bandwidths, and modulation orders—using a single pre-trained model. The MDX implementation along with trained weights will be open-sourced to ensure reproducibility [14].

References

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Mag. NET*, vol. 34, no. 3, pp. 134–142, 2019.

[2] X. Lin, "Artificial Intelligence in 3GPP 5G Advanced: A Survey," *IEEE ComSoc*, 2023. [Online]. Available: https://www.comsoc.org/publications/ctn/artificial-intelligence-3gpp-5g-advanced-survey

[3] E. Björnson, J. Hoydis, L. Sanguinetti *et al.*, "Massive MIMO networks: Spectral, energy, and hardware efficiency," *Foundations and Trends in Signal Processing*, vol. 11, no. 3-4, pp. 154–655, 2017.

[4] C. Zhang, Y.-L. Ueng, C. Studer, and A. Burg, "Artificial intelligence for 5G and beyond 5G: Implementations, algorithms, and optimizations," *IEEE J. Emerg. Sel. Top. Circ. Syst.*, vol. 10, no. 2, pp. 149–163, 2020.

[5] M. Honkala, D. Korpi, and J. M. Huttunen, "Deeprx: Fully convolutional deep learning receiver," *IEEE J. WCOM*, vol. 20, no. 6, pp. 3925–3940, 2021.

[6] D. Korpi, M. Honkala, J. M. Huttunen, and V. Starck, "Deeprx MIMO: Convolutional MIMO detection with learned multiplicative transformations," in *IEEE Conference on ICC*, 2021, pp. 1–7.

[7] R. Wiesmayr, S. Cammerer, F. A. Aoudia, J. Hoydis, J. Zakrzewski, and A. Keller, "Design of a standard-compliant real-time neural receiver for 5G NR," 2024. [Online]. Available: https://arxiv.org/abs/2409.02912

[8] S. Cammerer *et al.*, "A neural receiver for 5G NR multi-user MIMO," in *IEEE GLOBECOM Workshops*, Dec. 2023, pp. 329–334.

[9] "Data Processing Units, Empowering 5G carrier, enterprise and AI cloud data infrastructure." [Online]. Available: https://www.marvell.com/products/data-processing-units.html

[10] Y. Zhang, M. Bertuletti, S. Riedel, A. Vanelli-Coralli, and L. Benini, "TeraPool-SDR: An 1.89TOPS 1024 RV-Cores 4MiB Shared-L1 Cluster for Next-Generation Open-Source Software-Defined Radios," in *IEEE GLSVLSI*, 2024, pp. 86–91.

[11] "How we Won the Acceleration Architecture Debate." [Online]. Available: https://www.qualcomm.com/news/onq/2023/03/how-we-won-the-acceleration-architecture-debate

[12] Nvidia Corporation, "NVIDIA H100 Tensor Core GPU," 2024. [Online]. Available: https://www.nvidia.com/en-us/data-center/h100/

[13] Intel Corporation, "Intel Architecture Day 2021 Presentation: Alder Lake, Sapphire Rapids, Ponte Vecchio, Xe-HPG," Presentation and Briefing Materials, 2021. [Online]. Available: https://www.anandtech.com/show/16881/intel-architecture-day-2021-a-sneak-peek-at-the-next-decade/

[14] [Online]. Available: https://github.com/Mahdi-Abdollahpour/mdx

[15] 3GPP, "Physical channels and modulation," TS 38.211, March 2025, version 18.6.0.

[16] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conference on CVPR*, July 2017, pp. 1251–1258.

[17] J. Hoydis *et al.*, "Sionna: An open-source library for next-generation physical layer research," 2023. [Online]. Available: https://arxiv.org/abs/2203.11854

[18] 3GPP, "Physical layer procedures for data," TS 38.214, March 2025, version 18.6.0.

[19] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," Tech. Rep. TR 38.901, March 2024, version 18.0.0.