

Towards AI-Native Fronthaul: Neural Compression for NextG Cloud RAN

Chenghong Bian, *Student Member, IEEE*, Yulin Shao, *Member, IEEE*, Deniz Gündüz, *Fellow, IEEE*

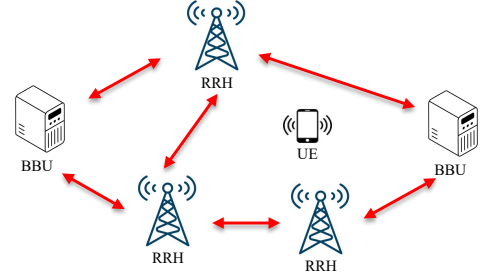
Abstract—The rapid growth of data traffic and the emerging AI-native wireless architectures in NextG cellular systems place new demands on the fronthaul links of Cloud Radio Access Networks (C-RAN). In this paper, we investigate neural compression techniques for the Common Public Radio Interface (CPRI), aiming to reduce the fronthaul bandwidth while preserving signal quality. We introduce two deep learning-based compression algorithms designed to optimize the transformation of wireless signals into bit sequences for CPRI transmission. The first algorithm utilizes a non-linear transformation coupled with scalar/vector quantization based on a learned codebook. The second algorithm generates a latent vector transformed into a variable-length output bit sequence via arithmetic encoding, guided by the predicted probability distribution of each latent element. Novel techniques such as a shared weight model for storage-limited devices and a successive refinement model for managing multiple CPRI links with varying Quality of Service (QoS) are proposed. Extensive simulation results demonstrate notable Error Vector Magnitude (EVM) gains with improved rate-distortion performance for both algorithms compared to traditional methods. The proposed solutions are robust to variations in channel conditions, modulation formats, and noise levels, highlighting their potential for enabling efficient and scalable fronthaul in NextG AI-native networks, aligning with the current 3GPP research directions.

Index Terms—C-RAN, AI-native fronthaul, CPRI compression, vector quantization, VQ-VAE, variable-rate compression.

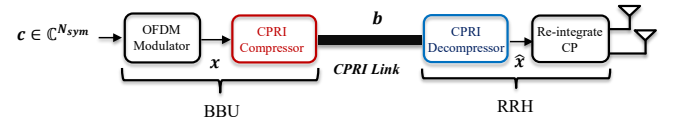
I. INTRODUCTION

As mobile networks evolve toward the 6G and NextG era, characterized by ubiquitous intelligence and ultra-dense deployments, the pressure on fronthaul capacity becomes increasingly critical. In particular, the demand for real-time, high-throughput transmission between distributed radio units and centralized baseband processing units, typified in Cloud Radio Access Network (C-RAN) architectures, poses significant challenges [1]. C-RAN enhances spectral and energy efficiency by decoupling Remote Radio Heads (RRHs) from Baseband Units (BBUs), but relies on high-speed fronthaul links, such as the Common Public Radio Interface (CPRI) [2], [3], to transport raw physical-layer signals. Without effective compression, this data exchange can quickly overwhelm fronthaul resources, hindering scalability and performance in future-generation systems. Thus, CPRI compression emerges as a key enabler for scalable, efficient C-RAN deployments aligned with 6G and NextG network design.

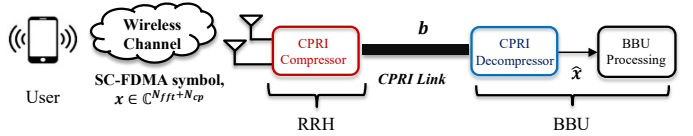
In advancing the efficiency of signal transmission over the CPRI link, significant strides have been made through various



(a) The connection of RRHs to BBUs via CPRI links within a C-RAN architecture.



(b) The processing of the BBU and RRH in the downlink scenario.



(c) The uplink scenario.

Fig. 1: Illustration of the CPRI link for both downlink and uplink scenarios.

research efforts. At the core of these innovations is the development of sophisticated CPRI compression techniques aimed at enhancing transmission efficiency and signal quality. The foundational approach [4] focuses on exploiting the inherent redundancy within time-domain signals, wherein the authors devise a method incorporating multi-rate filtering to pare down the numbers of samples to be transmitted. This process is complemented by the application of non-uniform scalar quantization, effectively translating each I/Q sample into a compressed bit representation, marking a pivotal step forward in CPRI link optimization.

Building upon this groundwork, [5] introduces a vector quantization scheme. By organizing the input sequence into vectors, and mapping these to the nearest codewords within a specially curated codebook – derived through a modified Lloyd algorithm – this method significantly improves compression efficiency. Additionally, the integration of entropy coding further enhances this system, yielding better reconstruction performance and a notable advancement over the scalar quantization approach. A filter with an error-feedback loop is deployed in [6], resulting in

C. Bian and D. Gündüz are with the Department of Electrical and Electronic Engineering, Imperial College London (E-mails: {c.bian22, d.gunduz}@imperial.ac.uk).

Y. Shao is with the Department of Electrical and Electronic Engineering, The University of Hong Kong (E-mail: yshao@hku.hk).

even more impressive compression performance. This approach relies on element-wise quantization of filter's output signal, which ingeniously shifts quantization noise away from OFDM subcarriers carrying vital information to those that do not, thereby improving reconstruction performance.

Expanding the scope of CPRI compression, researchers have also explored its application in distributed systems [7], [8] and multiple-input multiple-output (MIMO) scenarios [9]. These studies underscore the versatility and critical importance of CPRI compression techniques in refining the functionality and efficiency of C-RAN systems, setting a precedent for future innovations in 5G-Advanced and beyond.

The integration of deep learning (DL) into wireless communications in recent years has heralded a significant paradigm shift, demonstrating its efficacy in enhancing various aspects of the field, including signal processing [10], [11], source coding [12], [13], channel coding [14], [15], and joint source-channel coding [16]. This pioneering research has effectively leveraged domain-specific insights to refine deep neural network (DNN) architectures and their training methodologies, underscoring the versatility and power of deep learning technologies in addressing complex communication challenges.

Driven by the substantial advancements facilitated by DL in wireless communications, researchers have developed machine learning based approach for CPRI compression [8], [17], [18]. In particular, the authors in [8] consider a scenario where the user is served by a central BBU cooperatively via multiple RRHs. Different transformation matrices at the RRHs are learned to perform dimension reduction/expansion followed by uniform quantization. In [17], the authors optimize the CPRI compression problem using a deep reinforcement learning approach under the latency and packet loss constraints. In our case, we consider the user is served by the BBU connected to a single RRH and utilize the EVM as the performance metric. The most relevant work is [18] which applies a DNN for non-linear transformation on a per-element basis, essentially functioning as a non-uniform scalar quantizer and achieves only marginal improvements over traditional methods, such as those described in [4].

In this paper, we present two innovative neural compression schemes which extend and surpass the work identified in [18]. These schemes are designed to efficiently reduce the bit rate over CPRI links while ensuring high-quality signal reconstruction at the receiving end. Specifically, both proposed schemes employ a non-linear transformation, executed through a recurrent neural network (RNN), to process the input signal into a compact latent vector. The first scheme applies scalar or vector quantization to this latent vector on an element-wise or block-wise basis, respectively. The second scheme, on the other hand, rounds each element of the latent vector to the nearest integer value, and utilizes a specially designed probability prediction network to estimate the likelihood of each quantized element's occurrence. Then, an arithmetic encoder translates these probabilities into a compact bit sequence for efficient CPRI link transmission.

In addressing the practical deployment challenges of our proposed neural compression scheme, we focus on two critical scenarios: 1) the presence of multiple CPRI links between the RRH and BBU, and 2) the existence of storage limitations on

these devices. Each scenario presents unique requirements and opportunities for optimization.

- 1) In the first scenario, the CPRI links exhibit varied levels of reliability (as indicated by their packet error rates) and successive refinement transmission is a natural solution – the bit sequence for a coarse reconstruction is transmitted over the most reliable CPRI link, while supplementary refinement data is scheduled over less reliable links. To effectively implement this strategy, we have developed a specialized neural network architecture tailored for successive refinement. This design not only accommodates the diverse reliability of CPRI links but also ensures high-quality signal reconstruction.
- 2) In the second scenario, where storage capacity on the devices is constrained, we propose a novel solution through the development of a unified compression model. This model is adept at navigating different rate-distortion (R-D) trade-offs, a capability previously explored in variable-rate image compression research [19]–[22]. Our architecture draws inspiration from significant works in the field, yet it distinguishes itself by its efficiency, specifically tailored for CPRI compression. This efficiency is pivotal in achieving the desired storage reduction without compromising on the quality of compression, thereby addressing a crucial need within future wireless communication systems.

Our main contributions are summarized as follows:

- We present two AI-native neural compression schemes that substantially advance traditional CPRI compression techniques, providing a principled framework for fronthaul bit-rate reduction in NextG Cloud RAN architectures. By integrating non-linear transformation functions and employing unique latent quantization techniques, these schemes enhance the R-D performance, demonstrating a critical leap forward in compression efficiency.
- Through a statistical analysis of input signal distributions, we uncover the inherent correlation among signal elements, a revelation that challenges the adequacy of previous approaches and motivates our use of non-linear transformations. This insight directly informs the design of a successive refinement compression model tailored for environments with CPRI links of varying reliability, as well as a unified model that adeptly navigates storage constraints without sacrificing performance.
- Extensive numerical experiments validate the superior performance of our proposed schemes over traditional benchmarks, particularly in terms of error vector magnitude (EVM). Additionally, we demonstrate the robustness of our models under diverse conditions, including mismatched signal-to-noise ratios (SNRs), modulation formats, and filter taps. The practical efficacy of both the unified model for storage optimization and the successive refinement strategy for leveraging link reliability further highlights the viability of our design for robust 6G fronthaul standardization.

Notations: Throughout the paper, scalars are represented by normal-face letters (e.g., x), while uppercase letters (e.g., X) represent random variables. Matrices and vectors are denoted by bold upper and lower case letters (e.g., \mathbf{X} and \mathbf{x}), respectively. Transpose and Hermitian operators are denoted by $(\cdot)^\top$, $(\cdot)^\dagger$,

respectively. $\Re(x)$ ($\Im(x)$) denotes the real (imaginary) part of a complex variable x . $\lceil x \rceil$ ($\lfloor x \rfloor$) denote the nearest integer which is larger (smaller) than x . We denote by $|x|$ and L_x the magnitude and length of a vector x , respectively. We will also use the notation $[N]$ to denote the set of positive integers up to N , i.e., $[N] = \{1, 2, \dots, N\}$.

II. PROBLEM FORMULATION

A. System model

This section presents the model for signal compression over the CPRI link within the C-RAN architecture, addressing both uplink and downlink scenarios, as shown in Fig. 1. The process begins with the generation of an information bit sequence. This bit sequence undergoes channel coding and modulation to produce Quadrature Amplitude Modulation (QAM) symbols, denoted by $c \in \mathbb{C}^{N_{sym}}$, where N_{sym} denotes the number of QAM symbols.

In the uplink scenario, mobile users utilize Single Carrier-Frequency Division Multiple Access (SC-FDMA): the QAM symbols, c , are first converted to the frequency domain by an N_{sym} -point discrete Fourier transform (DFT). The resulting signal vector is then zero-padded to expand it into an N_{fft} -dimensional vector, which is subsequently transformed back to the time domain via an N_{fft} -point inverse DFT (IDFT). A Cyclic Prefix (CP) consisting of N_{cp} elements is added to this signal to combat the effects of multipath propagation and maintain subcarrier orthogonality across the transmission link. After passing through the uplink channel, the signal received at the RRH, denoted by $x \in \mathbb{C}^{N_{fft}+N_{cp}}$, will be forwarded to the CPRI compression module for efficient transmission over the CPRI link.

In the downlink scenario, the BBU directly assigns the QAM symbols c to the OFDMA subcarriers, leaving $N_{fit} - N_{sym}$ unoccupied subcarriers as a guard band. Then, the QAM symbols are transformed to the time domain using an N_{fit} -point IDFT. Typically, the BBU appends a CP to these symbols before feeding them into the CPRI compression module for transmission to the RRH. However, given the absence of channel noise in the downlink scenario, the CP merely replicates the final N_{cp} elements of an OFDM symbol. Leveraging this redundancy, we can improve the compression efficiency by removing the CP prior to compression. This modification yields a CP-less signal, represented as $x \in \mathbb{C}^{N_{fft}}$, that undergoes compression within the CPRI compression module. The CP will be re-integrated at the RRH, ensuring the signal's integrity during transmission while enhancing the overall compression efficiency over the CPRI link.

B. CPRI compression

The CPRI compression module at the RRH (BBU) for uplink (downlink) compression consists of four principal modules: decimation, block scaling, non-linear transformation, and source encoding. The comprehensive workflow of the CPRI compression process is illustrated in Fig. 2. In the following, we delve into the specifics of each module.

1) *Decimation*: The decimation module is introduced to reduce the time domain redundancy caused by the guard band which carries no information. To this end, standard multi-rate filter is adopted. We first upsample the input signal, x , by a factor of K . This is achieved by inserting zeros among samples:

$$x^u(j) = \begin{cases} x(n), & j = Kn; \\ 0, & j \neq Kn, \end{cases} \quad (1)$$

and passing x^u through a low-pass filter to preserve the spectrum of interest while discarding the others, as zero insertion in the time domain leads to a periodic replication in the frequency domain. Then, the filtered signal is downsampled by M times, where $M > K$ such that the number of samples is reduced after the upsampling and downsampling processes. The output of the Decimation module, denoted by $x' \in \mathbb{C}^{N'}$, where $N' = (N_{fft} + N_{cp})K/M$ and $N' = N_{fft}K/M$ for the uplink and downlink scenarios, respectively.

2) *Block Scaling*: The block scaling module scales the input signal x' to limit the maximum value. To this end, x' is first partitioned into multiple blocks where each block is comprised of N_s complex symbols. For the k -th block, we first determine the scaling factor, t_k , as follows:

$$t'_k = \max_{i \in [kN_s, (k+1)N_s)} \{|\Re(x'(i))|, |\Im(x'(i))|\},$$

$$t_k = \begin{cases} \lceil t'_k \rceil, & \lceil t'_k \rceil \leq 2^{Q_s} - 1, \\ 2^{Q_s} - 1, & \lceil t'_k \rceil > 2^{Q_s} - 1. \end{cases} \quad (2)$$

Note that t_k is a positive integer, and can be represented using Q_s bits. The corresponding scaled output, $s = [s_1^\top, \dots, s_{N_t}^\top]^\top$, is given by

$$s_k = x'_k / t_k, \quad (3)$$

where x'_k is the k -th block of x' and $N_t \triangleq \lceil \frac{N'}{N_s} \rceil$.

3) *Transformation and source encoding*: We next transform the scaled data s into a bit sequence. Existing methods in CPRI compression adopt scalar quantization or vector quantization approaches:

- *Scalar quantization* [4]. In this approach, each element of the input signal s is quantized individually. A one-dimensional quantization codebook, $\mathcal{C} \in \mathbb{R}^{2^{Q_s}}$ with 2^{Q_s} levels, is trained via an iterative algorithm [23]. The I/Q components of the quantized symbols $\hat{s}(i)$ are given by

$$\Re(\hat{s}(i)) = \arg \min_n |\mathcal{C}_n - \Re(s(i))|,$$

$$\Im(\hat{s}(i)) = \arg \min_n |\mathcal{C}_n - \Im(s(i))|. \quad (4)$$

Each I/Q component is then represented using Q bits.

- *Vector quantization* [5]. Instead of quantizing s element-wise, vector quantization organizes the input signal s into blocks, where each block consists b I/Q components. A specialized codebook $\mathcal{C} \in \mathbb{R}^{2^{bQ} \times b}$, where each vector component is quantized using Q bits, is generated using a modified Lloyd algorithm [5]. This method not only retains the advantages of scalar quantization but also reduces redundancy significantly by considering the statistical dependencies among multiple components. To further improve the compression efficiency, this method

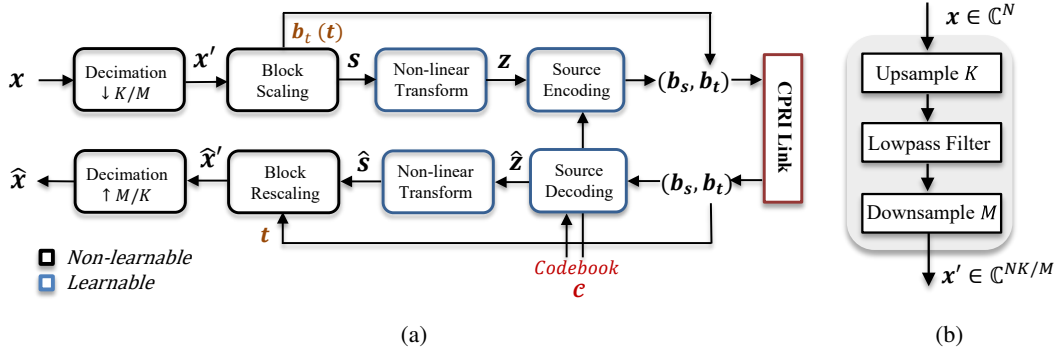


Fig. 2: (a) The flowchart of the unified CPRI compression procedure. In particular, the non-linear transformation only applies to the proposed neural compression algorithms and is parameterized by neural networks. The codebook, \mathcal{C} , can be obtained either by clustering (i.e., in the conventional schemes) or learning (i.e., the proposed latent quantization and neural compression schemes). (b) Details of the decimation module, where the input is a N -dimensional vector.

assigns different number of bits to different entries of the codebook, as opposed to uniform bit assignment, where each entry of the codebook is represented by bQ bits. This can be achieved by variable-length source coding algorithms, taking the frequency of entries into account.

The output of the transformation and source encoding modules is a bit sequence, b_s , representing the quantized version of s . This sequence is combined with b_t , the bit representation of the scaling vector $t \triangleq [t_1, \dots, t_{N_t}]$ introduced in (2), to form the complete bit sequence b , which is then transmitted over the CPRI link. The compression ratio (CR) for this process is calculated as follows

$$\text{CR} = \frac{L_b}{2 \times 15 \times N'} = \frac{K(2\alpha Q N' + Q_s N_t)/M}{30 N'}. \quad (5)$$

Here, the denominator represents the total number of bits required to encode the original signal without compression, where 2 accounts for the I and Q components, and 15 refers to the bit precision per I/Q component by convention [4]. The numerator represents the total bits in the compressed signal b , where

- $\alpha \in (0, 1]$ denotes the reduction of bit cost from the source coding algorithm.
- Q is the number of bits per quantized I/Q component.
- Q_s is the number of bits used to represent the scaling factor.
- N_t is the total number of scaling factors applied.
- N' is the sequence length, adjusted for both uplink and downlink scenarios.

4) *Decoder*: The CPRI decompression process is a direct mirror of the compression sequence. Upon reception of the bit sequence b from the CPRI link, the decoder segments it into b_t and b_s , which are used to reconstruct the scaling vector t and the quantized signal \hat{s} , respectively. The block rescaling operation is conducted as follows

$$\begin{aligned} \hat{x}'_k &= \hat{s}_k t_k, \\ \hat{x}' &= [\hat{x}'_1, \dots, \hat{x}'_{N_t}]^T. \end{aligned} \quad (6)$$

Finally, we adopt a multi-rate filtering process where \hat{x}' is first upsampled by a factor of M , filtered to retain only the necessary spectral components, and then downsampled by K ,

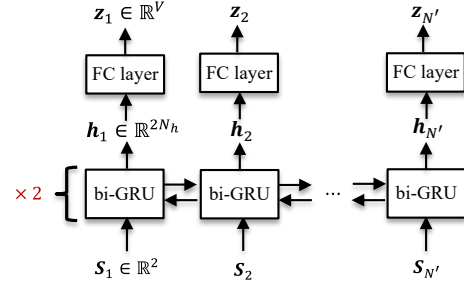


Fig. 3: The neural network architecture of the non-linear transformation function, $f_s(\cdot)$ at the encoder which transforms input signal S to the latent vector, z .

resulting in the final time-domain reconstruction, \hat{x} . Note that in the downlink scenario, a CP is added to \hat{x} before transmission through the channel. In the uplink scenario, synchronization is necessary to locate the start of the SC-FDMA frame, followed by the removal of CP from \hat{x} to prepare for further processing.

As highlighted in [5], instead of assessing the performance of CPRI compression algorithms solely in the time domain, it is crucial to evaluate the quality of signal reconstruction in the frequency domain, where the actual data transmission occurs. The frequency-domain representations of the original and reconstructed signals, denoted by x_f and \hat{x}_f , respectively, are obtained by performing a DFT on their time-domain counterparts. The error vector magnitude (EVM) is defined as

$$\text{EVM}(\%) = \sqrt{\frac{\sum_{n \in \mathbb{O}} |\hat{x}_f(n) - x_f(n)|^2}{\sum_{n \in \mathbb{O}} |x_f(n)|^2}} \times 100\%, \quad (7)$$

where \mathbb{O} denotes the set of occupied subcarrier indices, with a cardinality equal to N_{sym} . The EVM can also be expressed in decibels (dB):

$$\text{EVM (dB)} = 10 \log_{10} \left(\frac{\sum_{n \in \mathbb{O}} |x_f(n)|^2}{\sum_{n \in \mathbb{O}} |\hat{x}_f(n) - x_f(n)|^2} \right). \quad (8)$$

III. NEURAL CPRI COMPRESSION WITH LATENT QUANTIZATION

A. Motivation

The non-linear transformation and source encoding modules form the cornerstones of CPRI compression. As detailed in Section II-B, traditional solutions use a scalar or vector quantization scheme. However, both schemes rely on a unified codebook, wherein all the elements or blocks of elements from the input signal s are quantized in accordance with predefined levels. These schemes function by directly applying quantization to block-scaled data s . Such methods achieve optimal results under the assumption that the elements or blocks being quantized are independent and identically distributed (i.i.d.). However, this assumption is invalid by analyzing the output of the decimation module, as demonstrated in Proposition 1.

Proposition 1. *The output of the decimation module, x' , exhibits zero mean, i.e., $\mathbb{E}(x'_n) = 0$. The covariance for the elements of x' is given by:*

$$\text{Cov}(x'_n, x'_m) = \frac{P}{N'} \sum_{k=k^*}^{N'-k^*-1} e^{j\beta(n-m)k}, \quad (9)$$

$\forall n, m \in [N'], \beta \triangleq \frac{2\pi}{N'}$, where $k^* > 0$ represents the number of OFDM subcarriers dedicated to the guard band, and P is the average power of the QAM symbols.

Proof. See Appendix A. \square

Proposition 1 reveals that the individual elements (or blocks) generated by the decimation module do not adhere to the i.i.d. assumption, nor does its block-scaled version, s . Therefore, the scalar and vector quantization approaches in [4] and [5] are sub-optimal. To address this suboptimality, our idea is to employ a learnable non-linear transformation, detailed in Section III-B, to transform the scaled data s into a latent space where the elements are more amenable to quantization. Following the transformation, Section III-C discusses the specific techniques used to quantize this neural transformed latent data. Different quantization strategies are explored to optimize the compression of the transformed data while maintaining the integrity and quality of the reconstructed signals.

Building on the above foundation, Section IV advances the discussion by detailing how the compression model employs a probabilistic learning approach. This approach involves predicting the probability distribution of the latent data, enabling more efficient and adaptive compression.

B. Learning-based Non-linear Transformations

Considering that s is a time-domain sequence, we utilize recurrent neural networks (RNNs) to parameterize non-linear transformation functions. Specifically, we employ bi-directional Gated Recurrent Units (bi-GRUs) [24], followed by fully connected layers, to facilitate this transformation.

To illustrate this process using the CPRI compression scenario for uplink (which is similarly applicable to downlink), let us first consider the input signal, $s \in \mathbb{C}^{N'}$. This signal is

initially transformed into a real-valued matrix, $S \in \mathbb{R}^{2 \times N'}$, represented as

$$S = \begin{bmatrix} \Re(s^\top) \\ \Im(s^\top) \end{bmatrix}. \quad (10)$$

Subsequently, S undergoes a non-linear transformation through the function $f_s(\cdot)$, generating a latent representation, $z \in \mathbb{R}^{V N'}$:

$$z = f_s(S). \quad (11)$$

The neural network architecture for the transformation function $f_s(\cdot)$ is depicted in Fig. 3. This structure incorporates a two-layer bi-GRU with each layer having a hidden dimension of N_h . At each time step j , the input to the bi-GRU module, denoted by S_j , is a two-dimensional vector corresponding to the j -th column of S .

The output from the bi-GRU at the j -th step, h_j , is a vector of length $2N_h$, which results from concatenating the outputs of the forward and backward passes, each contributing a vector of length N_h . This concatenated vector h_j is then processed by a fully connected (FC) layer, which transforms it into the latent vector z_j of dimension V . The final latent representation z for the entire sequence is assembled by concatenating each z_j across all time steps:

$$z = [z_1^\top, \dots, z_{N'}^\top]^\top. \quad (12)$$

Given the latent z , we quantize it (detailed later in Section III-C) and obtain the quantized form \hat{z} . The corresponding bit sequence is then transmitted to the destination over the CPRI link. Upon receipt, the destination recovers \hat{z} and feeds it into the latent decoding function, $g_d(\cdot)$, which is designed to reconstruct S . Note that the neural network architecture for $g_d(\cdot)$ mirrors that of $f_s(\cdot)$ with the exception of the input dimension, which is V for $g_d(\cdot)$, as opposed to two.

During the decoding process, for each j -th time slot, \hat{z}_j is fed into the decoder. The bi-GRU then generates a hidden vector \hat{h}_j of size $2N_h$. This vector is subsequently processed by a FC layer that outputs a length-2 vector \hat{s}_j . The complete reconstructed real-valued matrix, \hat{S} , is compiled by stacking these vectors sequentially

$$\hat{S} = [\hat{s}_1, \dots, \hat{s}_{N'}]. \quad (13)$$

The complex reconstructed signal, denoted by \hat{s} , is then formed by taking the first row of \hat{S} as the real component and the second row as the imaginary component.

C. Quantization

In this section, we detail the process of quantizing the latent vector z to derive its quantized counterpart \hat{z} .

1) *Uniform Quantization:* We first consider uniform quantization of the latent vector. We start by clipping each element of the output from the latent encoding function within the range $[0, 1]$:

$$z = \text{clip}(f_s(S), 0, 1). \quad (14)$$

We then assume that each component of z is i.i.d. and follows a uniform distribution, $\mathcal{U}(0, 1)$. This assumption allows us to apply uniform scalar quantization individually to each element, which can be written as

$$\hat{z}_i = \frac{\lfloor (2^Q - 1)z_i + \frac{1}{2} \rfloor}{2^Q - 1}, \quad (15)$$

where Q represents the number of bits per element, and $\lfloor x + \frac{1}{2} \rfloor$ is the rounding operation. Notably, this rounding operation is not differentiable, which complicates the application of gradient descent during training phases.

To overcome this, we employ the straight through estimator (STE) [25], which allows the gradients of \hat{z} to pass through to z , facilitating the update of neural network parameters within the latent encoding function $f_s(\cdot)$:

$$\nabla_z \mathcal{L} \leftarrow \nabla_{\hat{z}} \mathcal{L}, \quad (16)$$

where \mathcal{L} denotes the loss function. As detailed in (7), our focus lies on optimizing the signal reconstruction specifically in the occupied subcarriers. Therefore, the loss function is defined as

$$\mathcal{L}_{uq} = \frac{\|x_f - \hat{x}_f\|_2^2}{\|x_f\|_2^2}. \quad (17)$$

The total number of bits required by the uniform quantization scheme can be written as

$$L_b = VQN' + Q_s \left\lceil \frac{N'}{N_s} \right\rceil. \quad (18)$$

However, a notable limitation of the uniform quantization scheme is the assumption that each element of z adheres to an i.i.d. uniform distribution. This assumption can lead to sub-optimal quantization performance, particularly when the actual distribution of the latent vector z diverges from this assumed uniformity. The implications of this divergence will be further explored and substantiated by the simulation results presented in Section V.

2) *Vector Quantization*: The strong i.i.d. uniform assumption can be relaxed by vector quantization: we partition z into blocks and treat each block as an independent unit that follows a categorical distribution, rather than assuming each element within the blocks is i.i.d. Specifically, we define $z = [z_1, \dots, z_{N_b}]^T$,¹ where each $z_i \in \mathbb{R}^b$ and $bN_b = VN'$. Each block z_i is independently quantized using a learned codebook, a process that mirrors the training approach of the Vector Quantization Variational Autoencoder (VQVAE) [26]. It is crucial to note that our goal diverges from that of the VQVAE. While the VQVAE primarily aims to generate new data samples, our objective is to develop a discrete latent representation that ensures high-quality reconstruction performance. Given a bit budget of Q bits per element, this necessitates a codebook $E \in \mathbb{R}^{2^{bQ} \times b}$, and we detail the training process in the following.

Rather than restricting the elements of z to the interval $[0, 1]$ as required by the uniform quantization method, we enhance vector quantization by selecting the nearest vector from codebook E for each block z_i . The quantization process is given by

$$\begin{aligned} \hat{z}_i &= E_j, \\ \text{s.t. } j &= \arg \min_{k \in [2^{bQ}]} \|z_i - E_k\|_2^2. \end{aligned} \quad (19)$$

where \hat{z}_i is the quantized output of the i -th block.

Similar to the uniform quantization, vector quantization does not inherently support gradient-based optimization due

¹With a slight abuse of notation, here we denote by z_i the individual blocks after partition. This usage differs from (12), wherein z_i specifies the output of the GRU corresponding to the i -th time step.

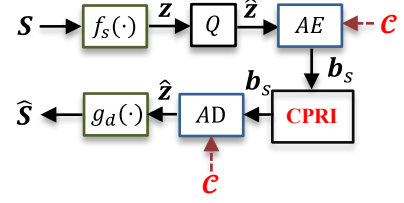


Fig. 4: The pipeline of the proposed neural compression model. The codebook, \mathcal{C} , is constructed using the probability prediction net illustrated in (24).

to its non-differentiable nature. To overcome this, we pass the gradient from the quantized latent \hat{z} back to the original latent z . Optimizing the vector quantization scheme involves tuning both the non-linear transformation functions and the codebook, ensuring effective reconstruction performance. The loss function, integrating vector quantization, is structured as

$$\mathcal{L}_{vq} = \frac{\|x_f - \hat{x}_f\|_2^2}{\|x_f\|_2^2} + \text{sg}[z] - \hat{z}\|_2^2 + \beta \|z - \text{sg}[\hat{z}]\|_2^2, \quad (20)$$

where $\text{sg}[\cdot]$ denotes the stop gradient operator. The first term of the loss function optimizes the non-linear transformation functions $f_s(\cdot)$ and $g_d(\cdot)$. The second term is designed to learn a good codebook by gradually fitting it to the encoder output, z . Since the second term aims to optimize the codebook rather than the latent, we apply the stop gradient operator, $\text{sg}[\cdot]$ to z . The final term is known as the commitment loss, which forces the latent to converge to the learned codebook, E . In this paper, we set $\beta = 1$ as the reconstruction performance is not sensitive with respect to different β values.

This vector quantization strategy reduces the limitations of assuming i.i.d. uniform distribution in latent quantization by allowing for a more nuanced grouping and coding of data blocks. This method can be viewed as an enhancement of traditional vector quantization approaches like those discussed in [5], with the critical difference being that our codebook is derived through learning rather than predefined algorithms such as Lloyd's algorithm. However, it is essential to acknowledge that our assumption about the i.i.d. nature of blocks could still deviate from reality. To address potential discrepancies and further refine our model, we will explore advanced compression frameworks in Section IV.

IV. NEURAL CPRI COMPRESSION WITH PROBABILITY PREDICTION

In the latent quantization approach outlined in (11) and (12), we generate N' vectors $\{z_j\}_{j \in [N']}, z_j \in \mathbb{R}^V$, through a non-linear transformation function $f_s(\cdot)$. These vectors can be assembled into a matrix $Z \in \mathbb{R}^{V \times N'}$:

$$Z = [z_1, \dots, z_{N'}]. \quad (21)$$

From an information-theoretical perspective, if the distribution of Z is known, it is possible to compress the latent vectors losslessly to a rate close to the entropy of Z , denoted by $H(Z)$, using lossless source coding algorithms. However, in practical scenarios where the modulation order, channel responses, and noise levels are unknown, the precise distribution of both s

and \mathbf{Z} remains elusive. To address this, we propose using a neural network, which we refer to as the probability prediction net, parameterized by ϕ , to model the distribution of \mathbf{Z} . This network aims to approximate the fully factorized probability of the latent variables, thus facilitating a more streamlined and efficient compression process.

A. Pipeline

We first overview the pipeline of the proposed neural compression approach with probability prediction introduced in the image compression literature [12], [13]. Initially, each element of the matrix \mathbf{Z} is rounded to the nearest integer, resulting in $\hat{\mathbf{Z}} = \lfloor \mathbf{Z} + \frac{1}{2} \rfloor$. It is important to note that any random variable $Z_{i,j}$ within the range $(\hat{z}_{i,j} - \frac{1}{2}, \hat{z}_{i,j} + \frac{1}{2}]$ will be rounded to the same value $\hat{z}_{i,j}$. Consequently, the probability of the quantized random variable $\hat{Z}_{i,j}$ can be written as

$$\Pr(\hat{Z}_{i,j} = \hat{z}_{i,j}) = P_{\phi_i} \left(\hat{z}_{i,j} + \frac{1}{2} \right) - P_{\phi_i} \left(\hat{z}_{i,j} - \frac{1}{2} \right), \quad (22)$$

where $P_{\phi_i}(\cdot), i \in [V]$ denotes the probability mass function (PMF) for the i -th channel (i.e., the i -th row of the latent \mathbf{Z}), and is parameterized by ϕ_i . The overall learnable parameters ϕ are a collection of the parameters of all V channels, i.e., $\phi \triangleq \{\phi_i\}_{i \in [V]}$. The training methodologies for these parameters are elaborated upon later in Section IV-B.

Subsequently, an arithmetic encoder processes $\hat{\mathbf{Z}}$ into a bit sequence \mathbf{b}_s following the probability model described in (22), which is then transmitted over the CPRI link. The length of \mathbf{b}_s can be closely approximate the predicted entropy of $\hat{\mathbf{Z}}$ to within one bit, as given by

$$L_{\mathbf{b}_s} < - \sum_{i,j} \log_2 \left[P_{\phi_i} \left(\hat{z}_{i,j} + \frac{1}{2} \right) - P_{\phi_i} \left(\hat{z}_{i,j} - \frac{1}{2} \right) \right] + 1. \quad (23)$$

Upon receiving the bit sequence, the arithmetic decoder is applied to reconstruct the latent matrix, $\hat{\mathbf{Z}}$, which is then input into the latent decoding function to produce the final reconstructed signal $\hat{\mathbf{s}}$.

During the deployment phase, it is more practical to use a lookup table instead of dynamically generating probabilities with the probability prediction network for each quantized value $\hat{z}_{i,j}$. This approach leverages the fact that the range of possible values for $\hat{z}_{i,j}$ can be precomputed and stored efficiently. Formally speaking, for the i -th channel, denote the support set of $\hat{z}_i = \{\hat{z}_{i,j}\}_{j \in [N']}$ as $[c_i^l, c_i^h] \cap \mathbb{Z}$, then the j -th entry of the lookup table corresponding to the i -th channel, denoted by $\mathcal{C}[i, j]$, is written as

$$\mathcal{C}[i, j] = P_{\phi_i} \left(j + c_i^l + \frac{1}{2} \right) - P_{\phi_i} \left(j + c_i^l - \frac{1}{2} \right), \quad (24)$$

satisfying $\sum_{j=0}^{c_i^h - c_i^l} \mathcal{C}[i, j] = 1, \forall i$. We refer the readers to [13] for more details concerning the probability prediction network which is omitted here due to the page limit.

B. Training the Compression Module

The rounding operation, which generates $\hat{\mathbf{Z}}$ from \mathbf{Z} , is not directly applicable in training due to its non-differentiable

nature, as the gradient is essentially zero almost everywhere. To address this challenge during training, we follow [12] and introduce a noise component sampled from a uniform distribution, $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$, to approximate the rounding error, defining $\tilde{\mathbf{Z}} = \mathbf{Z} + \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$. The probability density function (pdf) of the resulting distribution is given by

$$p(\tilde{\mathbf{Z}}|\phi) = \prod_{i,j} \left(p_{\tilde{Z}_{i,j}|\phi_i}(\phi_i) * \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \right) (\tilde{Z}_{i,j}). \quad (25)$$

To optimize the neural compression framework, including the non-linear transformation functions $f_s(\cdot), g_d(\cdot)$ and the probability prediction net $P_\phi(\cdot)$, we employ a loss function that targets both the quality of signal reconstruction and the efficiency of compression, reflecting a balanced R-D trade-off. The loss function is defined as

$$\mathcal{L} = \lambda \frac{\|\mathbf{x}_f - \hat{\mathbf{x}}_f\|_2^2}{\|\mathbf{x}_f\|_2^2} + I, \quad (26)$$

where λ is a tuning parameter that adjusts the emphasis between reconstruction fidelity and compression rate, and I represents the average bit cost per element, calculated as

$$I = \frac{1}{VN'} \mathbb{E}_{\mathbf{s} \sim p(\mathbf{s})} \mathbb{E}_{\tilde{\mathbf{Z}} \sim q(\tilde{\mathbf{Z}}|\mathbf{s})} \left(-\log_2(p(\tilde{\mathbf{Z}}|\phi)) \right), \quad (27)$$

where the posterior probability $q(\tilde{\mathbf{Z}}|\mathbf{s})$ is given by

$$q(\tilde{\mathbf{Z}}|\mathbf{s}) = \prod_i \mathcal{U} \left(\tilde{z}_{i,j} | z_{i,j} - \frac{1}{2}, z_{i,j} + \frac{1}{2} \right), \quad (28)$$

$$\mathbf{Z} = f_s(\mathbf{s}).$$

The posterior probability follows because $\tilde{\mathbf{Z}}$ is constructed by adding uniform noise to \mathbf{Z} .

We point out that (27) is essentially the cross entropy between the actual distribution of $\tilde{\mathbf{Z}}$, i.e., $\int q(\tilde{\mathbf{Z}}|\mathbf{s})p(\mathbf{s})d\mathbf{s}$ and the predicted distribution $p(\tilde{\mathbf{Z}}|\phi)$. Our goal is to minimize it by learning a $p(\tilde{\mathbf{Z}}|\phi)$ that closely matches the true distribution. Various models, each characterized by different λ settings, are trained to explore distinct points on the R-D curve, ensuring that the neural compression algorithm optimally balances the average compression rate and reconstruction quality in practical applications, as demonstrated in the simulations.

C. Successive Refinement Transmission

In practical network architectures, there can be multiple (virtual) links connecting the RRH to BBU. Each of these links can exhibit distinct Qualities of Service (QoS), typically differentiated by their error probabilities. In this context, we consider the successive refinement transmission strategy [4], which leverages this diversity in link quality to enhance data transmission robustness and efficiency.

In successive refinement transmission, the links are organized in order of increasing error probability, with the most reliable link designated as the base/first layer. This base layer is responsible for transmitting a coarse version of the original signal, \mathbf{s} , providing a foundational reconstruction. Subsequent layers, associated with links of lower reliability, are tasked with transmitting refinement packets that incrementally enhance the signal based on the previously reconstructed output. The key merit of the successive refinement model is that, even if an

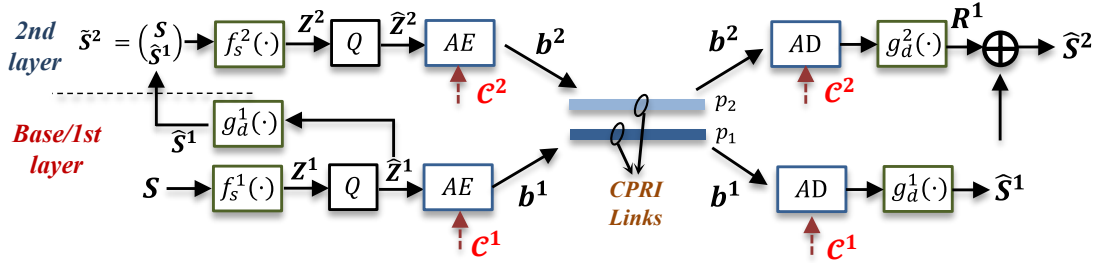


Fig. 5: An illustration of the successive refinement framework with $L = 2$. The encoder of the refinement (2nd) layer aims to produce a bit sequence, b_1 to generate the residual, R^1 . The final reconstruction of the $L = 2$ case is simply a summation of the coarse reconstruction \hat{S}^1 and the residual R^1 .

error occurs in the $(\ell + 1)$ -th packet, we are still able to obtain a usable version of the signal using the information from the first ℓ packets.

We consider L CPRI links, designating one as the base layer and the remaining $L - 1$ as refinement layers. Both decimation and block scaling modules are performed only once, and the input to the successive refinement encoder, S , remains consistent as defined in (10). The operation of the proposed successive refinement model is depicted in Fig. 5 and is described as follows.

1) *Base layer*: The base layer encoder, denoted as $f_s^1(\cdot)$, processes this input to generate Z^1 , with each element then quantized to integer values to form \hat{Z}^1 . This quantized latent representation is subsequently encoded into a bit sequence, b^1 , which is transmitted over the CPRI link. Upon receipt, the base layer decoder, $g_d^1(\cdot)$, decodes b^1 to produce a coarse reconstruction of the original signal, $\hat{S}^1 = g_d^1(\hat{Z}^1)$. The process for generating refinement information begins with $\ell = 2$ and can be extended to scenarios where $\ell > 2$.

2) *Refinement layers*: As shown in Fig. 5, the successive refinement encoder also has access to the base layer decoder, $g_d^1(\cdot)$, thus, it is able to generate \hat{S}^1 , which is identical to the base layer reconstruction at the decoder side. The second-layer encoder concatenates the coarse reconstruction with the original signal, S , to form a larger matrix $\tilde{S}^2 \in \mathbb{R}^{4 \times N'}$, which can be written as $\tilde{S}^2 = \begin{bmatrix} S \\ \hat{S}^1 \end{bmatrix}$. Then, the second-layer encoder, $f_s^2(\cdot)$, is employed: $Z^2 = f_s^2(\tilde{S}^2)$. Note that $f_s^2(\cdot)$ is identical to $f_s^1(\cdot)$ except for the input dimension. The output Z^2 will be quantized and arithmetically encoded to generate the bit sequence, b^2 . At the decoder side, the bit sequence is arithmetically decoded and we use a second-layer decoder, $g_d^2(\cdot)$, to estimate the residual information, R^1 . The reconstruction at the second layer can thus be written as $\hat{S}^2 = \hat{S}^1 + R^1$.

For $\ell > 2$, we have

$$\begin{aligned} \tilde{S}^\ell &= \begin{bmatrix} S \\ \hat{S}^{\ell-1} \end{bmatrix}; \\ Z^\ell &= f_s^\ell(\tilde{S}^\ell); \\ R^{\ell-1} &= g_d^\ell(\hat{Z}^\ell); \\ \hat{S}^\ell &= \hat{S}^{\ell-1} + R^{\ell-1}. \end{aligned} \quad (29)$$

After obtaining \hat{S}^L , \hat{x}_f is generated following the same procedure illustrated in Section II and the reconstruction performance

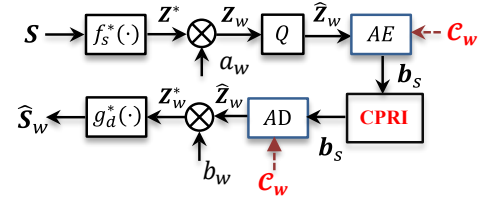


Fig. 6: Pipeline of the variable-rate compression framework, where we assume the model corresponding to the w -th compression level is applied.

is evaluated according to (7) and (8).

3) *Training methods*: The training of the successive refinement layers follows a structured approach. Initially, the neural networks for the base layer are trained via the standard training approach illustrated in Section IV-B, with hyper parameters λ_1 to balance the rate and the distortion terms. Then, the neural network parameters of the ℓ -th ($\ell > 1$) layer are successively trained by fixing the parameters of all previous layers $\ell' < \ell$, and focusing on optimizing the current layer with the loss function

$$\mathcal{L}_\ell = \lambda_\ell \frac{\|x_f - \hat{x}_f^\ell\|_2^2}{\|x_f\|_2^2} + I_\ell, \quad (30)$$

where I_ℓ denotes the estimated bit cost for compressing the latent Z^ℓ and \hat{x}_f^ℓ is obtained from \hat{S}^ℓ in (29).

Remark. The choice of λ_ℓ is crucial; as the reconstruction accuracy improves with each layer, λ_ℓ needs to be adjusted to maintain an effective balance in the diminishing $\frac{\|x_f - \hat{x}_f^\ell\|_2^2}{\|x_f\|_2^2}$ term, ensuring optimal performance across different layers.

D. Variable-Rate Compression

Another critical concern of CPRI compression in practical deployments is managing the diverse reconstruction requirements across different users. This diversity necessitates maintaining multiple neural network configurations at both the RRHs and BBUs, which can lead to significant storage challenges, especially at the RRHs. To effectively address this issue, we introduce a variable-rate compression scheme that not only accommodates varied user demands but also significantly reduces the storage footprint. The methodology is applicable to both uplink and downlink scenarios.

1) *Overview*: For each model with a specific λ value, RRH has to store the corresponding weights for the bi-GRU and FC modules as well as the lookup table introduced in (24) for entropy coding. This causes severe storage issue when the number of different λ values is large. Note that in our case, the lookup table \mathcal{C} is relatively small. Thus, we seek to save the storage by introducing a shared non-linear transformation module, $f_s(\cdot)$ for all the models targeting different R-D performances.

To achieve this, we start from the model with the best reconstruction performance (trained under the largest λ value), whose latent encoding and decoding functions are denoted as $f_s^*(\cdot)$ and $g_d^*(\cdot)$, respectively. For an input signal \mathbf{S} , we first obtain the latent vector, $\mathbf{Z}^* = f_s^*(\mathbf{S})$. Then, for the w -th ($w \in [W]$) reconstruction, we scale the latent by the scaling factor, $a_w \in (0, 1]$:

$$\begin{aligned} \mathbf{Z}_w &= a_w \mathbf{Z}^*, \\ 0 &< a_1 < a_2 < \dots < a_W = 1. \end{aligned} \quad (31)$$

Note that for the special case when $w = W$, we have $a_W = 1$, $\mathbf{Z}_W = \mathbf{Z}^*$ which corresponds to the model with the best reconstruction performance. The w -th scaled latent is then quantized as:

$$\hat{\mathbf{Z}}_w = \left\lfloor \mathbf{Z}_w + \frac{1}{2} \right\rfloor.$$

Note that the scaling operation will change the underlying distribution of the quantized latent variable, thus, the probability prediction net, $P_\phi^*(\cdot)$ optimized for the latent $\hat{\mathbf{Z}}^*$, is no longer feasible for the scaled one. To resolve this, for each $a_w, w \in [W - 1]$, we train a new probability prediction net, denoted by $P_{\phi_w}^w(\cdot)$, which will be detailed later. The decoding process also needs to be changed with respect to the scaling operation. To be precise, we introduce a set of re-scaling factors:

$$b_w = 1/a_w, \quad w \in [W] \quad (32)$$

for different models, and they will be directly multiplied with the quantized latents as:

$$\hat{\mathbf{Z}}_w^* = b_w \left\lfloor a_w \mathbf{Z}^* + \frac{1}{2} \right\rfloor \approx \hat{\mathbf{Z}}^*. \quad (33)$$

The approximation holds due to the fact that we choose $b_w = 1/a_w$. This is to ensure the input to the latent decoding function, $g_d^*(\cdot)$ has nearly identical distribution with its original input, $\hat{\mathbf{Z}}^*$, such that $g_d^*(\cdot)$ can produce satisfactory reconstruction without modifying its parameters.

2) *Obtaining $P_{\phi_w}^w(\cdot)$ and \mathcal{C}_w* : We then illustrate the acquisition of new probability prediction net $P_{\phi_w}^w(\cdot)$ and the codebook \mathcal{C}_w corresponding to the w -th reconstruction. In particular, we keep the neural network parameters of the non-linear transformation functions, $f_s^*(\cdot)$ and $g_d^*(\cdot)$ to be fixed during training and randomly initialize the parameters of $P_{\phi_w}^w(\cdot)$. The training objective can be expressed as:

$$\min_{\phi_w} I_w(\phi_w), \quad (34)$$

where I_w follows the same definition in (27) and is a function of the parameters, ϕ_w . Note that we omit the reconstruction term in (26) as it is independent with respect to ϕ_w and the loss function \mathcal{L}_w is simply I_w . In other words, we only need to

train a probability prediction net which predicts the distribution of the scaled latent, $\hat{\mathbf{Z}}_w^*$, to produce an accurate estimation of I_w . Since the parameters of the latent encoding and decoding functions are fixed, the training process can be made efficient. After obtaining $P_{\phi_w}^w(\cdot)$, we can construct the lookup table, \mathcal{C}_w for each of the models with $w \in [W - 1]$ following (24).

V. NUMERICAL EXPERIMENTS

A. Parameter Settings and Training Details

To verify the efficacy of the proposed schemes, we consider a 5G scenario [27], where the channel bandwidth, subcarrier spacing, FFT size, and the sampling frequency are set to 20 MHz, 60 kHz, 512 and 30.72 MHz, respectively. The decimation module for both uplink and downlink scenarios first upsamples the signal by 5 times followed by a low-pass filter, and finally downsamples the filtered signal by 8 times.

Next, we introduce the DNN parameters for the proposed latent quantization and neural compression schemes. The bi-GRU module of the non-linear transformation function of both schemes has 2 hidden layers with the hidden size equals to $N_h = 32$. We set $V = 2$ for the latent vectors $\{\mathbf{z}_j\}_{j \in [N]}; \mathbf{z}_j \in \mathbb{R}^V$. The batch size for both training and evaluation phases is set to 32. Adam optimizer is adopted with a varying learning rate, initialized to 10^{-4} and dropped by a factor of 0.8 if the validation loss does not improve in 20 consecutive epochs.

B. General Performance

1) *Downlink case*: We first evaluate the R-D performance of the downlink signal. In this case, CP is removed and the input to the non-linear transformation function is of 640-dimension. The modulation order is assumed to be 64. For both scalar and vector latent quantization approaches, we consider bits per element, $Q \in \{4, 5, 6, 7\}$, while the block size b is set to 2 for the vector quantization scheme. For the neural compression approach, different models are trained corresponding to $\lambda = \{10^2, 5 \times 10^2, 10^3, 5 \times 10^3\}$. The block scaling parameter, N_s is set to 320 for all these schemes. In other word, only one scaling factor, t , is used for the entire input sequence. Two conventional schemes are considered as benchmarks. To be precise, we implement the scalar quantization (SQ) proposed in [4] and the vector quantization (VQ) scheme from [5]. For a fair comparison, we assume they adopt the same N_s as in the proposed schemes. Their relative R-D performance in terms of EVM defined in (7) and (8) are shown in Fig. 7 (a) and (b), respectively.

As can be seen, the scalar quantization scheme in [4] yields the worst performance as the elements of the input sequence are non-i.i.d.. Both the vector quantization schemes, i.e., the learned scheme as well as that in [5] adopt entropy coding illustrated in (5) to further enhance the compression efficiency bringing an approximately 0.1 bit per element reduction. Both the learned scalar and vector latent quantization approaches achieve better performance compared with the conventional scalar and vector quantization schemes, respectively. The learned latent vector quantization outperforms its uniform quantization counterpart as it relaxes the assumption that each element follows an i.i.d. uniform distribution (while the conventional one adopts non-uniform quantization). Finally, we find the neural compression

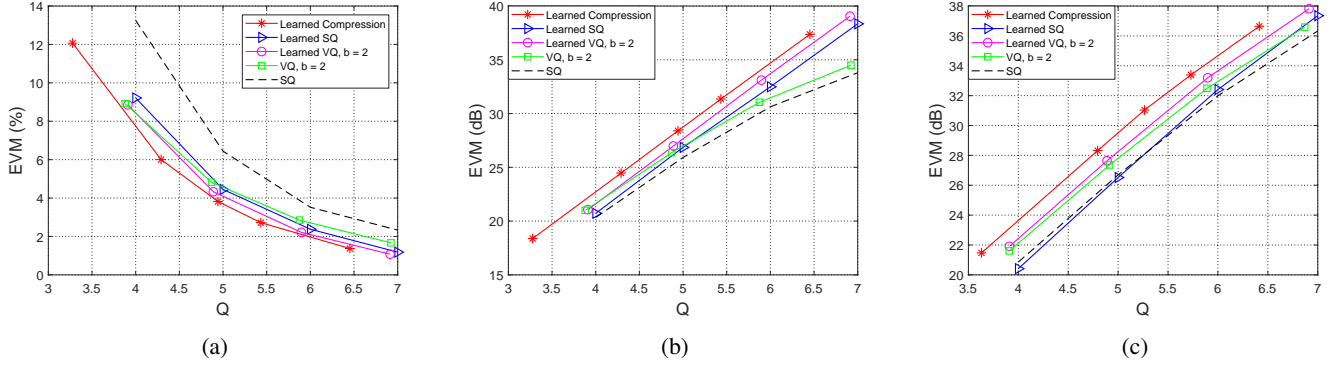


Fig. 7: Reconstruction performances of the proposed schemes for CPRI compression: (a) & (b) the EVM performance (in percentage and dB) of the reconstructed signal in the downlink scenario; (c) EVM in dB for the uplink scenario where the cyclic prefix as well as the received signal are jointly compressed.

algorithm outperforms all the other schemes as expected, as it dedicatedly models the latent distribution and arithmetically encodes the quantized latent using a variable-rate codebook based on its distribution.

2) *Uplink case:* For the uplink scenario, we assume that a single user² is communicating with the base station using the SC-FDMA modulation. A 7-tap multi-path fading channel is considered where the coefficient of each tap is modeled as a complex Gaussian random variable drawn from $\mathcal{CN}(0, 1)$. The SNR is set to 5 dB and the modulation order is 64.

Different from the downlink case, after receiving the uplink signal, the RRH compresses the **entire** signal including CP as it is used to perform synchronization at the BBU. We assume the CP length equals to 64 thus the input s to the latent encoding function is of 720 dimensions resulting in a block scaling parameter $N_s = 360$, i.e., a single scaling factor t for the whole input. The relative performance of the proposed learning-based schemes as well as the baselines is shown in Fig. 7 (c). The latent vector quantization scheme achieves superior EVM performance over the conventional VQ scheme. However, the learned latent uniform quantization scheme is outperformed by the conventional SQ scheme when $Q < 6$, which is because the learned scheme assumes uniform quantization whereas the conventional SQ adopts non-uniform quantization. These observations are slightly different from the downlink case which is due to the fact that the pdf of the input signal s changes after passing through the multi-path fading channel. The proposed neural compression scheme, on the other hand, shows its effectiveness which outperforms the conventional VQ by $Q \approx 0.3$ when EVM equals to 32 dB.

C. Performance with mismatched scenarios

We then provide simulation results for the mismatched scenarios. In particular, mismatched SNR values, constellation orders and number of channel taps are considered for the uplink scenario. Note that, since the neural compression model outperforms its latent quantization counterpart, we will focus on the neural compression scheme in the following simulations.

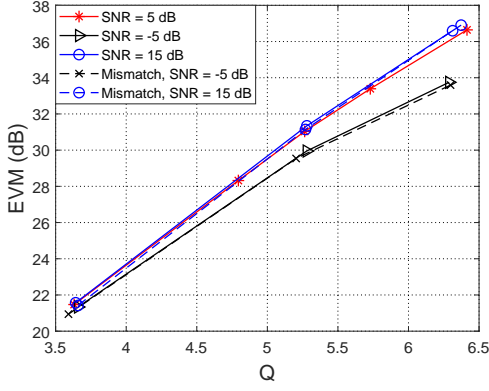
²It can be easily extended to the multi-user scenario, yet we will not show the corresponding simulations due to the page limit.

1) *Mismatched SNR values:* In this simulation, we adopt the previous setting where 64QAM symbols are transmitted over a noisy 7-tap multi-path fading channel. Different models are trained with three different SNR values, namely, $\{-5, 5, 15\}$ dB. We evaluate the model trained at 5 dB under SNR values of $\{-5, 15\}$ dB and the R-D curves are shown in Fig. 8 (a).

As can be seen, the proposed neural compression module shows its robustness against the SNR variations as the R-D curves evaluated at $\text{SNR} = \{-5, 15\}$ dB are similar or slightly worse compared to the matched case, where the model is trained and tested under the same SNR value. It is also observed that for the matched cases, the R-D performance under a large SNR value is better. This can be understood by considering the extreme cases: when $\text{SNR} = \infty$ where no noise is added, the signal exhibits certain patterns and the neural compression module may utilize such patterns for better compression. On the other hand, when SNR is sufficiently low, the signal follows a Gaussian distribution, since Gaussian random variable has the largest entropy with a fixed variance, the R-D curve would be less satisfactory.

2) *Mismatched modulation orders:* The previous simulations assume 64QAM symbols. We consider different constellation orders such as 4QAM and 16QAM and evaluate the model trained at 64QAM on the mismatched constellations. The simulation assumes a 7-tap multi-path fading channel with $\text{SNR} = 5$ dB. We found by experiments that the proposed scheme can achieve almost identical R-D performance regardless of the modulation order. Furthermore, the model trained at a specific constellation can achieve nearly the same performance when the modulation order changes. The simulation result is omitted due to the page limit.

3) *Mismatched number of taps:* In this part, we demonstrate the robustness of the proposed scheme by changing the number of channel taps, denoted as N_{tap} , of the noisy multi-path fading channel. In particular, the SNR of the channel is fixed at 5 dB and we consider $N_{\text{tap}} = \{1, 3, 7\}$. Then, the R-D performance with respect to different N_{tap} is shown in Fig. 8 (b). It can be seen that a better R-D curve is obtained when N_{tap} is larger. Moreover, the proposed scheme can achieve similar or slightly worse performance when there is a mismatch in the number of hops.



(a) Mismatched SNRs

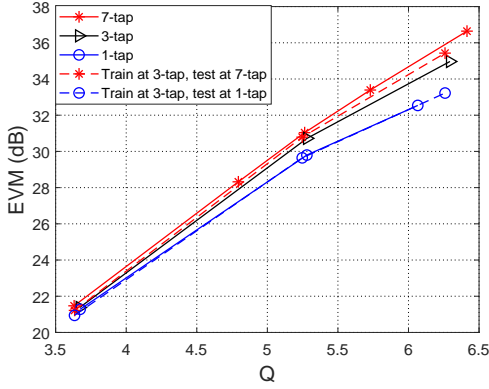
(b) Mismatched N_{tap}

Fig. 8: The R-D curves for the mismatched scenarios, namely, the mismatched SNR values and the number of channel taps. The dashed lines denote the R-D curve when a model trained with a specific SNR/number of hops is evaluated under a different setting.

D. Improved Neural Compression Solutions

1) *Successive refinement model*: We then show the performance of the successive refinement model illustrated in Section IV-C. In this simulation, the same setup in Fig. 7 (c) is considered and the maximum number of layers is set to 3, i.e., $L = 3$. The base layer is trained with $\lambda_1 = 10^2$ and the subsequent layers with indexes $\ell = 2, 3$ are trained with carefully chosen $\lambda_2, \lambda_3 = 10^3, 10^4$, respectively. It can be seen from Fig. 9 that the successive refinement scheme has a small gap with respect to the separately trained baselines. This is due to the fact that the successive refinement module requires reconstruction at the ℓ -th layer after receiving $\{b_1, \dots, b_\ell\}$. The separately trained models on the other hand, reconstruct the model after all the bit sequence, \mathbf{b} is received. A more stringent constraint results in a performance loss compared to the separately trained ones.

2) *Variable-rate compression*: Finally, we manifest the effectiveness of the proposed variable-rate compression model. The uplink scenario with 7-tap noisy multi-path fading channel and SNR = 5 dB is considered. The shared latent encoding and decoding functions, $f_s^*(\cdot), g_d^*(\cdot)$, are selected from the model trained with $\lambda = 5 \times 10^3$. $W = 5$ scaling factors are introduced,

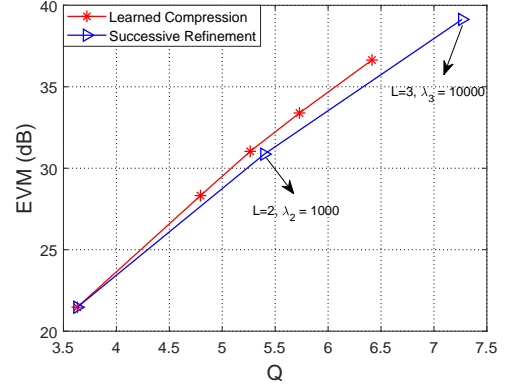


Fig. 9: The relative R-D performance between the successive refinement models and the separately trained ones. We consider $L = 3$ with $\lambda_\ell \in \{10^2, 10^3, 10^4\}$ for the successive refinement model.

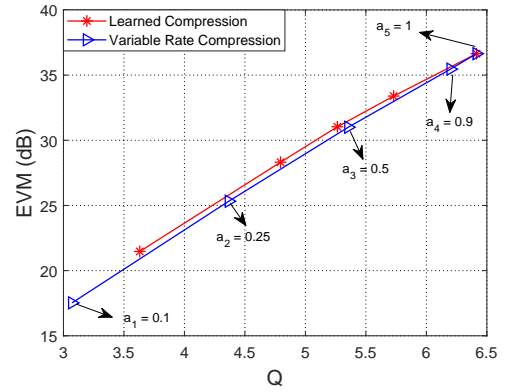


Fig. 10: The relative performance between the variable-rate compression model with shared non-linear transformation functions, $f_s^*(\cdot), g_d^*(\cdot)$ and the models trained with distinct $f_s(\cdot), g_d(\cdot)$.

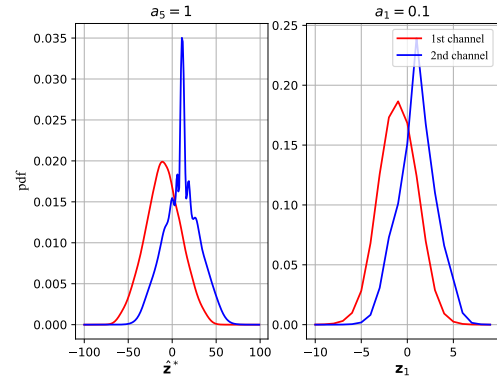


Fig. 11: We plot the entries of the codebook \mathcal{C}_W as well as the codebook \mathcal{C}_1 corresponding to $a_W = 1$ and $a_1 = 0.1$, respectively. Both the codebooks have two rows/channels and there is a shrinkage in the support set for the $a_1 = 0.1$ case.

namely, $a_w \in \{0.1, 0.25, 0.5, 0.9, 1\}$. The R-D performance of the variable-rate compression model with respect to the

separately trained models in Fig. 7 (c) is shown in Fig. 10.

As can be seen, the variable-rate compression model can achieve similar reconstruction performance with reduced storage cost compared to the separately trained models. In particular, the two curves coincide at $a_W = 1$ and a small gap of $Q \approx 0.1$ is observed when EVM equals to 25 dB. We then quantify the reduction in storage for the variable-rate compression model. For each of the separately trained model, the weights of the latent encoding and decoding functions, $f_s(\cdot)$ and $g_d(\cdot)$ are stored together with the corresponding codebook, \mathcal{C} . As a concrete example, approximately 2.5×10^4 floating-point parameters are introduced for the two-layer bi-directional GRU as well as the FC model shown in Fig. 3. In contrary, the codebook \mathcal{C} (or equivalently, \mathcal{C}_W), for the model with $\lambda = 5 \times 10^3$ is only comprised of 2×200 entries. We note that each entry corresponds to a non-zero probability value and for each of the $V = 2$ channels, all the corresponding entries summed up to be 1. Due to the fact that the size of the codebook is negelectable compared to the weights of the non-linear transformation functions, the proposed scheme reduces the storage cost to $1/W$ compared to the previous scheme.

We further provide the learned pdf, i.e., the entries of the codebook, \mathcal{C}_w , of the scaled latent vector, \mathbf{Z}_w . In particular, the entries of \mathcal{C}_W and \mathcal{C}_1 are shown in Fig. 11. It is clear from the figure that the support set of \mathcal{C}_1 shrinks approximately 1/10 compared to that of the \mathcal{C}_W . This matches with the analysis in Section IV-D, that the entropy of \mathbf{Z}_w reduces after scaling. The figure also shows that two different channels for the codebook follow two different distributions.

VI. CONCLUSION

This paper proposed two AI-native fronthaul compression algorithms: latent quantization and neural compression, for efficient signal transmission over the CPRI link in C-RAN architectures. These methods extend traditional decimation and block scaling frameworks through learned non-linear transformations and latent source coding strategies, offering substantial improvements in rate-distortion performance. To address practical deployment scenarios in NextG networks, we further introduced two system-level enhancements: a successive refinement model for adaptive transmission over heterogeneous CPRI links and a variable-rate compression framework using shared neural components to reduce the storage on resource-constrained devices. Extensive evaluations confirmed that the proposed models consistently outperform conventional compression schemes in both uplink and downlink settings, and remain robust under mismatched channel conditions, SNRs, and modulation formats offering valuable implications for standardization efforts in 6G.

APPENDIX A

ON THE DISTRIBUTION OF THE TIME DOMAIN SIGNAL

This appendix provides detailed derivation of the distribution of the decimation module output, \mathbf{x}' , shown in Proposition 1 of Section III.

We are interested in exploring whether the elements of the input signal, \mathbf{s} , which is the input to the transformation and source encoding modules, are i.i.d.. In the downlink case, \mathbf{s} is obtained by first applying IFFT to N_{fft} subcarriers where

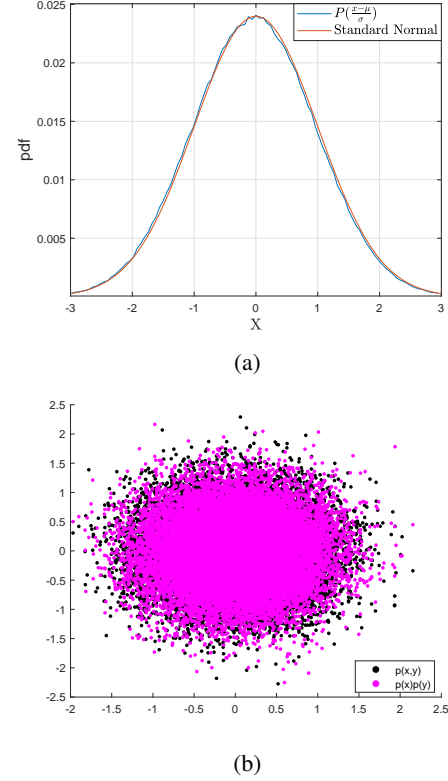


Fig. 12: Visualization of the probability distribution for the random variables, X and Y : (a) the marginal distribution of $\frac{X-\mu}{\sigma}$ deviates from the standard normal distribution; (b) different variables, X and Y are correlated which is verified by comparing the joint distribution $p(x, y)$ with the product of the marginal distributions, $p(x)p(y)$.

N_{sym} of them are occupied by i.i.d. QAM symbols, denoted by \mathbf{c} , followed by decimation and block scaling. We will show for the downlink case³ that the elements in \mathbf{s} are non-i.i.d. which motivates the introduction of the non-linear transformation functions in Section III.

We first focus on the output signal of the decimation module, \mathbf{x}' . Note that we carefully select the K and M values⁴ and design a low-pass filter such that the spectrum of \mathbf{x}' , denoted as \mathbf{w} , is still comprised of the same QAM symbols, \mathbf{c} , but with a narrower guard band (in other words, less number of zero subcarriers) compared to the spectrum of \mathbf{x} . In particular, \mathbf{w} can be expressed as:

$$\mathbf{w} = [\mathbf{0}_{k^*}^\top, \mathbf{c}^\top, \mathbf{0}_{k^*}^\top]^\top, \quad (35)$$

where⁵ $k^* = (N' - N_{sym})/2$. The n -th element of \mathbf{x}' can then be expressed as:

$$x'_n = \frac{1}{\sqrt{N'}} \sum_{k=k^*}^{N'-k^*-1} w_k e^{j \frac{2\pi k n}{N'}}. \quad (36)$$

³It is easy to show the same property for the uplink scenario.

⁴We have the relation $\frac{N_{sym}}{N_{fft}} < \frac{K}{M}$.

⁵We assume $(N' - N_{sym})$ is an even number which can be satisfied via zero padding of the QAM symbols, \mathbf{c} .

The expectation of x'_n is expressed as:

$$\mathbb{E}(x'_n) = \frac{1}{\sqrt{N'}} \sum_{k=k^*}^{N'-k^*-1} \mathbb{E}(w_k^r + jw_k^i) e^{j\frac{2\pi nk}{N'}} = 0, \quad (37)$$

where w_k^r and w_k^i represent the real and imaginary parts of the QAM symbol, w_k , respectively. Since w_k^r and w_k^i are i.i.d. and can be seen as pulse-amplitude modulation (PAM) symbols, we have $\mathbb{E}(w_k^r) = \mathbb{E}(w_k^i) = 0$. The covariance between x'_n and x'_m can be derived using (37) and the fact that different w_k elements are i.i.d.:

$$\begin{aligned} \text{Cov}(x'_n, x'_m) &= \frac{1}{N'} \mathbb{E} \sum_{k_1} \sum_{k_2} w_{k_1} e^{j\beta n k_1} w_{k_2}^\dagger e^{-j\beta m k_2} \\ &= \frac{P}{N'} \sum_{k=k^*}^{N'-k^*-1} e^{j\beta(n-m)k}, \end{aligned} \quad (38)$$

where we define $\beta \triangleq \frac{2\pi}{N'}$, $\mathbb{E}(|w_k|^2) \triangleq P$ and utilize the fact that $\mathbb{E}(w_{k_1}^\dagger w_{k_2}) = P\delta_{k_1, k_2}$. It is easy to see when $k^* = 0$, the covariance would always equal to zero unless $m = n$. However, in practical settings, k^* is non-negative to ensure that the low-pass filter preserves the signals located at the occupied subcarriers. Thus, the elements of x' are correlated (non-i.i.d.) and the subsequent block scaling operation does not change the property. This finishes the proof of Proposition 1 validating that transformations are needed to achieve a better CPRI compression performance.

We further analyze the marginal distribution of the n -th element, x'_n and the joint distribution of two different elements, denoted as x'_n and x'_m . We study the real part⁶ of x'_n , which is a random variable, denoted by X . According to the central limit theory, when N_{sym} is large, the average of i.i.d. random variables, w_k in (36), would converge to a normal distribution, $\mathcal{N}(\mu, \sigma^2)$. In the considered scenario, however, due to the weighted term, i.e., $e^{j\frac{2\pi nk}{N'}}$, the distribution of X deviates slightly from the normal distribution which can be derived using the characteristic function. Due to the page limit, we simply illustrate this by comparing the empirical distribution of $\frac{X-\mu}{\sigma}$, denoted as $P\left(\frac{X-\mu}{\sigma}\right)$ with the standard normal distribution in Fig. 12 (a). Note that sufficient number of X implementations are collected to estimate $P\left(\frac{X-\mu}{\sigma}\right)$.

We then illustrate the dependency of the real parts of the two random variables, x'_n and x'_m , which we denote them as X and Y . This is done by showing the difference between the joint distribution, denoted as $p(x, y)$ and the product of two marginal distributions, $p(x)p(y)$. We draw 2×10^4 samples from the two distributions which are as shown in Fig. 12 (b). As can be seen, the samples drawn from the two marginal distributions, $p(x)p(y)$, are more concentrated to the origin compared with those drawn from the joint distribution, $p(x, y)$. This further validates the correlation among different elements.

REFERENCES

- [1] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): a primer," *IEEE Netw.*, vol. 29, no. 1, pp. 35–41, 2015.
- [2] A. De la Oliva, J.-A. Hernandez, D. Larrabeiti, and A. Azcorra, "An overview of the CPRI specification and its application to C-RAN-based LTE scenarios," *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 152–159, 2016.
- [3] B. Guo, W. Cao, A. Tao, and D. Samardzija, "LTE/LTE-A signal compression on the CPRI interface," *Bell Labs Tech. J.*, vol. 18, no. 2, pp. 117–133, 2013.
- [4] D. Samardzija, J. Pastalan, M. MacDonald, S. Walker, and R. Valenzuela, "Compressed transport of baseband signals in radio access networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 9, pp. 3216–3225, 2012.
- [5] H. Si, B.-L. Ng, M.-S. Rahman, and J. Zhang, "A novel and efficient vector quantization based CPRI compression algorithm," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7061–7071, 2017.
- [6] K. F. Nieman and B. L. Evans, "Time-domain compression of complex-baseband LTE signals for cloud radio access networks," in *IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, 2013, pp. 1198–1201.
- [7] S. H. Park, O. Simeone, O. Sahin, and S. Shamai, "Robust and efficient distributed compression for cloud radio access networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 2, pp. 692–703, 2013.
- [8] R. Qiao, T. Jiang, and W. Yu, "Meta-learning-based fronthaul compression for cloud radio access networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 11015–11029, 2024.
- [9] J. Choi, B. L. Evans, and A. Gatherer, "Space-time fronthaul compression of complex baseband uplink LTE signals," in *IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–6.
- [10] N. Samuel, T. Diskin, and A. Wiesel, "Learning to Detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, 2019.
- [11] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4293–4308, 2017.
- [12] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Int. Conf. Learn. Represent. (ICLR)*, 2017.
- [13] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Int. Conf. Learn. Represent. (ICLR)*, 2018.
- [14] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.
- [15] E. Ozfatura, Y. Shao, A. G. Perotti, B. M. Popović, and D. Gündüz, "All you need is feedback: Communication with block attention feedback codes," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 3, pp. 587–602, 2022.
- [16] E. Boursoulatz, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Trans. Cogn. Commun. Netw.*, 2019.
- [17] A. Grönlund, A. Russo, Y. Jedra, B. Klaiqi, and X. Gelabert, "Constrained deep reinforcement learning for fronthaul compression optimization," in *IEEE Intl. Conf. Mach. Learn. Commun. Netw. (ICMLCN)*, 2024, pp. 498–504.
- [18] S. Yagi, M. Nakahara, K. Suzuoki, T. Kozuno, and D. Hisano, "Deep learning-based nonlinear quantizer for fronthaul compression," in *OptoElectronics and Commun. Conf. (OECC) and Int. Conf. Photonics in Switching and Computing (PSC)*, 2022, pp. 1–3.
- [19] F. Yang, L. Herranz, J. v. d. Weijer, J. A. I. Guitián, A. M. López, and M. G. Mozerov, "Variable rate deep image compression with modulated autoencoder," *IEEE Signal Process. Lett.*, vol. 27, pp. 331–335, 2020.
- [20] Z. Cui, J. Wang, S. Gao, T. Guo, Y. Feng, and B. Bai, "Asymmetric gained deep image compression with continuous rate adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 10532–10541.
- [21] T. Chen and Z. Ma, "Variable bitrate image compression with quality scaling factors," in *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2020, pp. 2163–2167.
- [22] C. Bian, Y. Shao, and D. Gündüz, "DEEPJSCC-1++: Robust and bandwidth-adaptive wireless image transmission," in *IEEE Global Commun. Conf. (GLOBECOM)*, 2023, pp. 3148–3154.
- [23] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, Academic Press, 2014.
- [24] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.
- [25] T.-Y. Tung, D. B. Kurka, M. Jankowski, and D. Gündüz, "DEEPJSCC-Q: Constellation constrained deep joint source-channel coding," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 4, pp. 720–731, 2022.
- [26] A. Van Den Oord, O. Vinyals, et al., "Neural discrete representation learning," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [27] S. Lagen, K. Wanuga, H. Elkotby, S. Goyal, N. Patriciello, and L. Giupponi, "New radio physical layer abstraction for system-level simulations of 5G networks," in *IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–7.

⁶The distribution of the imaginary part follows the same analysis.