SINR Sequence Compression and Quantization with VQ-VAE Method

Lingrui Zhu, Carsten Bockelmann, Armin Dekorsy
Department of Communications Engineering, University of Bremen, 28359, Bremen, Germany
Email: {zhu, bockelmann, dekorsy}@ant.uni-bremen.de

***Abstract*—In this study, we introduce an innovative signal to interference and noise ratio (SINR) time sequence feedback scheme based on vector quantization variational autoencoder (VQ-VAE). We compress the SINR sequence at the user equipment (UE) side and reconstruct it at the base station (BS) side. The reconstructed sequence is then utilized for SINR prediction at the BS. The VQ-VAE framework compresses SINR sequences into a compact embedding space involving several embedding vectors. Instead of transmitting the entire compressed SINR sequence back, we only need to transmit the index of the corresponding embedded vector. Based on the index, the sequence will be reconstructed. Moreover, a principal component analysis (PCA) based method is employed to reshape the distribution of the embedding space and compression performance is improved consequently. Our numerical simulations demonstrate that VQ-VAE combined with PCA achieves superior reconstruction and prediction accuracy while requiring fewer quantization bits compared to 3GPP commonly used method, differential quantization. Therefore, the proposed scheme is a promising solution for enhancing SINR sequence compression and prediction in wireless communication systems.**

***Index Terms*—SINR feedback/CQI, machine learning, quantization, sequence compression, .**

I. Introduction

In modern wireless communication systems, link adaptation (LA) is a commonly applied technique that involves selecting the appropriate modulation and coding scheme (MCS) based on the signal to interference and noise ratio (SINR) [1], [2]. To achieve this, user equipments (UE) report channel quality indicators (CQI) to the base station (BS) in order to provide SINR information. Due to factors like transmission delay in CQI feedback and the dynamic nature of the channel and interference, CQI information can become outdated because of channel aging [3]. Therefore, 3GPP also lists channel state information (CSI) compression and prediction as important use cases of artificial intelligence for 5G New Radio [4].

This issue has been tackled using various methods such as extrapolation [5], linear prediction with stochastic approximation [6], and Wiener filtering [7]. However, in the aforementioned works, prediction is assumed to take place at the UE side, primarily due to the inaccuracies and extra overhead of CQI feedback. Nevertheless, UE computational resources, particularly for low-cost devices, are severely constrained. In our research, we introduce SINR feedback schemes based on the vector quantized variational autoencoder (VQ-VAE) [8], which employs an autoencoder to compress SINR sequences, followed by the vector quantization of trainable embedding vectors. The decoder in VQ-VAE is able to reconstruct SINR sequences according to quantized embedding vectors. Thus, the proposed scheme enables SINR sequence prediction at the BS side.

Our contribution involves: i) introducing a VQ-VAE based scheme for SINR sequence compression, ii) proposing the principal component analysis binary splitting (PBS) re-initialization method to enhance the performance of compression and quantization, and iii) demonstrating through numerical results that, compared to the commonly used differential quantization (Diff-Quant) methods in 3GPP standardization [9], our proposed VQ-VAE combined with the PBS scheme can significantly reduce the number of quantization bits required for CSI feedback.

II. System Model

Considering a downlink (DL) transmission from BS to the UE in the presence of interferers denoted by a set $\mathcal{I}$, and white Gaussian noise, $w \sim \mathcal{N}(0, \sigma^2)$, the SINR in dB scale of DL transmission at time t can be written as:

$$\gamma[t] = 10\log_{10}[\frac{|h_s[t]|^2 \cdot p_s[t]}{\sigma^2 + \sum_{i\in\mathcal{I}} |h_i[t]|^2 \cdot p_i[t]}], \tag{1}$$

where h_s and p_s represent the channel gain and transmission power from BS to UE, respectively. The channel gain of interferers is denoted as h_i and the interferers power as p_i. The channel gain can be modeled using the following equation:

$$|h[t]|^2 = |\eta[t]|^2 \cdot \psi[t] \cdot \zeta[t], \tag{2}$$

where $\eta[t]$, $\psi[t]$, and $\zeta[t]$ stand for circular symmetric Gaussian distributed small scale fading [10], exponential path loss [11], and log-normal distributed shadowing [12], respectively.

The proposed SINR compression and prediction scheme is depicted in Fig. 1. At first, the channel state information reference signal (CSI-RS) is transmitted by the BS to estimate the SINR at the UE side. The SINR sequence at time t, $\mathbf{x}_t \in \mathbb{R}^L$, includes the SINR values in the previous L time slots and it can be written as:

$$\mathbf{x}_t = [\gamma[t-L+1], \gamma[t-L+2], ..., \gamma[t]]^T. \tag{3}$$

To reduce CSI feedback overhead, we compress sequence $\mathbf{x}_t$ into $\mathbf{z}_t \in \mathbb{R}^D$ with $D < L$, and transmit it to the BS side as shown in Fig. 1. On the BS side, an estimate of the SINR sequence $\hat{\mathbf{x}}_t$ is recovered based on $\mathbf{z}_t$. Then SINR sequence is predicted for the next M time slots, denoted by:

$$\mathbf{y}_t = [\gamma[t+1], \gamma[t+2], ..., \gamma[t+M]]^T. \tag{4}$$

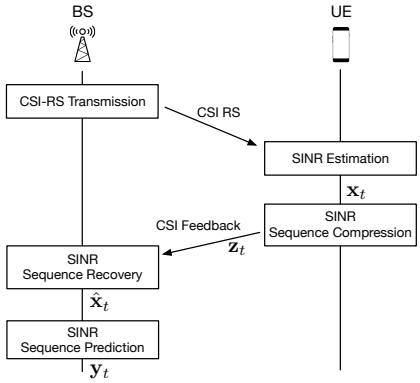

Fig. 1. Proposed scheme of SINR prediction at BS using SINR sequence compression for CSI feedback.

III. SINR Sequence Compression and Prediction

We use a vector quantized variational autoencoder (VQ-VAE) [8] to compress and quantize the SINR sequence at the UE. The SINR sequence will be predicted using LSTM models at the BS side.

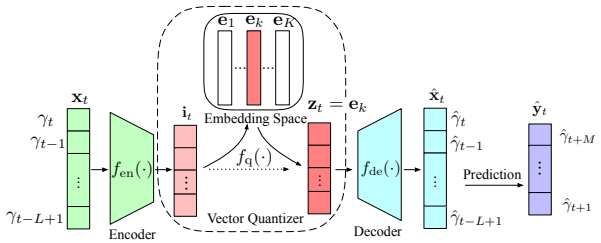

Fig. 2. The structure of VQ-VAE based CSI feedback consisting of encoder, vector quantizer, and decoder.

A. VQ-VAE CSI Feedback

The input of the encoder at time t, $\mathbf{x}_t \in \mathbb{R}^L$, according to Eq. (3) is defined as a sequence of SINR values. As depicted in Fig. 2, $f_{\text{en}}(\cdot)$ and $f_{\text{de}}(\cdot)$ represent functions of encoder and decoder, correspondingly. The latent variable $\mathbf{i}_t$, i.e., the output of the encoder is denoted as $\mathbf{i}_t = f_{\text{en}}(\mathbf{x}_t) \in \mathbb{R}^D$. The latent variable is discretely represented by a specific embedding vector, $\mathbf{e}_k \in \mathbb{R}^D$, from the embedding space $\mathbf{E} \in \mathbb{R}^{D \times K}$, which is defined as:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_K]. \tag{5}$$

The output of the vector quantizer (VQ) layer, quantized latent vector $\mathbf{z}_t$, is chosen as the embedding vector with the minimum squared Euclidean distance to $\mathbf{i}_t$:

$$\mathbf{z}_t = f_{\text{q}}(\mathbf{i}_t) = \arg\min_{\mathbf{e}_k \in \mathbf{E}} \|\mathbf{i}_t - \mathbf{e}_k\|_2^2. \tag{6}$$

The Voronoi set of embedding vector $\mathbf{e}_k$ contains the encoder outputs closest to it and can be defined as:

$$\mathcal{V}_k = \{\mathbf{i} \mid \forall l \neq k \;\; ||\mathbf{i} - \mathbf{e}_k||_2^2 \leq ||\mathbf{i} - \mathbf{e}_l||_2^2\}. \tag{7}$$

The decoder uses the quantized latent vector $\mathbf{z_t}$ to recover $\mathbf{x}$,

$$\hat{\mathbf{x}}_t = f_{\text{de}}(\mathbf{z}_t). \tag{8}$$

With VQ-VAE, the CSI feedback process consists of following steps: 1) the UE estimates SINR and obtains SINR sequence $\mathbf{x}_t$; 2) $\mathbf{x}_t$ is then compressed to $\mathbf{i}_t$, and mapped to the nearest embedding vector, $\mathbf{e}_k$; 3) the index, k, is transmitted back to the BS; 4) the BS recovers the SINR sequence $\hat{\mathbf{x}}_t$ according to $\mathbf{e}_k$.

B. VQ-VAE Training

In this section, two training strategies for VQ-VAE will be introduced. To improve the readability of the explanations, the subscript t will be omitted in the following.

1) Loss function with Embedding Loss (EL): The loss function of VQ-VAE comprises three components: *reconstruction loss*, *commitment loss*, and *embedding loss*, defined as [8]:

$$\mathcal{L}_{\text{EL}} = \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}_{\text{reconstruction loss}} + \overbrace{\beta \cdot \|\mathbf{i} - \text{sg}(\mathbf{z})\|_2^2}^{\text{commitment loss}} + \underbrace{\|\mathbf{z} - \text{sg}(\mathbf{i})\|_2^2}_{\text{embedding loss}}, \tag{9}$$

where $\text{sg}(\cdot)$, the stop gradient operator, indicates that the included parameters are non-updated constants during the back propagation. The reconstruction loss updates encoder and decoder parameters similar to the conventional autoencoder. The commitment loss trains the encoder to align latent variables with assigned embedding vectors, depicted as blue arrows in Fig. 3. Scaling factor for the commitment loss is denoted by β and we set $\beta = 0.25$ according to [8]. The embedding loss updates embedding vectors to converge towards the latent variables in the corresponding Voronoi set, drawn as the red arrows in the right part of Fig. 3.

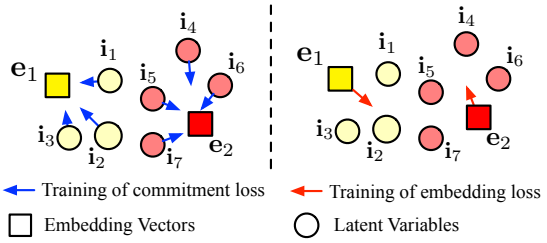

Fig. 3. The training process of commitment loss and embedding loss. Latent variables with the same color as the embedding vector are in the corresponding Voronoi cell. Commitment loss (blue) shifts latent variables closer to their assigned embedding vectors, while embedding loss (red) updates embedding vectors to be nearer to their corresponding latent variables.

2) Loss function with Exponential Moving Averages (EMA): Also in [8], EMA is introduced to update embedding vectors. When using EMA, the loss function of VQ-VAE only includes the reconstruction loss and the commitment loss:

$$\mathcal{L}_{\text{EMA}} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \beta \cdot \|\mathbf{i} - \text{sg}(\mathbf{z})\|_2^2. \tag{10}$$

The embedding vectors are not updated using the above loss function. Instead, the embedding vectors are updated through EMA as follows:

$$\mathbf{m}_k^{(\tau)} := \lambda \cdot \mathbf{m}_k^{(\tau-1)} + (1-\lambda) \cdot \sum_{\mathbf{i} \in \mathcal{V}_k'[\tau]} \mathbf{i}, \tag{11a}$$

$$N_k^{(\tau)} := \lambda \cdot N_k^{(\tau-1)} + (1-\lambda) \cdot n_k^{(\tau)}, \tag{11b}$$

$$\mathbf{e}_k^{(\tau)} := \frac{\mathbf{m}_k^{(\tau)}}{N_k^{(\tau)}}, \tag{11c}$$

where τ is iteration index. λ is decaying factor and we set $\lambda = 0.99$. The helper variable $\mathbf{m}_k^{(\tau)}$ represents the EMA accumulation. $n_k^{(\tau)} = |\mathcal{V}_k^{'}[\tau]|$ is the cardinality of Voronoi cell defined as:

$$\mathcal{V}_k^{'}[\tau] = \mathcal{V}^{(\tau)}(\mathbf{e}_k^{(\tau-1)}) = \{\mathbf{i}^{(\tau)} \mid \forall l \neq k, \\ ||\mathbf{i}^{(\tau)} - \mathbf{e}_k^{(\tau-1)}||_2^2 \leq ||\mathbf{i}^{(\tau)} - \mathbf{e}_l^{(\tau-1)}||_2^2\}.$$

Finally, $N_k^{(\tau)}$ is the normalization factor, which is calculated in terms of number of samples used to update $\mathbf{m}_k^{(\tau)}$ in Eq. (11a).

In [8], EMA and EL methods are listed as two separate methods. However, according to our derivation listed in the following, the EMA method can be transformed into the form of EL gradient descent with an adaptive learning rate.

Proposition 1. *Define* $\mathcal{L}_{\text{emb}} = \|\mathbf{z} - \text{sg}(\mathbf{i})\|_2^2$*, the EMA update Eqs.* (11a)*,* (11b)*, and* (11c) *can be written as:*

$$\mathbf{e}_k^{(\tau)} \approx \mathbf{e}_k^{(\tau-1)} - \alpha^{(\tau)} \cdot \frac{\partial \mathcal{L}_{emb}}{\partial \mathbf{e}_k}, \tag{12}$$

with $\alpha^{(\tau)} = (1-\lambda) \cdot \frac{n_k^{(\tau)}}{2 \cdot N_k^{(\tau)}}$.

Proof. From Eqs. (11a), (11b), and (11c), we can obtain (see Appendix):

$$\mathbf{e}_k^{(\tau)} = \mathbf{e}_k^{(\tau-1)} - \\ (1-\lambda) \cdot \frac{n_k^{(\tau)}}{2 \cdot N_k^{(\tau)}} \left[\frac{1}{n_k^{(\tau)}} \sum_{\mathbf{i}_j \in \mathcal{V}_k^{'}[\tau]} 2 \cdot (\mathbf{e}_k^{(\tau-1)} - \mathbf{i}_j) \right]. \tag{13}$$

According to Eq. (6), the embedding vector $\mathbf{e}_k$ is only affected by encoder outputs in corresponding Voronoi set. For the τ-th iteration, the gradient can be approximated as follows:

$$\frac{\partial \mathcal{L}_{emb}}{\partial \mathbf{e}_k} \approx \frac{1}{n_k^{(\tau)}} \sum_{\mathbf{i}_j \in \mathcal{V}_k^{'}[\tau]} 2 \cdot (\mathbf{e}_k^{(\tau-1)} - \mathbf{i}_j). \tag{14}$$

With the adaptive learning rate defined as $\alpha^{(\tau)} = (1-\lambda) \cdot \frac{n_k^{(\tau)}}{2 \cdot N_k^{(\tau)}}$, the update of embedding vector $\mathbf{e}_k$ can be written in the form of Eq. (12). ■

Hence, based on our derivation, the EMA method can be regarded as a gradient descent with adaptive learning rate which depends on the number of samples included in the the corresponding Voronoi set. The influence of the EMA method on the distribution of latent variables and corresponding embedding vectors will be illustrated and explained in the simulation section.

C. Embedding Initialization

The vector quantization described by Eq. (6) can be regarded as a clustering problem. In this context, embedding vectors act as centroids and encoder outputs as data points. For the clustering problem, centroids need to be initialized sufficiently dispersed within the data points distribution to capture underlying patterns [13].

In our work, we propose a method called PCA binary splitting (PBS) to initialize centroids. The data points space is iteratively split in a binary manner into K segments. The center points of these segments are then used to initialize embedding vectors. For each time of splitting, N data points are denoted as $\mathbf{I} = [\mathbf{i}_1, ..., \mathbf{i}_N] \in \mathbb{R}^{D \times N}$. At first, the covariance matrix $\mathbf{C} \in \mathbb{R}^{D \times D}$ of $\mathbf{I}$ will be calculated as:

$$\mathbf{C} = \frac{1}{N}(\mathbf{I}' \cdot \mathbf{I}'^T), \tag{15}$$

where $\mathbf{I}' = [\mathbf{i}_1 - \bar{\mathbf{i}}, ..., \mathbf{i}_N - \bar{\mathbf{i}}]$ represent the zero-meaned data points and $\bar{\mathbf{i}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{i}_n$ is the average of the data points. The covariance matrix $\mathbf{C}$ can be decomposed into eigenvectors and eigenvalues as:

$$\mathbf{C} = \mathbf{Q} \cdot \mathbf{\Lambda} \cdot \mathbf{Q}^{-1}, \tag{16}$$

where $\mathbf{Q} = [\mathbf{q}_1, ..., \mathbf{q}_D] \in \mathbb{R}^{D \times D}$ contains eigenvectors of $\mathbf{C}$ as its columns and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues.

The principal axis $\mathbf{q}'$ is selected as the eigenvector from $\mathbf{Q}$ with the largest corresponding eigenvalue and it represents the direction in the feature space along which the data exhibits the maximum variance. Then the projection of average of data points on the principal axis can be regarded as the reference to split. The segment function is defined as:

$$f_{\text{seg}}(\mathbf{i}_n) = \begin{cases} 0 & \text{if } \mathbf{i}_n^T \cdot \mathbf{q}' < T \\ 1 & \text{if } \mathbf{i}_n^T \cdot \mathbf{q}' \geq T \end{cases}, \text{ and } T = \bar{\mathbf{i}}^T \cdot \mathbf{q}' \tag{17}$$

where T is the projection of the average of data points on the principal axis. The projections of data points, $\mathbf{i}_n^T \cdot \mathbf{q}'$ are compared with T to split the data space equally.

The number of embeddings K is a power of 2 to maximize transmission efficiency. After $\log_2 K$ iterative segmentation steps, we can obtain K segments, denoted as $\mathcal{S}_1, \ldots, \mathcal{S}_K$. The average points of the segments, $\bar{\mathbf{i}}_k$ for $k = 1, ..., K$, are used to initialize embedding vectors $\mathbf{e}_k$:

$$\mathbf{e}_k = \bar{\mathbf{i}}_k = \frac{1}{|\mathcal{S}_k|} \sum_{\mathbf{i} \in \mathcal{S}_k} \mathbf{i} \quad \text{for } k = 1, ..., K \tag{18}$$

However, during the training phase of VQ-VAE, not only centroids (embedding vectors) are updated, but also data points (encoder outputs) are updated. Therefore, centroids need to be re-initialized repeatedly to guarantee the centroids are properly distributed in the data points space. As listed in Algorithm 1, the first N_{init} epochs are regarded as initialization phase during which frequent re-initialization will occur, and embedding vectors are re-initialized every N_{interval} epochs.

D. Complexity Analysis

The computational complexity of VQ-VAE is analyzed across the three main components: the encoder, the VQ layer, and the decoder. Taking the two-layers encoder and decoder as an example, the first encoder layer and the last decoder layer have the same dimension of L, the hidden layers in encoder and decoder have G neurons. Since the encoder and the decoder have symmetric structures, the complexities of both sides are the same: $\mathcal{O}(L \cdot G + G \cdot D)$. VQ layer calculates distances between a latent variable and K embedding vectors

Algorithm 1 VQ-VAE Training with PBS Re-initialization

Inputs: Training set $\mathcal{X}$.
Initialize: $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_K]$ initialized randomly
for $\tau = 1$ to N_{epochs} **do**
    update encoder, decoder, and embedding vectors using Eq. (9) or EMA method (11a), (11b), and (11c).
    if $\tau \mod N_{\text{interval}} = 0$ and $\tau < N_{\text{init}}$ **then**
        Calculate $\mathcal{P} = f_{\text{en}}(\mathcal{X})$
        Re-initialize embedding vectors using PBS, using Eqs. (15) to (18).
    end if
end for

with the dimension of D, therefore, the complexity for the VQ-layer is $\mathcal{O}(K \cdot D)$. The overall complexity of VQ-VAE can be represented as $\mathcal{O}(2 \cdot L \cdot G + 2 \cdot G \cdot D + K \cdot D)$.

The complexity of the PBS algorithm primarily arises from two components: the calculation (Eq. (15)) and the decomposition (Eq. (16)) of the covariance matrix. The calculation of the covariance matrix involves of matrix multiplication of $\mathbf{I}' = \mathbb{R}^{D \times N}$ and its transpose, so the complexity is $\mathcal{O}(D^2 \cdot N)$. The eigendecomposition of the covariance matrix $\mathbf{C} \in \mathbb{R}^{D \times D}$ entails a complexity of $\mathcal{O}(D^3)$. The total complexity of the PBS algorithm is $\mathcal{O}(D^3 + D^2 \cdot N)$. PBS operates $\log_2 K$ iterations until we have K segments. Each iteration divides the number of samples N by 2, but increases the times of segmentations by 2. Hence, the total complexity of the iterative PBS algorithm is $\sum_{a=0}^{(\log_2 K)-1} \mathcal{O}(2^a \cdot (D^3 + D^2 \cdot \frac{N}{2^a}))$.

E. Prediction Method

As depicted in Fig. 1, to predict future changes in SINR, the SINR sequence $\mathbf{y}_t = [\gamma_{t+1}, \gamma_{t+2}, ..., \gamma_{t+M}]^T$ is predicted on the BS side after recovering $\hat{\mathbf{x}}$. Long short-term memory (LSTM) is employed for sequence-to-sequence prediction, as demonstrated in [14]. In this work, a single layer LSTM model is utilized to predict SINR sequences as follows:

$$\hat{\mathbf{y}}_t = f_{\text{LSTM}}(\hat{\mathbf{x}}_t). \tag{19}$$

It is noteworthy that during the training of the LSTM model, we deliberately use the true sequence, $\mathbf{x}$, rather than the reconstructed sequence, $\hat{\mathbf{x}}$, to ensure that the LSTM model learns directly from the original data, preserving the inherent temporal dependencies and patterns of data.

IV. Numerical Results

A. Simulation Settings

An SINR sequence is generated following Eq. (1). It is assumed that three interference sources move around the UE in a 20 m $\times$ 20 m two-dimensional space with random directions and the speed of 2 m/s. The sampling frequency is set to $1\,\text{kHz}$. Parameters for interference scenarios are provided in Table I. Noise power is much lower than BS and interference transmission power, because here it is at the receiver end and will not experience any channel attenuation. In [15], our implementation codes are provided and further simulation details can be found in the repository.

TABLE I
Simulation Settings for Interference Scenario

Parameters	Values
Interferer Model	
Number of Interfers	3
Velocity of Interfers	2 m/s
Mobile model	Random Directional
Channel Model	
Path loss exponent	2.5
Shadow fading variance	5
Carrier frequency	3 GHz
Small Scale fading	Jakes fading model [10]
Normalized Power	
BS Tx power	1
Interferer Tx Power	0.02
Noise Rx Power	3e-4

The autoencoder features a compact structure with two dense layers in both the encoder and decoder. The structures of encoder and decoders are listed in Table II. The length of sequence $\hat{\mathbf{x}}_t$ is $L = 40$, therefore, the shape of the input layer of the encoder is also 40. The embedding space of vector quantization layer consists of 64 embedding vectors and the number of quataiztion bits is $\log_2 64 = 6$. The number of prediction steps is set to $M = 10$. A single layer LSTM model is adopted for SINR prediction.

According to the selection of loss function and the use of PBS initialization methods, four VQ-VAE algorithms are listed in Table III. For the algorithms using PBS, we set $N_{\text{init}} = 100$ and $N_{\text{interval}} = 20$, which means that embedding vectors are re-initialized every 20 epochs in the first 100 training epochs.

TABLE II
Network Structure of Encoder, Decoder, and LSTM

	Type	Params #	Output shape	Activation
Encoder	Input	0	40	None
	Dense	820	20	ReLU
	Dense	420	20	ReLU
Decoder	Dense	420	20	ReLU
	Dense	840	40	Linear
LSTM	LSTM	16896	64	Tanh
	Dense	650	10	Linear

TABLE III
VQ-VAE methods Variants

	EMA	EL	No Re-Init.	PBS Re-Init.
VQ-VAE-EMA	✓		✓	
VQ-VAE-EL		✓	✓	
VQ-VAE-EMA-PBS	✓			✓
VQ-VAE-EL-PBS		✓		✓

B. Numerical Results

The reconstruction loss is illustrated in Fig. 4. VQ-VAE-EL-PBS has the lowest reconstruction loss and the smallest variance. We also observe that EL-based methods exhibit faster convergence than EMA-based methods. Furthermore, when comparing methods with PBS re-initialization and those without, PBS is able to reduce the reconstruction loss. The advantage brought by PBS to EL methods is more obvious than that to EMA methods.

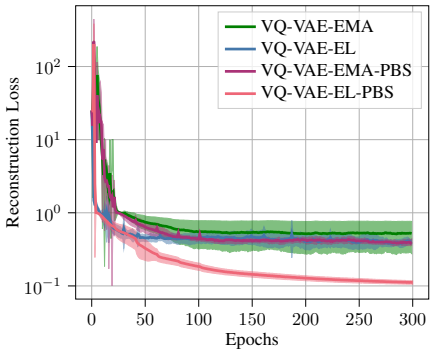

Fig. 4. Reconstruction loss of VQ-VAE algorithms. VQ-VAE-EL-PBS exhibits the lowest reconstruction loss, while the other three algorithms demonstrate similar levels of reconstruction loss.

Fig. 5 compares the number of updated embedding vectors. Methods with PBS re-initialization are able to use all 64 embedding vectors during the training process, but VQ-VAE-EL-PBS can utilize embedding space fully after only 20 epochs, much faster than VQ-VAE-EMA-PBS. For VQ-VAE-EMA and VQ-VAE-EL, not all embedding vectors can be used by the end of training, leading to the under-utilization of the embedding space and reducing the ability to capture patterns of sequences in the compact embedding space.

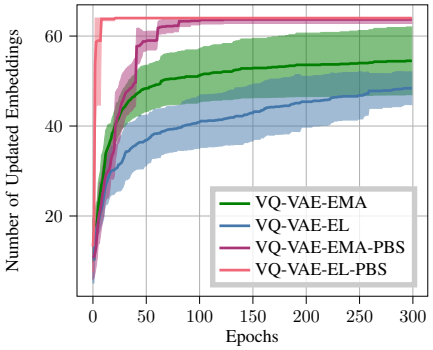

Fig. 5. Number of updated embedding vectors of VQ-VAE algorithms. PBS-based algorithms are able to utilize all 64 embedding vectors. Additionally, VQ-VAE-EMA is capable of utilizing more embedding vectors than VQ-VAE-EL

Besides the number of embedding vectors, we are also interested in the distribution of the latent variables on the embedding vectors. The entropy of the latent variables from different methods are illustrated in Fig. 6. The entropy is calculated as: $H = -\sum_{k=1}^{K}(p_k \log p_k)$, where $p_k = \frac{|\mathcal{V}_k|}{\sum_{j=1}^{K}|\mathcal{V}_j|}$ is the probability of latent variables assigned to k-th embedding vector and $\mathcal{V}_k$ is the Voronoi set defined in Eq. (7). Interestingly, even though both PBS methods have utilized all the embedding vectors, VQ-VAE-EL-PBS has higher entropy and the latent variables are distributed more evenly. Therefore, it is easier for VQ layers to capture underlying patterns.

Based on the above comparison, we can observe that the EMA with PBS method does not perform as well as EL with PBS does. As derived in proposition 1, the EMA method has a learning rate dependent on the number of samples assigned to

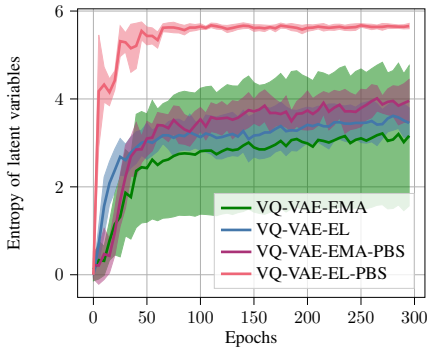

Fig. 6. Entropy of latent variables in VQ-VAE models. VQ-VAE-EL-PBS exhibits the highest entropy, allowing its VQ layers to capture underlying patterns more effectively. Consequently, VQ-VAE-EL-PBS also achieves the lowest reconstruction loss.

embedding vectors. The non-consistent learning rates would results in the concentration of latent variables around a few more frequently updated embedding vectors and the uneven distribution of latent variables for EMA, as also reflected in Fig. 6.

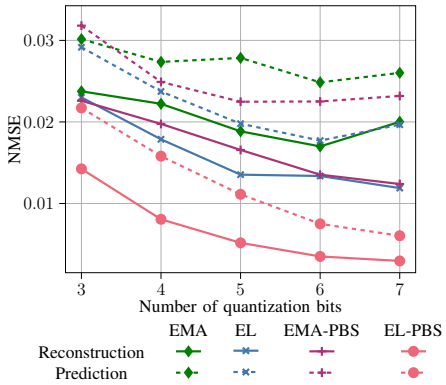

Fig. 7. Reconstruction and Prediction NMSE versus the number of quantization bits. Solid lines represent reconstructions and dash lines represent predictions. VQ-VAE-EL-PBS achieves the lowest NMSE for both reconstruction and prediction.

Fig. 7 presents a comparison of the normalized mean squared errors (NMSE) of the reconstruction and prediction, calculated as $\frac{\mathbb{E}\{\|\hat{\mathbf{x}}-\mathbf{x}\|_2^2\}}{\mathbb{E}\{\|\mathbf{x}\|_2^2\}}$ and $\frac{\mathbb{E}\{\|\hat{\mathbf{y}}-\mathbf{y}\|_2^2\}}{\mathbb{E}\{\|\mathbf{y}\|_2^2\}}$, respectively. It is observed that the VQ-VAE-EL-PBS model achieves the lowest NMSE for both reconstruction and prediction tasks. Furthermore, the ranking of models based on reconstruction NMSE aligns with their ranking based on prediction NMSE. This consistency underscores the critical importance of effective compression and quantization to achieve accurate predictions.

For link adaptation algorithms, according to the system requirements, the required accuracy for prediction is also different. In Figure 8, the required numbers of quantization bits to achieve prediction NMSE below the target NMSE are compared for VQ-VAE-EL-PBS and Diff-Quant [9]. Diff-Quant is the common quantization method in 3GPP standardization; it first takes samples from the sequence and then quantizes the differences between adjacent samples. According to [9], we set

the number of quantization bits for a single sampling point to 4 bits and the quantization step to 1 dB. Therefore, the number of quantization bits for Diff-Quant must be a multiple of 4. In this comparison, we assume that the number of samples can be adjusted to meet various system requirements. After reconstruction from Diff-Quant, we use the same LSTM model to perform the prediction and compare it to VQ-VAE-EL-PBS. We can observe that the VQ-VAE-EL-PBS scheme is able to significantly reduce the required number of quantization bits. It is worthwhile to mention that when the number of bits is set to 40 for Diff-Quant, the NMSE decreases directly to below 0.015, skipping the range $[0.015, 0.020]$, so the number of bits remains at 40 for both target NMSE values of 0.015 and 0.02.

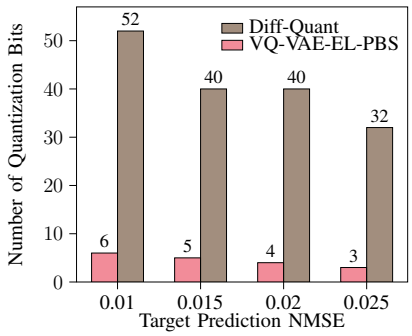

Fig. 8. Comparison of Required Number of Quantization for VQ-VAE-PBS and Diff-Quant. Compared to Diff-Quant, VQ-VAE-EL-PBS can achieve the same prediction NMSE range while requiring significantly fewer quantization bits

V. Conclusion

In this work, we introduce a SINR sequence feedback scheme based on VQ-VAE, enabling SINR sequence prediction at the BS side. To enhance the efficiency of compression algorithms, we incorporate the PBS re-initialization algorithm with VQ-VAE to optimize the distribution of the embedding space. Our numerical simulations demonstrate the advantages of the VQ-VAE approach for sequence compression and prediction. By incorporating PBS, we achieve even greater reductions in feedback overhead and a significant improvement in reconstruction and prediction accuracy. In our future work, we plan to expand the scenarios to include MIMO systems and integrate our approach with the link adaptation problem.

Appendix
Derivation of Equation (13)

The Eq. (11c) can be rewritten as:

$$\begin{aligned}\mathbf{e}_k^{(\tau)} =& \frac{1}{N_k^{(\tau)}} \cdot \left(\lambda \cdot \mathbf{m}_k^{(\tau-1)} + (1-\lambda) \cdot \sum_j^{n_k^{(\tau)}} \mathbf{i}_{k,j} \right). \\ =& \frac{1}{N_k^{(\tau)}} \cdot \left(\lambda \cdot N_k^{(\tau-1)} \cdot \mathbf{e}_k^{(\tau-1)} + (1-\lambda) \cdot \sum_j^{n_k^{(\tau)}} \mathbf{i}_{k,j} \right).\end{aligned}$$

Eq. (11b) indicates $\lambda \cdot N_k^{(\tau-1)} = N_k^{(\tau)} - (1-\lambda) \cdot n_k^{(\tau)}$, and we have:

$$\begin{aligned}\mathbf{e}_k^{(\tau)} =& \frac{1}{N_k^{(\tau)}} \cdot \left(N_k^{(\tau)} \cdot \mathbf{e}_k^{(\tau-1)} + (1-\lambda) \cdot \sum_j^{n_k^{(\tau)}} (\mathbf{i}_{k,j} - \mathbf{e}_k^{(\tau-1)}) \right) \\ =& \mathbf{e}_k^{(\tau-1)} - (1-\lambda) \cdot \frac{1}{N_k^{(\tau)}} \left(\sum_j^{n_k^{(\tau)}} (\mathbf{e}_k^{(\tau-1)} - \mathbf{i}_{k,j}) \right) \\ =& \mathbf{e}_k^{(\tau-1)} - (1-\lambda) \cdot \frac{n_k^{(\tau)}}{2 \cdot N_k^{(\tau)}} \left(\frac{1}{n_k^{(\tau)}} \sum_j^{n_k^{(\tau)}} 2 \cdot (\mathbf{e}_k^{(\tau-1)} - \mathbf{i}_{k,j}) \right).\end{aligned}$$

References

[1] K. I. Pedersen, G. Monghal, I. Z. Kovacs, T. E. Kolding, A. Pokhariyal, F. Frederiksen, and P. Mogensen, "Frequency domain scheduling for ofdma with limited and noisy channel feedback," in *2007 IEEE 66th Vehicular Technology Conference*. IEEE, 2007, pp. 1792–1796.

[2] L. Zhu, C. Bockelmann, T. Schier, S. E. Hajri, and A. Dekorsy, "Nolla: Non-linear outer loop link adaptation for enhancing wireless link transmission," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2023, pp. 1–6.

[3] A. Kuhne and A. Klein, "Throughput analysis of multi-user ofdma-systems using imperfect cqi feedback and diversity techniques," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 8, pp. 1440–1450, 2008.

[4] X. Lin, "An overview of the 3gpp study on artificial intelligence for 5g new radio," *arXiv preprint arXiv:2308.05315*, 2023.

[5] M. Ni, X. Xu, and R. Mathar, "A channel feedback model with robust sinr prediction for lte systems," in *2013 7th European Conference on Antennas and Propagation (EuCAP)*. IEEE, 2013, pp. 1866–1870.

[6] R. Abi Akl, S. Valentin, G. Wunder, and S. Stanczak, "Compensating for cqi aging by channel prediction: The lte downlink," in *2012 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2012, pp. 4821–4827.

[7] X. Xu, M. Ni, and R. Mathar, "Improving qos by predictive channel quality feedback for lte," in *2013 21st International Conference on Software, Telecommunications and Computer Networks-(SoftCOM 2013)*. IEEE, 2013, pp. 1–5.

[8] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[9] 3GPP, "5G NR Physical layer procedures for data," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.214, 2018, version 15.3.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/15.03.00_60/ts_138214v150300p.pdf

[10] P. Dent, G. E. Bottomley, and T. Croft, "Jakes fading model revisited," *Electronics letters*, vol. 13, no. 29, pp. 1162–1163, 1993.

[11] V. Erceg, L. J. Greenstein, S. Y. Tjandra, S. R. Parkoff, A. Gupta, B. Kulic, A. A. Julius, and R. Bianchi, "An empirically based path loss model for wireless channels in suburban environments," *IEEE Journal on selected areas in communications*, vol. 17, no. 7, pp. 1205–1211, 1999.

[12] S. Lu, J. May, and R. J. Haines, "Effects of correlated shadowing modeling on performance evaluation of wireless sensor networks," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*. IEEE, 2015, pp. 1–5.

[13] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for k-modes clustering," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7444–7456, 2013.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[15] L. Zhu, *VQ-VAE SINR compression*, 2024. [Online]. Available: https://github.com/LingruiZhu/VQ_VAE_SINR_compression