Research Article

Remote Inference Over Dynamic Links via Adaptive Rate Deep Task-Oriented Vector Quantization

Eyal Fishel[1], May Malka[1], Nir Shlezinger[1], Shai Ginzach[2]

1. School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel; 2. Rafael Advanced Defense Systems (Israel), Haifa, Israel

A broad range of technologies rely on *remote inference*, wherein data acquired is conveyed over a communication channel for inference in a remote server. Communication between the participating entities is often carried out over rate-limited channels, necessitating data compression for reducing latency. While deep learning facilitates joint design of the compression mapping along with encoding and inference rules, existing learned compression mechanisms are static, and struggle in adapting their resolution to changes in channel conditions and to dynamic links. To address this, we propose Adaptive Rate Task-Oriented Vector Quantization (ARTOVeQ), a learned compression mechanism that is tailored for remote inference over dynamic links. ARTOVeQ is based on designing nested codebooks along with a learning algorithm employing progressive learning. We show that ARTOVeQ extends to support low-latency inference that is gradually refined via successive refinement principles, and that it enables the simultaneous usage of multiple resolutions when conveying high-dimensional data. Numerical results demonstrate that the proposed scheme yields remote deep inference that operates with multiple rates, supports a broad range of bit budgets, and facilitates rapid inference that gradually improves with more bits exchanged, while approaching the performance of single-rate deep quantization methods.

1. Introduction

As data demands and data diversity grow, digital communication systems are increasingly embracing collaborative networks designed for reliable and task-specific communication. This trend is particularly evident in next-generation technologies such as the Internet of Things and autonomous vehicles, where achieving accurate inference over rate-limited communication channels with low latency is essential[1]. Task-based (or goal-oriented) communication has emerged as a necessary and innovative solution for remote inference systems[2], which tend to operate in two distinct stages. The first stage occurs at the edge or sensing device, where acquired data is conveyed over a rate-limited channel after undergoing compression (source coding) and channel coding[3]. The second stage takes place at the receiver, which extracts the information needed for the task, e.g., classify an image[4].

Separating the processing involved with communicating data from that associated with inference facilitates the design of remote inference systems, and supports implementation on top of existing communication protocols. However, separation also often comes at the cost of notable overhead in communication resources, leading to excessive latency, which is often a crucial factor[5]. This downgrade in performance is a result of the inference task being typically very specific, while the data source is encoded such that it can be entirely recovered, regardless of the task at hand[6]. As such, several studies have attempted to bridge this gap in order to facilitate remote inference over-rate limited links. These include task-based quantization[7][8], semantics-aware coding[9][10], and goal-oriented communications[11][12]. A common characteristic of these works involves encoding the source based on the inference task rather than prioritizing complete signal reconstruction. This approach supports compact representations, which in turn facilitate lower communication latency compared with the separation based designs[13].

Designing task-based compression mechanisms based on statistical models tends to be complicated and is limited to simple tasks that can be represented as linear[7] and quadratic mappings[14][15]. Yet, data-driven approaches have been shown to yield accurate remote inference mechanisms for generic tasks with compact representations. This is achieved by leveraging joint learning of the compression mechanism along with a deep neural network (DNN)-aided inference rule[16][17]. Such designs employ DNN-based encoder-decoder architectures, while constraining the latent features to a fixed bit representation via uniform quantization[18][19][20], scalar quantization[21][22][23], and vector quantization[24][25]. Such forms of neural compression, which were shown to achieve highly compressed representation of image[26], video[27], and audio[28] (see detailed survey in[29]), can be naturally converted into remote inference systems. This is achieved by assigning the encoder and decoder to the sensing and inferring devices, respectively, while training the overall system for the desired inference metric[30].

The majority of DNN-aided compression algorithms operate in a static single-rate manner. Namely, the encoder maps the sensed data into a fixed-length bit sequence, which is then processed by the decoder module[26][31]. In the context of remote inference, this operation induces two notable challenges when communicating over time-varying links: (i) Once trained, the model's compression rate can not be modified, making it difficult for remote inference systems to adapt to changing channel conditions. Consequently, the system must either adopt a worst-case compression rate, increasing latency, or maintain multiple encoder-decoder model's for different rates, adding complexity. (ii) Inference only begins after all the compressed features arrive and are decoded at the inferring device, which has to wait for the entire bit sequence representation to be received before it can provide any form of output. These limitations highlight the need for DNN-aided remote inference systems that can operate at different rates and perform inference with minimal latency, ideally starting as soon as the first bits are received.

Several studies have proposed DNN-aided compression methods that are not subject to (i) and/or (ii), while focusing on task-invariant compression, i.e., when the decoder recovers the sensed data (typically an image). The first family of multi-rate methods is that of *variable-rate* DNN-aided compression, which still require the complete bit sequence to be received for decoding (thus still subject to (ii)), but can operate with different bit rates[32][33][34][35][36][37][38][39]. The encoder and decoder can be designed to operate with different rates by using multi-scale[32], conditional[33], modulated[34], and slimmable[35] encoder-decoder architectures, or alternatively by masking the latent features[36][37] or integrating adaptive normalization[38]. While all these works utilized uniform scalar quantizers for quantization,[39] proposed a variable rate compression mechanism that uses vector quantization by training an external Seq2Seq model to generate the codebook on demand. The second family of multi-rate DNN-aided quantization methods is based on *progressive* compression[40][41][42][43][44]. This is typically achieved by using recurrent neural network (RNN) based encoders[40][41][42], which at each step reconstruct the input and encode the residual, such that when each RNN output is decoded, an additional residual term is obtained. Alternative approaches to DNN-aided progressive compression transform the input into a set of features ordered by importance. These features are fed into uniform scalar quantizers, whose output is used by the decoder to recover the input with growing accuracy[43][44].

Despite advancements, current multi-rate DNN-aided compression methods have several limitations in the context of remote inference. Specifically, while multi-rate methods can be adapted to a task-based setting by replacing the decoder with a DNN-aided inference rule, existing methods do not extend to a progressive operation. This is partially due to the fact their focus is mostly on the encoder-decoder architecture, employing simple uniform scalar mappings for quantization (with the exception of[39], which supports adjustable vector quantizers at the cost of excessive complexity in runtime, and without enabling progressive operation). Existing progressive DNN-aided compression methods are highly geared towards a non-task-based setting, where the decoder recovers the input, and progressive operation is obtained by gradual compression of additional features and residual terms that are informative of the input. While one can potentially still employ such architecture in a task-based setting by inferring based on the separately recovered input, such separation-based approaches are known to be inefficient in task-based quantization[4].

In this work, we tackle the aforementioned gap by designing ARTOVeQ, a multi-rate DNN-aided remote inference scheme that naturally supports a progressive operation. ARTOVeQ is based on a remote inference model that uses a trainable adaptive vector quantization, allowing data compression and inference at multiple rates while using the same underlying architecture. Inspired by nested quantization techniques, we introduce a high-resolution quantization codebook that can be successively decomposed into sub-codebooks of lower resolution[45][46]. The usage of such nested-style learned codes naturally extends to a progressive operation, where compression is carried out as a form of successive refinement[47]. This approach supports multi-rate quantization with a single codebook, thereby providing an adaptable and economical solution for varying communication environments. As our focus is on the learned codebook rather than on the encoder and decoder architecture, ARTOVeQ can be combined with existing DNN-aided compression mechanisms.

Our main contributions are summarized as follows:

- **Rate-adaptive learned task-oriented vector quantization:** We propose a multi-rate task-based vector quantizer that extends the established Vector Quantization Variational Autoencoder (VQ-VAE) model[24], which supports remote inference with multi-rate learned vector quantization. Our ARTOVeQ learns a single codebook that subsumes lower-rate codewords, trained via a progressive learning scheme[48] that ensures the remote user (decoder) can reliably infer at various rates, providing a sense of adaptability and reliability.
- **Mixed resolution implementation:** To support a broad range of bit rates with fine granularity we formulate a mixed-resolution implementation of the task-based quantizer. In this model, different

features are quantized with different subsets of the learned codebook, thus spanning various different bit rates, while having the encoder learn to map relevant information into features that are quantized with higher resolution.

- **Progressive quantization via successive refinement:** We extend the learned multivariate codebook of ARTOVeQ to represent nested codewords, that are naturally applicable in a progressive manner. This allows the multi-rate DNN-aided remote inference system to provide predictions of the task with the first codeword received, while gradually improving its reliability with each incoming bit.
- **Extensive experimentation:** We extensively evaluate our proposed ARTOVeQ for remote image classification, using the popular edge-oriented MobilenetV2 architecture[49] for the encoder-decoder. Our experiments, which use the CIFAR-100 and Imagewoof datasets, demonstrate that the proposed scheme results in a single model which for all considered rates approaches the performance of multiple single-rate VQ-VAE models, each optimized for a specific rate, while benefiting from mixed-rate implementation and extending to progressive operation with only a minor performance impact.

The rest of this paper is organized as follows: Section II reviews the system model and some preliminaries; our rate-adaptive remote inference scheme is presented in Section III and evaluated in Section IV. Section V concludes the paper.

Throughout this paper, we use boldface-uppercase for matrices, e.g., $\mathbf{X}$, and boldface-lowercase for vectors, e.g., $\mathbf{x}$. We denote the jth entry of vector $\mathbf{x}$ and the (i,j)th entry of matrix $\mathbf{X}$ by $[\mathbf{x}]_j$ and $[\mathbf{X}]_{i,j}$, respectively. We use $\|\|$, and $\mathbb{E}[\cdot]$ for the ℓ_2 norm, and stochastic expectation, respectively.

II. System Model and Preliminaries

In this section, we review some essential preliminaries and present the system model under consideration. We begin by reviewing basic quantization principles in Subsection II-A. Then, we formulate our remote inference problem in Subsection II-B, and discuss existing mechanisms for DNN-aided remote inference in Subsection II-C.

A. Quantization

Quantization is concerned with the representation of a continuous-valued signal using a finite number of bits[50]. The discrete representations produced through quantization should in general effectively represent signals, even at low resolution, while maintaining acceptable reconstruction performance.

Formally, a quantizer, denoted by $\mathcal{Q}_S^{n,k}(\cdot)$ is a mapping from continuous-valued inputs in $\mathbb{R}^n$ into discrete-valued outputs in $\mathcal{Q} \subset \mathbb{R}^k$ using $\log_2 S$ bits. The set $\mathcal{Q}$, whose cardinality is $|\mathcal{Q}| = S$, represents the *quantization codebook*. This codebook defines the set of possible discrete outputs, forming the basis for the two-stage quantization mapping: Initially, an encoding function maps the continuous input $\mathbf{x} \in \mathbb{R}^n$ into a discrete set $\{1, 2, \ldots, S\}$. Then, a decoding function maps each item in this discrete set into an associated codeword. Conventionally, $n = k$, and the codeword constitutes a reconstruction of the input. However, in *task-based quantization*, the codeword represents some desired information that must be extracted from the input, and thus k can differ from n [7]. When $n = 1$, the quantizer is *scalar*, while $n > 1$ denotes a *vector quantizer*.

B. Problem Formulation

We consider a remote inference setting comprised of a sensing device and an inferring user. At time t, the sensing device captures an input data sample $\mathbf{x}_t$, which is conveyed to the inferring user for providing a prediction $\hat{y}_t$. For instance, $\mathbf{x}_t$ can represent an image captured at a remote camera, while $\hat{y}_t$ is the predicted class of the content of the image. The users communicate over a rate-limited channel which is modeled as a bit-pipeline with time-dependent capacity, denoted as C_t, measuring bits per time unit [25]. Consequently, the latency required to transmit B_t bits at time t is given by $\tau_t = \frac{B_t}{C_t}$. The system is illustrated in Fig. 1.

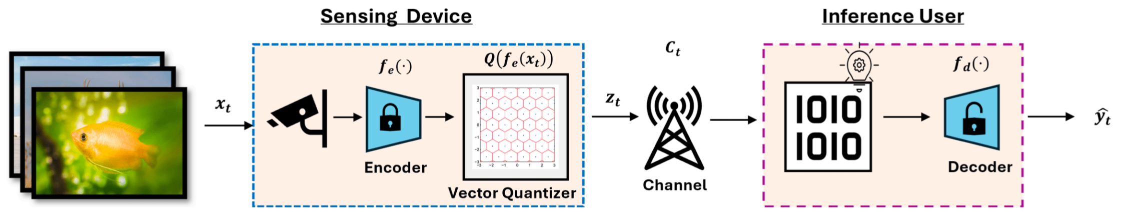


Figure 1. Remote inference system illustration

For conveying $\mathbf{x}_t$, a quantization mechanism is employed, consisting of: (i) an encoder at the sensing device, denoted $f_e(\cdot)$, that maps $\mathbf{x}_t$ into a B_t bits representation denoted $\mathbf{z}_t$; and (ii) a decoder $f_d(\cdot)$ implementing a decision rule at the inferring user that outputs $\hat{y}_t$ based on $\mathbf{z}_t$. It is assumed that the sensing device knows the current channel capacity C_t, and that the capacity has some lower bound $C_{\min} > 0$.

We focus on a data-driven setting. During design, one has access to a data set consisting of labeled examples $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, that is, N pairs of inputs and desired outputs for design purposes. Our goal is to design a remote inference system based on two performance measures:

P1 Accuracy of the predictions $\hat{y}_t$, where we specifically focus on classification tasks;

P2 Latency of the inference procedure, which we constrain to be at most $\tau_{\max}$ (with $\tau_{\max} \geq \frac{1}{C_{\min}}$).

In principle, the sensing device can be designed to carry out the complete inference procedure. However, we concentrate on the common setting in which only partial pre-processing can be applied due to, e.g., hardware limitations[1].

C. DNN-Aided Remote Inference

A natural approach to design data-driven remote inference system is to partition a DNN suitable for the task at hand between the sensing and inferring devices, resulting in a trainable encoder-decoder model[51]. However, compressing the latent representation using a finite number of bits poses a problem owing to the non-differentiable nature of continuous-to-discrete mappings, and the desire to adjust the bit rate based on channel variations to meet latency constraints. This limits the ability to jointly learn the encoder and decoder mappings using conventional gradient-based deep learning tools. As such, various solutions have been proposed, including modeling scalar quantizers as additive noise during training[18][20], and soft-to-hard approximations[22][31].

Another approach to bypass the non-differentiable step is to use straight-through gradient estimators. A well-known example of this approach is the well-established VQ-VAE[24], illustrated in Fig. 2. Gradients are passed through the quantization step, allowing for joint learning of the encoder, codebook, and the decoder, despite the non-differentiable nature of the quantization. This joint optimization forms the foundations for the VQ-VAE model, which consists of three components: a DNN encoder, $f_e(\cdot)$, a quantization codebook, $\mathcal{Q}$, and a DNN decoder $f_d(\cdot)$. The codebook $\mathcal{Q}$ is comprised of $|\mathcal{Q}| = S$ vectors of size d. The input sample, $\mathbf{x}_t$, is processed by the encoder into $\mathbf{x}_t^e = f_e(\mathbf{x}_t)$ which serves as a low-dimensional representation of the input. Subsequently, the vector $\mathbf{x}_t^e$ is decomposed into M vectors of size d, denoted $\{\mathbf{x}_{t,m}^e\}_{m=1}^{M}$, and each is represented by the closest codeword in $\mathcal{Q}$. Thus, the latent representation $\mathbf{z}_t$ is the stacking of

$$\mathbf{z}_{t,m} = \arg\min_{\mathbf{e}_j \in \mathcal{Q}} \|\mathbf{x}^e_{t,m} - \mathbf{e}_j\|_2^2. \tag{1}$$

The quantized $\mathbf{z}_t$ is processed by the decoder into $\hat{y}_t = f_d(\mathbf{z}_t)$, and the number of bits conveyed is $B_t = M \cdot \log_2 S$.

To jointly train the encoder-decoder while learning the codebook $\mathcal{Q}$, the VQ-VAE uses a loss function comprised of three terms as follows:

$$\mathcal{L}_{tot}(y_t; \mathbf{x}_t) = \mathcal{L}(y_t; \hat{y}_t) + \|sg(\mathbf{x}^e_t) - \mathbf{z}_t\|_2^2 + \beta\|\mathbf{x}^e_t - sg(\mathbf{z}_t)\|_2^2, \tag{2}$$

where $sg(\cdot)$ is the stop-gradient operator. The first term in (2), $\mathcal{L}(y_t; \hat{y}_t)$, is the task-dependent loss, (e.g., cross entropy for classification). The second term is the VQ-loss, which moves the codebook vectors closer to the encoder outputs. The third term is the commitment loss, which causes the encoder outputs to be similar to the codebook vectors. The hyperparameter $\beta > 0$ balances the influence of the commitment loss on $\mathcal{L}_{tot}$. While alternative loss measures have been recently proposed for training the VQ-VAE to boost improved utilization of its codebook[52][53], (2) is to date the common loss used for training such DNN-aided vector quantizers, see[54][55]. The loss in (2) is stated for a given codebook size S, resulting in a model that is fixed to a given bit budget B_t.

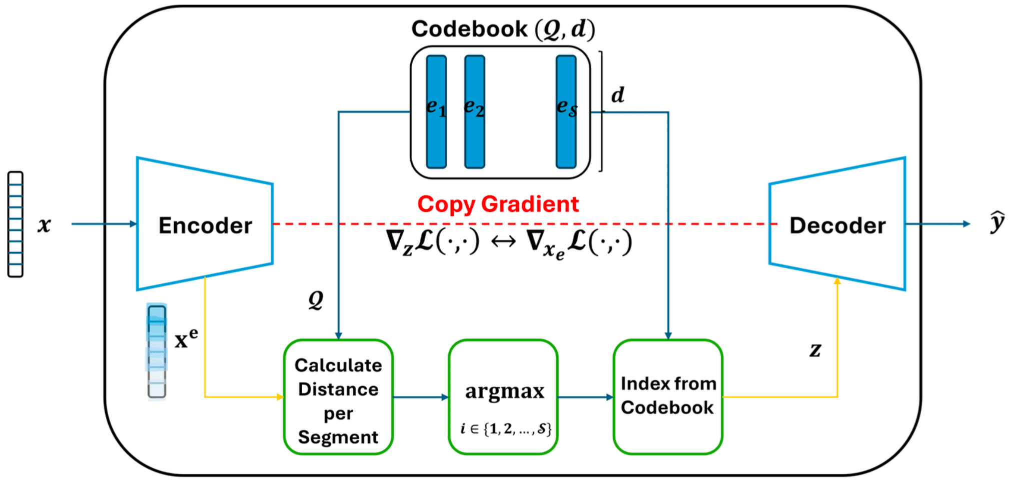

Codebook $(\mathcal{Q}, d)$
e_1
e_2
e_S
d
x
Encoder
Copy Gradient
$\nabla_z\mathcal{L}(\cdot,\cdot) \leftrightarrow \nabla_{x_e}\mathcal{L}(\cdot,\cdot)$
Decoder
$\hat{y}$
$\mathcal{Q}$
$\mathbf{x}^e$
z
Calculate Distance per Segment
argmax
$i \in \{1, 2, \ldots, S\}$
Index from Codebook

Figure 2. VQ-VAE architecture. The encoder maps the input $\mathbf{x}$ into the features $\mathbf{x}^e$, which is divided into M sub-vectors of size $d \times 1$. Each sub-vector undergoes the vector quantization mechanism, which selects an embedding based the distance from the codebook vectors. The decoder is applied to the collection of quantized sub-vectors for inference.

III. ARTOVeQ

In this section we introduce the proposed ARTOVeQ, designed for remote inference over dynamic channels as formulated in Subsection II-B. We commence by detailing its high level rationale in Subsection III-A, after which we present its trainable rate-adaptive vector codebook in Subsection III-B. We then show in Subsections III-C-III-D how the design of ARTOVeQ naturally extends to support multi-rate and progressive quantization, respectively, with a single codebook. We conclude with a discussion provided in Subsection III-E

A. High Level Rationale

The VQ-VAE algorithm of [24], recalled in Subsection II-C, can be used for high performance remote inference (in the sense of *P1*) when employed over a static channel (in which the capacity and latency constraints, dictating the bit budget B_t, are fixed), owing to its ability to learn task-oriented vector quantization codebooks. Nonetheless, its application for remote inference is not suitable for dynamic channels, as it cannot adapt its bit rate to the the channel conditions. Moreover, its operation is non-progressive, i.e., the decoder needs to receive all bits representing the codeword for inference, which limits its minimal inference latency (*P2*).

Our proposed ARTOVeQ builds on the ability of VQ-VAE to learn task-oriented multi-resolution codebooks, while overcoming its lack of flexibility and progressiveness by handling a codebook that accommodates multiple-bit resolutions. This is achieved by incorporating the following aspects:

A1 A single codebook $\mathcal{Q}$ is designed to support all multi-level bit resolutions by restricting it to be decomposable into sub-codebooks that are used for reduced bit rates.

A2 A dedicated training algorithm is proposed, which combines principled initialization for vector quantization based on the Linde–Buzo–Gray (LBG) algorithm[56], alongside a gradual learning mechanism that allows the same decoder to be reused with all sub-codebooks.

A3 By further restricting the learned codebook to take the form of nested vector quantization[45], we enable a progressive operation, where on each incoming bit the decoder can successively refine its predication.

In the following subsections we design ARTOVeQ by gradually incorporating *A1*-*A3* into its design.

B. Rate-Adaptive Learned Vector Codebook

Here, we design a VQ-VAE-based architecture that enables multi-rate vector quantization, thus meeting *A1*, and present a training algorithm that enables rate adaptive task-oriented quantization following *A2*.

1. Architecture

Using the VQ-VAE architecture outlined in Subsection II-C, which is generic in the sense that it is invariant of the specific DNNs used for the encoder and decoder, we construct a single codebook that accommodates multiple resolutions by iteratively doubling its number of codewords. Specifically, for each quantization level, $l = 1, 2, \ldots \log_2 S$, up to some maximum compression rate $S = |\mathcal{Q}|$, a dedicated codebook is maintained $\mathcal{Q}_l$. The process begins with constructing the 1-bit resolution codebook, followed by the 2-bit resolution codebook, and so forth, until the maximum compression rate is reached. This design guarantees that

$$\mathcal{Q}_1 \subset \mathcal{Q}_2 \subset \cdots \subset \mathcal{Q}_{\log_2 S}. \tag{3}$$

From (3) it follows that the first two codewords are derived from $\mathcal{Q}_1$; the first four are derived from $\mathcal{Q}_2$; and so on. The users thus manage a unified codebook encompassing all quantization resolutions, avoiding storing individual codebooks for each resolution.

As new samples $\mathbf{x}_t$ become available to the sensing device, the quantization level is initially determined using

$$l_t = \max\left\{ l \in \{1, 2, \ldots \log_2 S\} \mid \frac{l}{M \cdot C_t} \leq \tau_{\max} \right\}. \tag{4}$$

After determining the quantization level, remote inference is performed on the discrete outputs at the central server.

2. Training

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, the training algorithm sets the encoder $f_e(\cdot)$, the codebook $\mathcal{Q}$, and the decoder $f_d(\cdot)$, through a gradual learning process. This approach is organized in three stages, designed to enable the model to operate at progressively higher resolutions over time while retaining previously acquired knowledge.

Stage 1: Encoder-Decoder Initialization

The first training stage uses $\mathcal{D}$ to obtain a warm start for the encoder-decoder configuration $f_e(\cdot), f_d(\cdot)$. This is achieved by training both models as a sequential DNN without including quantization, i.e., mapping an input $\mathbf{x}$ into $f_d(f_e(\mathbf{x}))$. In particular, using the task-dependent loss $\mathcal{L}$, the empirical risk that guides the initial setting of $f_e(\cdot), f_d(\cdot)$ is given by

$$\mathcal{L}_{\mathcal{D}}^{init}(f_e, f_d) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\mathcal{L}(f_d(f_e(\mathbf{x}_i)), y_i). \tag{5}$$

Stage 2: Codebook Initialization

Next, we initialize the vector codebook $\mathcal{Q}$ with S codewords of size $d \times 1$. To that aim, we first pass every $\mathbf{x}_i \in \mathcal{D}$ in the trained encoder, and divide its output $\mathbf{x}_i^e$ into M sub-vectors of size $d \times 1$, denoted $\{\mathbf{x}_{i,m}^e\}_{m=1}^M$. These sub-vectors are aggregated into a new unlabeled dataset $\mathcal{D}_Q = \{\{\mathbf{x}_{i,m}^e\}_{m=1}^M\}_{i=1}^N$. The obtained $\mathcal{D}_Q$ is used to initialize the codebook with S codewords $\{\mathbf{e}_k^{LBG}\}_{k=1}^S$ using the LBG algorithm[56]. The LBG algorithm iteratively constructs a codebook for a single rate by creating non-task-based vector quantizers, with the goal of minimizing distortion (measured via the ℓ_2 norm) in its codeword representation over $\mathcal{D}_Q$. Specifically, it seeks to minimize the following loss function:

$$\mathcal{L}_{\mathcal{D}_Q}^{LBG}(\{\mathbf{e}_k\}) = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}^e \in \mathcal{D}_Q}\min_{k=1,\ldots,S}\|\mathbf{x}^e - \mathbf{e}_k\|_2. \tag{6}$$

This principled codebook initialization facilitates tackling a core challenge in training VQ-VAEs, i.e., the frequent learning of under-used codewords[53], without having the alter the VQ-VAE loss such that it can be utilized for boosting support of multiple rates in the subsequent stage.

Stage 3: Task-Based Joint Adaptation

The codebook vectors are then jointly updated as learnable parameters, along with the encoder and decoder. In this stage, we sequentially refine the model for each quantization level $l = 1, 2, \ldots, \log_2 S$. For each level l, the codebook $\mathcal{Q}_l$ is constructed by expanding the previous codebook $\mathcal{Q}_{l-1}$ with additional code vectors, initially drawn from the LBG initialized $\{\mathbf{e}_k^{LBG}\}_{k=1}^S$.

Specifically, when training at step $l \in \{1, \ldots, \log_2 S\}$, one already has a codebook $\mathcal{Q}_{l-1}$ along with the encoder-decoder trained so far. Thus, the extended codebook $\mathcal{Q}_l$ is initialized by setting its first 2^{l-1} codewords, denoted $\{\mathbf{e}_1^{(l)}, \ldots, \mathbf{e}_{2^{l-1}}^{(l)}\}$, to be the same ordering of codewords in $\mathcal{Q}_{l-1}$, denoted

$\{\mathbf{e}_1^{(l-1)},\ldots,\mathbf{e}_{2^{l-1}}^{(l-1)}\}$, while setting the remaining 2^{l-1} codewords to be the corresponding codewords from $\{\mathbf{e}_k^{LBG}\}_{k=1}^{S}$. Then, to learn $\mathcal{Q}_l$ while having the decoder be suitable for all sub-codebooks in $\mathcal{Q}_l$, we further train $f_e(\cdot)$, $\mathcal{Q}_l$, and $f_d(\cdot)$ using a loss measure which accounts the inference accuracy achieved with all codebooks of quantization levels up to l, while encouraging the first 2^{l-1} codewords $\mathcal{Q}_l$ not to deviate much from those already learned. This loss at step l is

$$\mathcal{L}_{tot}^{(l)}(y_t;\mathbf{x}_t) = \sum_{j=1}^{l} \mathcal{L}(y_t;\hat{y}_t^{(j)}) + \|sg(\mathbf{x}_t^e) - \mathbf{z}_t^{(j)}\|_2^2$$

$$+\beta_j\|\mathbf{x}_t^e - sg(\mathbf{z}_t^{(j)})\|_2^2 + \eta l \sum_{k=1}^{2^{l-1}} \|\mathbf{e}_k^{(l)} - \mathbf{e}_k^{(l-1)}\|_2^2. \qquad (7)$$

In (7), $\mathbf{z}_t^{(j)}$ is the vector obtained by quantizing $\mathbf{x}_t^e = f_e(\mathbf{x}_t)$ using the first 2^j codewords in $\mathcal{Q}_l$, while $\hat{y}_t^{(j)} = f_d(\mathbf{z}_t^{(j)})$.

Equation (7) encapsulates the cumulative impact of quantization levels up to l. The first three terms are based on the VQ-VAE training loss as in (2), aggregated over all resolutions. The last term promotes rate adaptability, with the hyperparameter $\eta \geq 0$ governing its impact. The overall loss using dataset $\mathcal{D}$ is

$$\mathcal{L}_{\mathcal{D}}^{tot}(f_e,\mathcal{Q},f_d) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\sum_{l=1}^{\log_2 S} \mathcal{L}_{tot}^{(l)}(y_i;\mathbf{x}_i). \qquad (8)$$

A concise depiction of the training algorithm, where mini-batch stochastic gradient descent is employed for training in Stages 1 and 3, is presented in Algorithm 1, and the overall procedure is illustrated as Fig. 3.

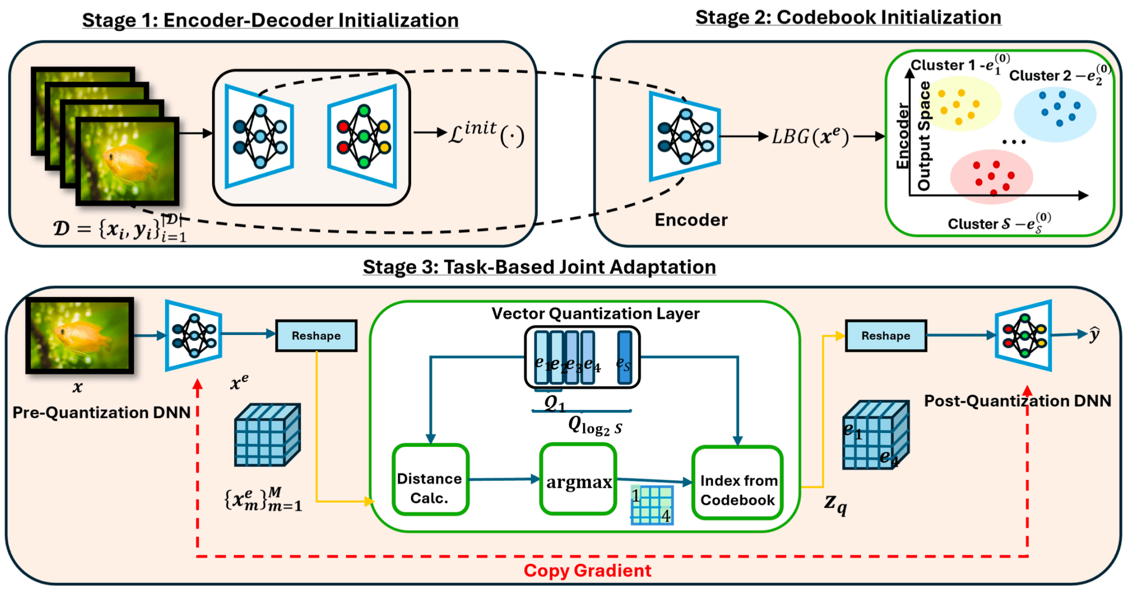


Figure 3. ARTOVeQ training illustration

```
Algorithm 1: ARTOVeQ Training
   Input: Dataset $\mathcal{D}$; Bits limit $S$;
          Loss hyperparameters $\{\eta_l\}$ and $\{\beta_j\}$
   Stage 1: Encoder-Decoder Initialization;
1  for epoch $= 0, 1, \ldots$ do
2      Randomly divide $\mathcal{D}$ into $P$ batches $\{\mathcal{D}_p\}_{p=1}^P$;
3      for $p = 1, \ldots, P$ do
4          Compute loss on $\mathcal{D}_p$ using (5);
5          Update $f_e$ and $f_d$ using loss gradient;
   Stage 2: Codebook Initialization;
6  Obtain $\mathcal{D}_{\rm Q}$ by applying $f_e(\cdot)$ to each $\boldsymbol{x} \in \mathcal{D}$;
7  Set $\{\boldsymbol{e}_k^{\rm LBG}\}_{k=1}^S$ from $\mathcal{D}_{\rm Q}$ using LBG;
   Stage 3: Task-based Joint Adaptation;
8  Set $\mathcal{Q}_0 = \emptyset$;
9  for quantization level $l \in \{1, 2, \ldots, \log_2 S\}$ do
10     Init $\mathcal{Q}_l$ by adding $\{\boldsymbol{e}_k^{\rm LBG}\}_{k=2^{l-1}+1}^{2^l}$ to $\mathcal{Q}_{l-1}$;
11     for epoch $= 0, 1, \ldots$ do
12         Randomly divide $\mathcal{D}$ into $P$ batches $\{\mathcal{D}_p\}_{p=1}^P$;
13         for $p = 1, \ldots, P$ do
14             Compute loss on $\mathcal{D}_p$ using (8);
15             Update $f_e$, $\mathcal{Q}_l$, and $f_d$ using loss gradient;
   Output: Trained $f_e$ and $f_d$; codebook $\mathcal{Q} = \mathcal{Q}_{\log_2 S}$.
```

C. Mixed Resolution ARTOVeQ

ARTOVeQ learns a task-oriented vector quantizer using a single codebook that can be applied across multiple resolutions. Still, once a bit budget $l \in \{1, \ldots, \log_2 S\}$ is fixed, the same l-bit codebook is applied to each features sub-vector, at an overall budget at time t of $B_t = M \cdot l$ bits per input. However, the fact that the same codebook $\mathcal{Q}$ can be decomposed into multiple codebooks of different resolutions can be leveraged to quantize high dimensional inputs with mixed resolutions applied to different features.

1. Architecture

To formulate the mixed resolution ARTOVeQ, we recall that the encoder output $\mathbf{x}_t^e$ is divided into the M sub-vectors $\{\mathbf{x}_{t,m}^e\}_{m=1}^M$. Each features segment is assigned a specific sub-codebook based on a designated bit resolution S_m, with a total bit budget at time t is $B_t = \sum_m \log_2 S_m$ representing the sum of bits allocated across all segments. The resulting bit budget B_t can thus take any value in the range $[M, M+1, \ldots, M \log_2 S]$, indicating that the mixed resolution design provides high bit budget flexibility and granularity, a property not achieved with alternative variable rate learned quantizer architectures whose focus is on the encoder-decoder architecture, e.g., [40][41][42][43][44]. An illustration of the mixed resolution ARTOVeQ can be seen in Fig. 4.

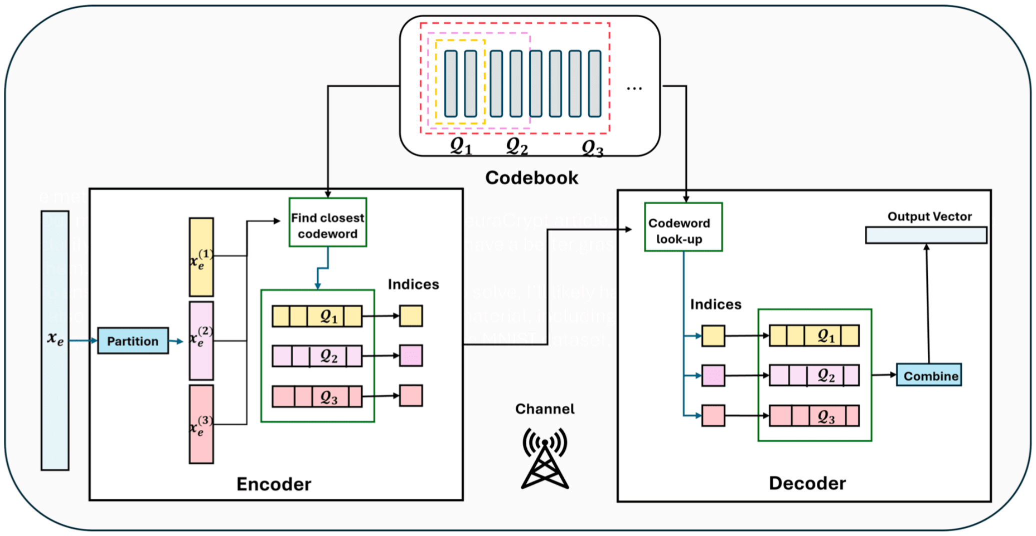


Figure 4. Mixed resolution ARTOVeQ illustration. Different colors represent different quantization resolutions.

2. Training

The training of mixed resolution ARTOVeQ follows the same procedure as in Algorithm 1, with a slight modification applied in **Stage 3**. Here, instead of progressively increasing the resolution of the codebook and having it employed for quantizing all M features, we gradually increase the resolution of the first $d \times 1$ features $\mathbf{x}_{t,1}^{e}$, after which we increase the resolution of quantizing $\mathbf{x}_{t,2}^{e}$, and so on. The rationale in this form of gradual learning draws inspiration from classical image compression methods based on quantizing different components with different resolution, e.g., [57]. In doing so, we aim to consistently have some features quantized with improved resolution, such that the task-based encoder-decoder be encouraged to embed there features that are more informative with respect to the task.

D. Progressive ARTOVeQ

While the training procedure used by ARTOVeQ is based on progressive learning, where the resolution of intermediate features gradually grows during training [48], the resulting quantizer does not immediately support progressive quantization. Specifically, for a chosen bit budget B_t, the codewords do not support progressive decoding, namely, the decoder has to have access to all bits representing the compressed features in order to infer. Nonetheless, while the formulation of the codebook in Subsection III-B only allows variable-rate operation, the fact that what one learns is the multi-resolution codebook implies that it can naturally extend to have a progressive codebooks, whose codewords incrementally build on prior representations, as a form of successive refinement.

1. Architecture

To support progressive quantization, we alter the codebook constraint of (3) to be one which supports successive refinement of initial low-resolution representations of the codewords. Drawing inspiration from nested quantization, which is typically considered in the context of uniform[45] and lattice codebooks[58], we constrain each intermediate codebook $\mathcal{Q}_l$ to represent a one bit refinement of $\mathcal{Q}_{l-1}$. Mathematically, for each $l \in \{1, \dots, \log_2 S\}$ there exist $d \times 1$ vectors $\tilde{\mathbf{e}}_1^{(l)}, \tilde{\mathbf{e}}_2^{(l)}$ such that

$$\mathcal{Q}_l = \mathcal{Q}_{l-1} + \{\tilde{\mathbf{e}}_1^{(l)}, \tilde{\mathbf{e}}_2^{(l)}\}, \tag{9}$$

with $+$ being the Minkowski set sum, thus $|\mathcal{Q}_l| = 2 \cdot |\mathcal{Q}_{l-1}|$.

The constrained codebook form in (9) enables progressive recovery via successive refinement. Specifically for an encoder output $\mathbf{x}_t^e$ and its decomposition into $\{\mathbf{x}_{t,m}^e\}$, the decoder only needs one bit per each sub-vector to recover their representation in $\mathcal{Q}_1$ and use it for inference. With the next M bits, the decoder obtains the improved representation in $\mathcal{Q}_2$, and uses it to improve its inference output, and so on.

2. *Training*

Progressive ARTOVeQ is based on the learned task-based vector quantizer detailed in Subsection III-B, while introducing an alternative constraint on the learned multivariate codebook in the form of (9). Accordingly, the training procedure of progressive ARTOVeQ is based on the learning procedure stated in Algorithm 1, with three main differences introduced to support the constrained progressive form (9):

- Since the LBG algorithm is based on clustering the inputs without accommodating the desired constrained form, the initialization of the codebook in **Stage 2** is omitted.
- For each quantization level l, the aspects of the codebook that are learned are the two difference vectors $\tilde{\mathbf{e}}_1^{(l)}, \tilde{\mathbf{e}}_2^{(l)}$. These are randomly initialized for each resolution.
- As the codebooks no longer satisfy $\mathcal{Q}_{l-1} \subset \mathcal{Q}_l$, but instead hold (9), the regularizer encouraging the former in $\mathcal{L}_{tot}^{(l)}$ is canceled, i.e., we set $\eta_l = 0$ for each l in (7).

The resulting training algorithm is summarized as Algorithm 2.

Algorithm 2: Progressive ARTOVeQ Training

```
   Input: Dataset $\mathcal{D}$; Bits limit $S$;
          Loss hyperparameters $\{\beta_j\}$
1  Initialize $f_e$ and $f_d$ via Stage 1;
2  Set $\mathcal{Q}_0 = \emptyset$ and $\eta_l \equiv 0$;
3  for quantization level $l \in \{1, 2, \ldots, \log_2 S\}$ do
4      Randomize $\tilde{\mathbf{e}}_1^{(l)}, \tilde{\mathbf{e}}_2^{(l)}$;
5      for epoch $= 0, 1, \ldots$ do
6          Randomly divide $\mathcal{D}$ into $P$ batches $\{\mathcal{D}_p\}_{p=1}^{P}$;
7          for $p = 1, \ldots, P$ do
8              Set $\mathcal{Q}_l$ via (9);
9              Compute loss on $\mathcal{D}_p$ using (8);
10             Update $f_e$, $\tilde{\mathbf{e}}_1^{(l)}, \tilde{\mathbf{e}}_2^{(l)}$, $f_d$ using loss gradient;
   Output: Trained $f_e$, $f_d$; differences $\{\tilde{\mathbf{e}}_1^{(l)}, \tilde{\mathbf{e}}_2^{(l)}\}_{l=1}^{\log_2 S}$.
```

E. Discussion

The proposed ARTOVeQ is designed to facilitate learning a single task-based vector quantization codebook that supports finer granularity and progressive decoding through the use of multiple resolution. Accordingly, it is particularly suitable for remote inference over time-varying communication links, e.g., with mobile users[1]. This flexibility allows the compression rate to be adjusted according to dynamic channel conditions, ensuring accurate inference, while supporting a broad range of multiple resolutions (via mixed-resolution among different feature sub-vectors), as well as allowing the decoder to provide rapid inference and gradually improve it via successive refinement. Adaptivity is achieved through nested codebooks, and progressive learning techniques, allowing the system to refine its performance over successive iterations or stages of operation. As ARTOVeQ focuses on learning the quantization codebook and does not restrict the task-based mappings f_e and f_d, it can be integrated in various DNN architectures.

While our setup primarily focuses on a pair of sensing and inferring users, this methodology is extensible to collaborative inference among multiple edge devices. Our approach assumes that the instantaneous channel capacity (C_t) is known, enabling the sensing user to determine the appropriate quantization level. However, a potential extension of our scheme could allow it to function without prior knowledge of channel capacity, dynamically tuning the quantization rate during the remote inference process. Another potential aspect for future exploration, which stems from the ability to learn a variable rate and progressive vector quantization codebook integrated into a remote inference system, is its ability to enhance data privacy and security. Recent studies have shown that well-designed compression strategies can enhance privacy, providing an additional benefit to our rate-adaptive scheme[59][60]. Furthermore, recent advancements in randomized neural networks demonstrate their potential for ensuring privacy[61]. While ARTOVeQ has the potential of supporting such extensions, they would necessitate reformulation of the learning procedure, and are thus left for future investigation.

IV. Numerical Experiments

In this section, we present the results of our numerical experiments[1]. We first detail our experimental setup in Subsection IV-A, after which we detail our four main studies, each focusing on a distinct aspect of our approach for image classification: variable-rate task-based compression

(Subsection IV-B), mixed resolution compression (Subsection IV-C), progressive compression (Subsection IV-D), and remote inference over dynamic channels (Subsection IV-E).

A. Experiential Setup

To evaluate our quantization scheme, we use two datasets: CIFAR-100 and Imagewoof. CIFAR-100 consists of 60,000 diverse images with dimensions $3 \times 32 \times 32$, spanning 100 classes and thus encompassing a wide variety of source distributions. Imagewoof contains 10,000 images at a higher resolution of $3 \times 64 \times 64$, but with a slightly narrower set of 10 classes, each representing different dog breeds. This combination allows us to assess our method's robustness across a large number of source distributions in CIFAR-100, and under a more specific, yet high-resolution, distribution in Imagewoof.

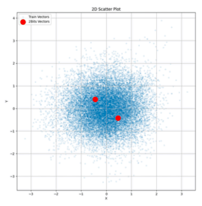
For our evaluation, we employed the MobileNetV2[49] architecture to accommodate edge device constraints, partitioning it into an encoder $f_e(\cdot)$, comprising the first four residual blocks, and a decoder $f_d(\cdot)$ with the remaining blocks. The encoder-decoder and codebooks were jointly trained using the Adam optimizer, with learning rate $1 \cdot 10^{-4}$ and batch sizes 32 and 16 for CIFAR-100 and Imagewoof, respectively.

We evaluate the performance in terms of test accuracy achieved with the following quantization methods: $1)$ ARTOVeQ (as detailed in Subsection III-B); $2)$ a single-rate VQ-VAE, in which a different codebook is trained for each bit budget, constituting an upper-bound on the performance achievable with a single codebook shared among all resolutions; $3)$ mixed resolution ARTOVeQ (detailed in Subsection III-C); $4)$ progressive ARTOVeQ (detailed in Subsection III-D); $5)$ residual VQ-VAE (RVQ-VAE)[40]; and 6) Single-Rate LBG, in which LBG[56] is applied anew to the learned encoder output for quantization for each codebook size. In each experiment, the encoder's output was divided into M segments, and the quantizer is applied to each sub-vector.

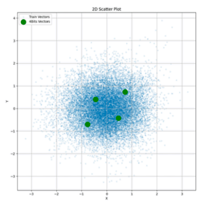
B. Variable-Rate Task-Based Compression

We first assess the performance of variable-rate ARTOVeQ in an environment where bit-rate availability may vary over time, demonstrating its capability to enable remote inference across a broad range of communication conditions. We show that, despite utilizing a single codebook across multiple resolutions, ARTOVeQ remains competitive with single-rate VQ-VAE and outperforms other benchmark models.

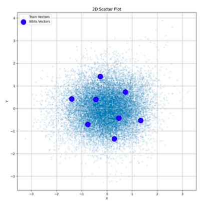
Learned Codebooks: A defining characteristic of ARTOVeQ is its nested codebook structure. Fig. 5 illustrates the progression of codebooks $\mathcal{Q}_1, \ldots, \mathcal{Q}_8$ for $d = 2$ on the CIFAR-100 dataset. As the bit resolution increases, the codebook vectors progressively capture the latent state distribution with greater precision, leading to improved performance at higher resolutions.



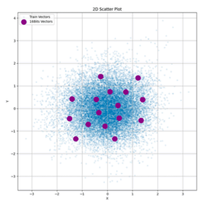
(a) $\mathcal{Q}_1$



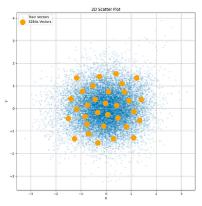
(b) $\mathcal{Q}_2$



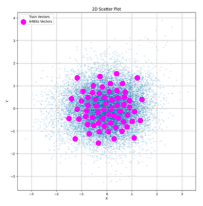
(c) $\mathcal{Q}_3$



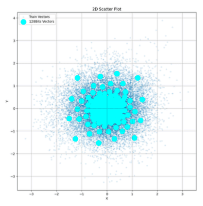
(d) $\mathcal{Q}_4$



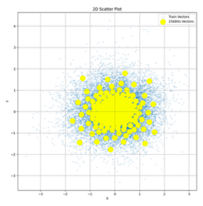
(e) $\mathcal{Q}_5$



(f) $\mathcal{Q}_6$



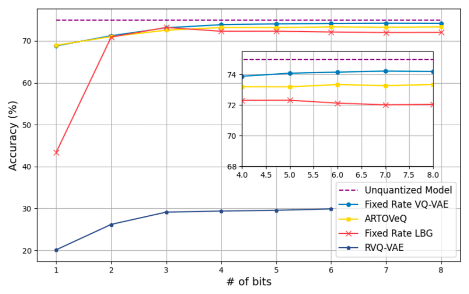
(g) $\mathcal{Q}_7$



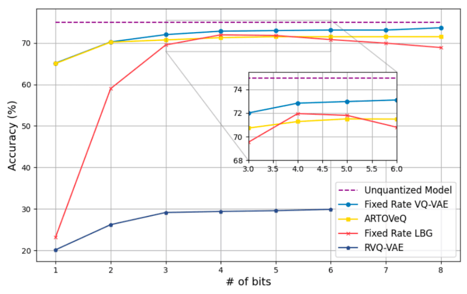
(h) $\mathcal{Q}_8$

Figure 5. Learned codebook vectors. Embedding dimensions $d = 2$

Performance Evaluation: Having showcased the codebook structures learned by ARTOVeQ, we proceed to evaluating its performance when integrated into a remote inference system. The results achieved for CIFAR-100 are reported in Fig. 6. There, we observe the trade-offs between compression via quantization and performance in ARTOVeQ compared to other variable-rate and single-rate baselines. As expected, each approach exhibits an increasing trend in performance before tapering off and saturating at higher resolutions, typically around 4–5 bits for per sub-vector.



(a) Accuracy vs. bits, $d = 2$: ARTOVeq outperforms RVQ-VAE and LBG, closely matching fixed-rate VQ-VAE

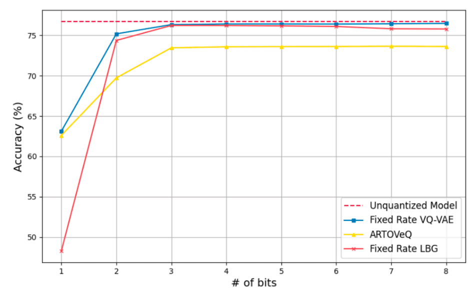


(b) Accuracy vs. bits, $d = 4$: ARTOVeq outperforms RVQ-VAE and LBG, while slightly underperforming fixed-rate VQ-VAE

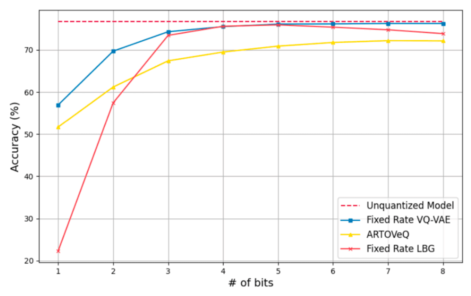
Figure 6. CIFAR-100 Accuracy as a function of bits per sub-vector for varying codebook vector sizes d.

ARTOVeQ consistently performs just slightly below the single-rate VQ-VAE, with a performance drop of approximately 0.8% for $d = 2$. Some performance degradation is observed when transitioning from $d = 2$ to $d = 4$ as the number of bits per codeword is remains constant, while the dimensionality of the codewords increases, i.e., the quantizaiton rate is reduced. Despite this, ARTOVeQ still outperforms both single-rate LBG and RVQ-VAE. This accuracy degradation relative to single-rate VQ-VAE can be attributed to the constraints imposed by ARTOVeQ's nested codebook structure. While ARTOVeQ supports multiple bit resolutions within a single codebook, it is limited in its ability to independently optimize for each resolution, a benefit that single-rate VQ-VAE possesses. The LBG algorithm consistently ranks second to last, likely due to the diverse distribution of CIFAR-100.

Similar results are observed in the Imagewoof data, as reported in Fig. 7. As seen in Fig. 7, as opposed to CIFAR-100, here the single codebook of ARTOVeQ results in its reaching a small gap in performance from fixed-rate methods, that use a different codebook for each resolution. This can be attributed to the higher redundancy at higher resolutions of the source data, and its smaller number of labels, which allows fixed rate methods to obtain suitable codebooks for sufficient number of bits at the output of the encoder. ARTOVeQ maintains a consistent 2% performance gap, which reflects the challenge of achieving an optimal latent representation given the constraints imposed by the nested structure's. RVQ-VAE, which achieves the lowest accuracy on CIFAR-100, was shown to be instable and fail to faithfully learn, and was thus not included in the figure. ARTOVeQ consistently performs well across both datasets, demonstrating its robustness in diverse scenarios.



(a) Accuracy vs. bits $d = 2$: ARTOVeQ underperforms compared to fixed-rate methods



(b) Accuracy vs. bits $d = 4$: ARTOVeQ underperforms compared to fixed-rate methods

Figure 7. Imagewoof: Comparison of accuracy vs. total number of bits for different values of d: the unquantized MobileNetV2, the single-rate standalone VQ-VAE, ARTOVeQ, and LBG

In terms of complexity, ARTOVeQ operates in a one-shot manner, in contrast to the iterative encoding-decoding process of RVQ-VAE, which relies on residuals. As a result, ARTOVeQ offers a significant advantage in computational efficiency. In RVQ-VAE, the multiple forward pass iterations required to achieve higher resolutions substantially increase computational demand.

C. Mixed-Resolution Compression

We proceed to evaluating the ability of ARTOVeQ to leverage its multi-resolution codebook to quantize different sub-vectors with different resolutions. For this task, we partitioned the encoder output, $\mathbf{x}_t^e$, into four blocks with a manual bit allocation strategy. The first segments were assigned the highest bit representations, following a policy where the largest bit share is allocated to the first segment, with subsequent segments receiving progressively lower bit resolutions that collectively sum to a predefined bit budget B_t. The aim of this study is the examine the performance of using mixed resolution codebooks compared to identical resolution ones with the same overall bit budget (which we contrasted with various benchmarks in Subsection IV-B).

The CIFAR-100 and Imagewoof results corresponding to $d = 2$ and $d = 4$ are shown in Figs. 8-9, respectively. There, we compare accuracy for different values of total number of bits assigned across four quantizers (that are applied to each four sub-vectors). In the identical resolution case, all quantizers have the same codebook, while in the mixed resolution case, the first quantizer uses more codewords compared to the remaining ones. Our findings reveal that, for both datasets, mixed-resolution configurations consistently outperform their identical resolution counterparts, though the improvement varies with the number of bits. This demonstrates that the finer granularity enabled by mixed-resolution compression, combined with task-based learning, allows for a more refined latent space representation, resulting in improved performance. As seen in Fig. 8, this effect is particularly evident in lower bit budgets, between 4-10 bits, where the richer bit spectrum enables more effective learning and performance gains.

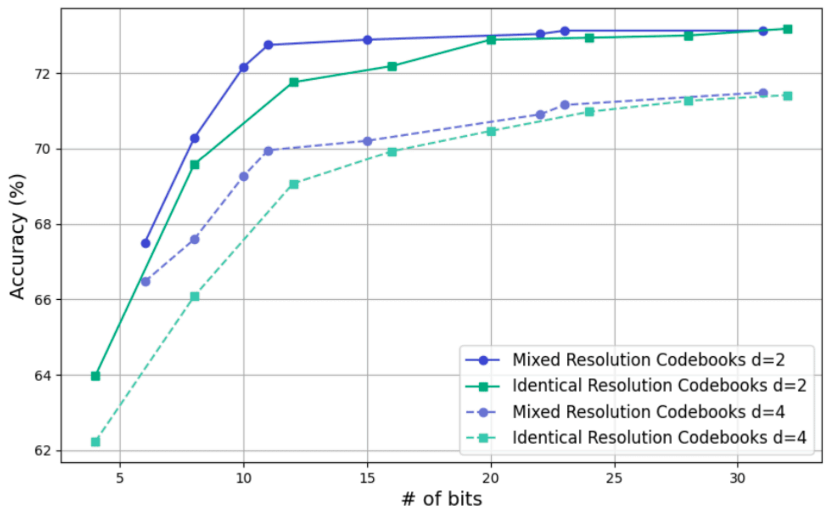


Figure 8. CIFAR-100: Accuracy versus total number of bits for four configurations—mixed resolution and identical resolution. Solid lines correspond to $d = 2$, while dashed lines represent $d = 4$. Mixed resolution demonstrates a broader range of allocated bits, leading to improved performance.

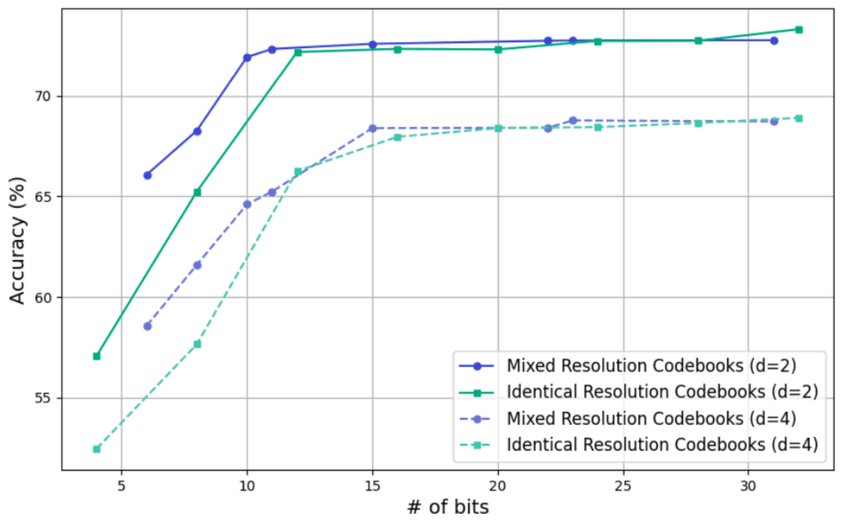


Figure 9. Imagewoof: Accuracy versus total number of bits for mixed resolution and identical resolution. Solid lines correspond to $d = 2$, while dashed lines represent $d = 4$. The performance of mixed resolutions closely aligns with that of identical resolutions.

Quantitatively, we observe a performance gap of approximately $0.4\% - 0.7\%$ for $d = 2$ and $0.2\% - 1.5\%$ for $d = 4$ on the CIFAR-100 dataset within the 8-20 bit range. Similarly, for the Imagewoof dataset, the gap ranges from $0.2\% - 3\%$ for $d = 2$ and $1.6\% - 4\%$ for $d = 4$. These findings suggest that improved performance can be achieved with a smaller bit budget. At the higher end of the bit spectrum, identical-resolution configurations tend to closely match the performance of mixed-resolution ones for both datasets. However, due to the redundancy and narrower distribution of Imagewoof, this alignment is reached at a lower bit budget. In all cases, mixed-resolution configurations offer the advantage of flexible memory usage.

D. Progressive Compression

We proceed by evaluating the progressive quantization codebook version of ARTOVeQ, with its successive refinement approach. Specifically, we aim to assess how effectively the progressive constraint and its corresponding learning technique balance compression efficiency and task accuracy, and to compare the performance of successive bit increments against the variable-rate ARTOVeQ evaluated in Subsection IV-B.

Our findings, shown in Fig. 10 for CIFAR-100 and in Fig. 11 for Imagewoof, demonstrate that, as expected, variable-rate ARTOVeQ consistently outperforms the more constrained progressive codebook across all bit resolutions and quantization embeddings ($d = 2$ and $d = 4$). Nonetheless, progressive codebooks manage to approach the performance of variable-rate ARTOVeQ owing to its dedicated learning technique, within some performance gap that varies between the considered tasks. The discrepancy is attributed to the the strict progressive constraint, which, while allowing for incremental decoding with minimal latency, comes at the cost of some performance degradation compared to variable rate ARTOVeQ. This is particularly evident for Imagewoof with $d = 4$.

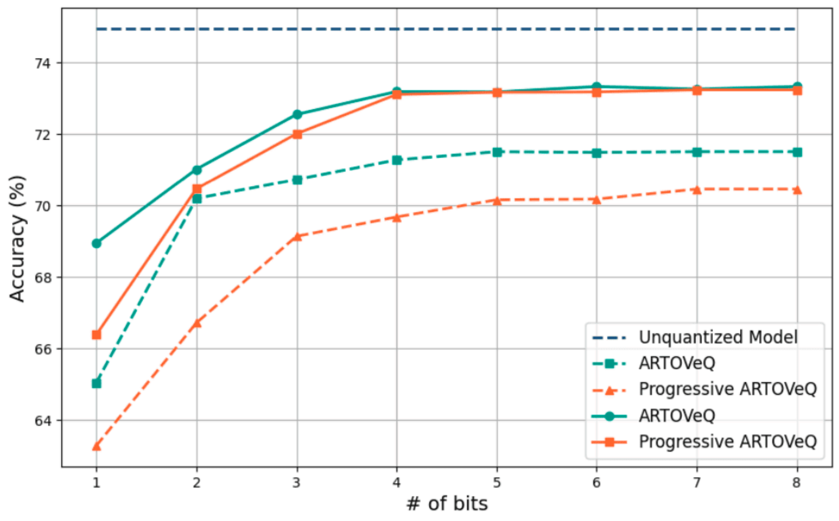


Figure 10. CIFAR-100: Comparison of accuracy as a function of the total number of bits for successive refinement and variable-rate ARTOVeQ. Solid lines indicate $d = 2$, and dashed lines indicate $d = 4$. Variable-rate ARTOVeQ consistently outperforms successive refinement, with the performance gap most prominent at lower bit rates (2–3 bits).

Despite the performance gap, the results show that both variable-rate ARTOVeQ and progressive ARTOVeQ begin to saturate around 6 bits, with only marginal improvement beyond this point. From a complexity standpoint, both techniques operate in a one-shot fashion; however, progressive quantization has the advantage of minimal latency, as inference can begin immediately after the first bit is received, whereas variable-rate ARTOVeQ requires the entire bit sequence.

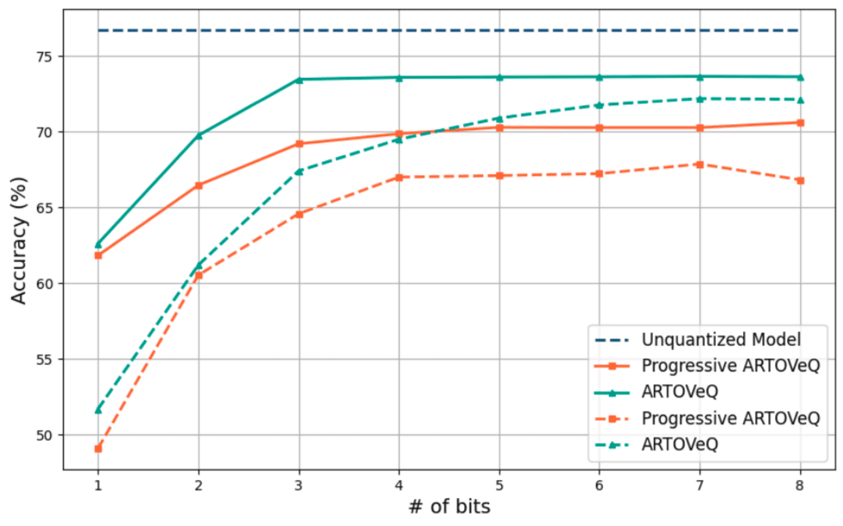


Figure 11. Imagewoof: Accuracy comparison as a function of the total number of bits for successive refinement and variable-rate ARTOVeQ. Solid lines represent $d = 2$, while dashed lines represent $d = 4$. ARTOVeQ demonstrates superior performance, with a consistent gap of approximately 2% across all bit rates.

E. Remote Inference over Dynamic Channels

The experimental studies so far have evaluated the different versions of ARTOVeQ, all trained to support multiple quantization resolutions, in a given bit rate. As the motivation for ARTOVeQ is to facilitate remote inference over dynamic channel with a single codebook, we next evaluate its aggregated performance when repeatedly applied for remote inference with changing channel conditions.

Experimental Setup: To evaluate the performance of models in a dynamic channel environment, we consider a communication system where the channel capacities C_t fluctuate over time with coherence duration τ. At a given time t, the channel can support a maximal bit-rate $B_t = \tau \cdot C_t$, such that the per-codebook bit-budget B_t/M takes values in $\{1, 2, \ldots, 8\}$. To reflect variability, we simulate three distinct channel scenarios: $S1$ a uniform distribution of bit-rates; $S2$ scenarios where lower bit-rates are more likely, and $S3$ scenarios where higher bit-rates are more likely. Theses are obtained by setting

$$\Pr\left(B_t = M \cdot b\right) = \frac{e^{kb}}{\sum_{b'=1}^{8} e^{kb'}}, \quad b \in \{1, 2, \ldots 8\},$$

where we set $k = 0, -0.25, 0.25$ to obtain scenarios $S1$, $S2$ and $S3$, respectively.

Model Evaluation: We compare the average accuracy of ARTOVeQ and Progressive ARTOVeQ, which both maintain a single DNN and a single codebook, to two main benchmarks: The first is remote inference system that maintains eight different fixed-rate encoder-VQ-VAE-decoder chains, constituting the most flexible yet extremely costly alternative. We also compare to using a single fixed-rate encoder-VQ-VAE-decoder designed with codebook sizes $\log_2 S \in \{1, 4, 8\}$. As the latter operates at a fixed rate, it fails to convey the samples within the coherence time when its rate surpasses that supported by the channel.

The results obtained with the CIFAR-100 and the Imagewoof dataset sare reported in Table 1. The experimental results across CIFAR-100 and Imagewoof datasets reveal consistent performance behaviors for both $d = 2$ and $d = 4$. As expected, Multiple Fixed-Rate VQ-VAE consistently achieves the highest inference accuracy, while being only within a minor gap from ARTOVeQ across all scenarios. The progressive ARTOVeQ, designed with the mechanism of incremental codebook vector's improvement, shows a slight performance drop of approximately $0.5\% - 2\%$ on CIFAR-100 and $3\% - 4\%$ on Imagewoof. Conversely, the Single-Rate VQ-VAE struggles in scenarios where the channel's supported bit-rate is insufficient to meet the model's pre-defined bit-rate requirement. This limitation highlights the lack of versatility, as it can not perform inference under constrained channel conditions. These results indicate on the ability of ARTOVeQ and its variants to support flexible remote inference over dynamic channels.

d	Model		CIFAR-100			ImageWoof		
			$S1$	$S2$	$S3$	$S1$	$S2$	$S3$
2	**Multiple Fixed-Rate VQ-VAE**		72.95	72.00	73.65	74.59	72.75	75.76
	Single-Rate VQ-VAE	1-bit	68.82	68.82	68.82	63.11	63.11	63.11
		4-bit	46.17	28.79	60.95	47.75	29.78	63.05
		8-bit	9.27	3.30	18.98	9.56	3.40	19.57
	ARTOVeQ		72.35	71.59	72.90	71.73	70.00	72.90
	Progressive ARTOVeQ		71.85	70.72	72.66	68.59	67.14	69.64
4	**Multiple Fixed-Rate VQ-VAE**		71.64	70.31	72.61	72.65	69.59	74.77
	Single-Rate VQ-VAE	1-bit	65.15	65.15	65.15	56.94	56.94	56.94
		4-bit	45.53	28.40	60.11	47.19	29.43	62.31
		8-bit	9.21	3.27	18.84	9.53	3.39	19.51
	ARTOVeQ		70.41	69.45	71.07	67.08	63.52	69.80
	Progressive ARTOVeQ		68.76	67.53	69.68	63.77	60.82	65.82

Table 1. Performance comparison of various models for time-varying channels.

V. Conclusions

We proposed ARTOVeQ, a DNN-based remote inference mechanism with a single multi-resolution codebook that supports multi-rate vector quantization with both identical, mixed-level, and progressive resolutions. We devised a method to learn nested codebooks via a dedicated gradual learning scheme, enabling a single model to operate at various resolutions. Our numerical analyses highlight the performance trade-offs between our rate-adaptive mechanisms and model-based as well as data-driven alternatives for task-based vector quantization, showing the ability of ARTOVeQ to learn a remote inference system with single codebook whose performance approaches systems where each rate is trained individually.

Notes

Parts of this work were presented at the 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) as the paper[62].

Footnotes

1 The source code and hyperparameters used in this experimental study are available at https://github.com/eyalfish/ARTOVeQ.

References

1. a, b, c *Shlezinger N, Bajic IV (2022). "Collaborative Inference for AI-Empowered IoT Devices". IEEE Internet of Things Magazine.* **5** *(4): 92–98.*
2. ^ *Zou H, Zhang C, Lasaulce S, Saludjian L, Poor HV (2022). "Goal-oriented quantization: Analysis, design, and application to resource allocation". IEEE J Sel Areas Commun.* **41** *(1): 42--54.*
3. ^ *Cover TM, Thomas JA. Elements of information theory. John Wiley & Sons; 1999.*
4. a, b *Chen J, Fang Y, Khisti A, Ozgur A, Shlezinger N, Tian C (2025, early access). "Information Compression in the AI Era: Recent Advances and Future Challenges." IEEE J. Sel. Areas Commun..*
5. ^ *Shisher MKC, Sun Y, Hou I-H (2024). "Timely communications for remote inference". IEEE/ACM Transactions on Networking.* **32** *(5): 3824–3839.*
6. ^ *Lu Z, Li R, Lu K, Chen X, Hossain E, Zhao Z, Zhang H (2024). "Semantics-Empowered Communications: A Tutorial-cum-Survey." IEEE Commun. Surveys Tuts..* **26** *(1): 41--79.*
7. a, b, c *Shlezinger N, Eldar YC, Rodrigues MR (2019). "Hardware-Limited Task-Based Quantization". IEEE Trans Signal Process.* **67** *(20): 5223–5238.*
8. ^ *Neuhaus P, Shlezinger N, Dörpinghaus M, Eldar YC, Fettweis G (2021). "Task-based analog-to-digital converters". IEEE Trans. Signal Process..* **69***: 5403–5418.*
9. ^ *Agheli P, Pappas N, Kountouris M (2022). "Semantics-aware source coding in status update systems." In: IEEE International Conference on Communications Workshops (ICC Workshops). pp. 169–174.*
10. ^ *Kountouris M, Pappas N (2021). "Semantics-empowered communication for networked intelligent systems". IEEE Commun. Mag..* **59** *(6): 96–102.*

11. ^Zhang C, Zou H, Lasaulce S, Saad W, Kountouris M, Bennis M (2022). "Goal-oriented communications for the IoT and application to data compression." IEEE Internet of Things Magazine. **5** (4): 58–63.

12. ^Di Lorenzo P, Merluzzi M, Binucci F, Battiloro C, Banelli P, Strinati EC, Barbarossa S (2023). "Goal-ori ented communications for the IoT: System design and adaptive resource optimization." IEEE Internet of Things Magazine. **6** (4): 26–32.

13. ^Gündüz D, Qin Z, Aguerri IE, Dhillon HS, Yang Z, Yener A, Wong KK, Chae C-B (2022). "Beyond transm itting bits: Context, semantics, and task-oriented communications." IEEE J Sel Areas Commun. **41** (1): 5 –41.

14. ^Salamtian S, Shlezinger N, Eldar YC, M\u00e9dard M (2019). "Task-Based Quantization for Recoverin g Quadratic Functions Using Principal Inertia Components." In: Proc. IEEE Int. Symp. Inf. Theory.

15. ^Bernardo NI, Zhu J, Eldar YC, Evans J (2023). "Design and analysis of hardware-limited non-uniform task-based quantizers." IEEE Trans. Signal Process.. **71**: 1551--1562.

16. ^Xie H, Qin Z, Li GY, Juang BH (2021). "Deep learning enabled semantic communication systems". IEEE Trans. Signal Process.. **69**: 2663–2675.

17. ^Shao J, Mao Y, Zhang J (2021). "Learning task-oriented communication for edge inference: An inform ation bottleneck approach." IEEE J. Sel. Areas Commun.. **40** (1): 197–211.

18. a, bJankowski M, Gündüz D, Mikolajczyk K (2020). "Wireless image retrieval at the edge". IEEE J Sel Are as Commun. **39** (1): 89–100.

19. ^Torfason R, Mentzer F, Agustsson E, Tschannen M, Timofte R, Van Gool L (2018). "Towards image und erstanding from deep compression without decoding." arXiv preprint arXiv:1803.06131.

20. a, bBallé J, Chou PA, Minnen D, Singh S, Johnston N, Agustsson E, Hwang SJ, Toderici G (2020). "Nonline ar transform coding". IEEE J Sel Topics Signal Process. **15** (2): 339–353.

21. ^Shlezinger N, Eldar YC (2021). "Deep task-based quantization". Entropy. **23** (1): 104.

22. a, bShlezinger N, Amar A, Luijten B, van Sloun RJG, Eldar YC (2022). "Deep task-based analog-to-digit al conversion". IEEE Trans Signal Process. **70**: 6021--6034.

23. ^Vol T, Danial L, Shlezinger N. "Power-Aware Task-Based Learning of Neuromorphic ADCs." In: IEEE I nternational Conference on Acoustics, Speech and Signal Processing (ICASSP); 2024. p. 5850-5854.

24. a, b, c, dVan Den Oord A, Vinyals O (2017). "Neural discrete representation learning." Advances in Neural Information Processing Systems. **30**.

25. a, bMalka M, Farhan E, Morgenstern H, Shlezinger N (2024, early access). "Decentralized Low-Latency Collaborative Inference via Ensembles on the Edge". IEEE Trans. Wireless Commun..

26. ^a, b^*Mishra D, Singh SK, Singh RK (2022). "Deep architectures for image compression: a critical review." Signal Processing.* ***191****: 108346.*

27. ^*Habibian A, van Rozendaal T, Tomczak JM, Cohen TS (2019). "Video compression with rate-distortion autoencoders." In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 703 3–7042.*

28. ^*Zeghidour N, Luebs A, Omran A, Skoglund J, Tagliasacchi M. "Soundstream: An end-to-end neural au dio codec." IEEE Trans. Acoust., Speech, Signal Process..* ***30****: 495--507, 2021.*

29. ^*Yang Y, Mandt S, Theis L, et al. An introduction to neural data compression. Foundations and Trends® in Computer Graphics and Vision.* ***15****(2):113-200, 2023.*

30. ^*Li G, Liu L, Wang X, Dong X, Zhao P, Feng X. "Auto-tuning neural network quantization framework for collaborative inference between the cloud and edge." In: International Conference on Artificial Neural N etworks (ICANN). Springer; 2018. p. 402–411.*

31. ^a, b^*Agustsson E, Mentzer F, Tschannen M, Cavigelli L, Timofte R, Benini L, Gool LV. "Soft-to-hard vecto r quantization for end-to-end learning compressible representations." In: Advances in Neural Informat ion Processing Systems. 2017. p. 1141–1151.*

32. ^a, b^*Cai C, Chen L, Zhang X, Gao Z (2019). "Efficient Variable Rate Image Compression With Multi-Scale Decomposition Network". IEEE Trans. Circuits Syst. Video Technol..* ***29*** *(12): 3687-3700.*

33. ^a, b^*Choi Y, El-Khamy M, Lee J (2019). "Variable Rate Deep Image Compression With a Conditional Auto encoder." In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3146-3154.*

34. ^a, b^*Yang F, Herranz L, Van De Weijer J, Guitián JA, López AM, Mozerov MG (2020). "Variable rate deep i mage compression with modulated autoencoder". IEEE Signal Process Lett.* ***27****: 331–335.*

35. ^a, b^*Yang F, Herranz L, Cheng Y, Mozerov MG. "Slimmable compressive autoencoders for practical neural image compression." In: IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 49 98–5007.*

36. ^a, b^*Lee J, Jeong S, Kim M (2022). "Selective compression learning of latent representations for variable-rate image compression". Advances in Neural Information Processing Systems.* ***35****: 13146–13157.*

37. ^a, b^*Gupta R, BV S, Kapoor N, Jaiswal R, Nangi SR, Kulkarni K. "User-guided variable rate learned image compression." In: IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2022. p. 1753-175 8.*

38. ^a, b^*Duan Z, Lu M, Ma J, Huang Y, Ma Z, Zhu F (2024). "QARV: Quantization-Aware ResNet VAE for Loss y Image Compression." IEEE Trans. Pattern Anal. Mach. Intell..* ***46*** *(1): 436-450.*

39. ^a, b, c^ *Seo J, Kang J (2024). "RAQ-VAE: Rate-Adaptive Vector-Quantized Variational Autoencoder". arXiv preprint arXiv:2405.14222. Available from: https://arxiv.org/abs/2405.14222.*

40. ^a, b, c, d^ *Toderici G, O'Malley SM, Hwang SJ, Vincent D, Minnen D, Baluja S, Covell M, Sukthankar R (2015). "Variable rate image compression with recurrent neural networks". arXiv preprint arXiv:1511.06085.*

41. ^a, b, c^ *Toderici G, Vincent D, Johnston N, Jin Hwang S, Minnen D, Shor J, Covell M (2017). "Full resolution image compression with recurrent neural networks." In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5306–5314.*

42. ^a, b, c^ *Johnston N, Vincent D, Minnen D, Covell M, Singh S, Chinen T, Hwang SJ, Shor J, Toderici G (2018). "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks." In: IEEE Conference on Computer Vision and Pattern Recognition. 2018. p. 4385–4393.*

43. ^a, b, c^ *Lee JH, Jeon S, Choi KP, Park Y, Kim CS (2022). "DPICT: Deep progressive image compression using trit-planes." In: IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 16113–16122.*

44. ^a, b, c^ *Hojjat A, Haberer J, Landsiedel O. "Prog{DTD}: Progressive learned image compression with double-tail-drop training." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2023. p. 1130–1139.*

45. ^a, b, c^ *Abdi A, Fekri F (2019). "Nested dithered quantization for communication reduction in distributed training." arXiv preprint arXiv:1904.01197. Available from: https://arxiv.org/abs/1904.01197.*

46. ^ *Equitz WH, Cover TM (1991). "Successive refinement of information". IEEE Trans. Inf. Theory.* **37** *(2): 269–275.*

47. ^ *Cheng S, Xiong Z (2005). "Successive refinement for the Wyner-Ziv problem and layered code design." IEEE Trans Signal Process.* **53** *(8): 3269–3281.*

48. ^a, b^ *Du R, Xie J, Ma Z, Chang D, Song YZ, Guo J (2021). "Progressive learning of category-consistent multi-granularity features for fine-grained visual classification". IEEE Trans. Pattern Anal. Mach. Intell..* **44** *(12): 9521–9535.*

49. ^a, b^ *Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. "Mobilenetv2: Inverted residuals and linear bottlenecks." In: IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 4510–4520.*

50. ^ *Gray RM, Neuhoff DL (1998). "Quantization." IEEE Trans. Inf. Theory.* **44** *(6): 2325–2383.*

51. ^ *Kang Y, Hauswald J, Gao C, Rovinski A, Mudge T, Mars J, Tang L (2017). "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge." ACM SIGARCH Computer Architecture News.* **45** *(1): 615--629.*

52. ^Gautam T, Pryzant R, Yang Z, Zhu C, Sojoudi S (2023). *"Soft Convex Quantization: Revisiting Vector Quantization with Convex Optimization". arXiv preprint arXiv:2310.03004.*

53. a, bFifty C, Junkins RG, Duan D, Iger A, Liu JW, Amid E, Thrun S, R\'e C (2024). *"Restructuring Vector Quantization with the Rotation Trick." arXiv preprint arXiv:2410.06424.*

54. ^Huang M, Mao Z, Chen Z, Zhang Y (2023). *"Towards accurate image coding: Improved autoregressive image generation with dynamic vector quantization." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22596–22605.*

55. ^Dong X, Bao J, Zhang T, Chen D, Zhang W, Yuan L, Chen D, Wen F, Yu N, Guo B (2023). *"PeCo: Perceptual codebook for BERT pre-training of vision transformers." Proceedings of the AAAI Conference on Artificial Intelligence.* 37 *(1): 552–560.*

56. a, b, cLinde Y, Buzo A, Gray R (1980). *"An Algorithm for Vector Quantizer Design". IEEE Trans. Commun.. 28 (1): 84-95. doi:10.1109/TCOM.1980.1094577.*

57. ^Shusterman E, Feder M (1994). *"Image compression via improved quadtree decomposition algorithms". IEEE Trans. Image Process..* 3 *(2): 207--215.*

58. ^Zamir R, Shamai S, Erez U (2002). *"Nested linear/lattice codes for structured multiterminal binning." IEEE Trans. Inf. Theory.* 48 *(6): 1250--1276.*

59. ^Alvar SR, Baji\'c IV (2021). *"Scalable privacy in multi-task image compression." In: IEEE International Conference on Visual Communications and Image Processing (VCIP), 2021.*

60. ^Lang N, Sofer E, Shaked T, Shlezinger N (2023). *"Joint privacy enhancement and quantization in federated learning". IEEE Trans. Signal Process..* 71*: 295–310.*

61. ^Esfahanizadeh H, Wu W, Ghobadi M, Barzilay R, M\u00e9dard M. *"Infoshape: Task-based neural data shaping via mutual information." In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2023.*

62. ^Malka M, Ginzach S, Shlezinger N (2023). *"Learning Multi-Rate Vector Quantization for Remote Deep Inference." In: IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW).*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.