Unified Multivariate Gaussian Mixture for Efficient Neural Image Compression

Xiaosu Zhu[1] Jingkuan Song[1*] Lianli Gao[1] Feng Zheng[2] Heng Tao Shen[1]

[1]Center for Future Media, University of Electronic Science and Technology of China
[2]Southern University of Science and Technology

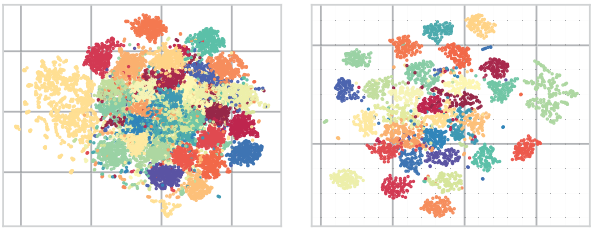
xiaosu.zhu@outlook.com, jingkuan.song@gmail.com, shenhengtao@hotmail.com

Abstract

Modeling latent variables with priors and hyperpriors is an essential problem in variational image compression. Formally, trade-off between rate and distortion is handled well if priors and hyperpriors precisely describe latent variables. Current practices only adopt univariate priors and process each variable individually. However, we find inter-correlations and intra-correlations exist when observing latent variables in a vectorized perspective. These findings reveal visual redundancies to improve rate-distortion performance and parallel processing ability to speed up compression. This encourages us to propose a novel vectorized prior. Specifically, a multivariate Gaussian mixture is proposed with means and covariances to be estimated. Then, a novel probabilistic vector quantization is utilized to effectively approximate means, and remaining covariances are further induced to a unified mixture and solved by cascaded estimation without context models involved. Furthermore, codebooks involved in quantization are extended to multi-codebooks for complexity reduction, which formulates an efficient compression procedure. Extensive experiments on benchmark datasets against state-of-the-art indicate our model has better rate-distortion performance and an impressive $3.18\times$ *compression speed up, giving us the ability to perform real-time, high-quality variational image compression in practice. Our source code is publicly available at* https://github.com/xiaosu-zhu/McQuic.

1. Introduction

As a crucial technique in image processing, lossy image compression has been studied for an extended period [17, 20, 30, 38]. The goal is to achieve high perceptual reconstruction performance, extreme compression rate, and efficient processing pipeline. Classical lossy image compression standards, *e.g.*, JPEG [33, 39], BPG [6], HEIF [35], VVC [8], have been widely applied and adopted as fundamental components in almost all image processing software. However, the explosion of multimedia content in the digital era still raises urgent requests to find an effective and efficient compressor to tackle storage costs.



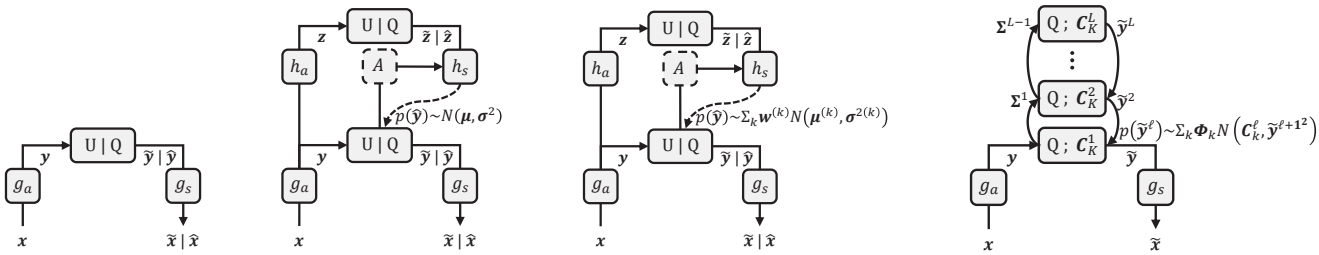
(a) $\boldsymbol{y}^1$ (b) $\boldsymbol{y}^2 = \left(\boldsymbol{y}^1 - \mathfrak{y}^1\right)_{\downarrow}$

Figure 1. UMAP [24] projection of 128-d latent vectors with a toy 2-level 32-codeword model from 24 Kodak images. **Left**: Latent vectors extracted from analysis transform are correlated and can be described by multivariate Gaussian mixture. **Right**: Next level's latents are under similar distribution.

Distinct from the above traditional codecs, learnable neural image compression is proposed by exploiting advantages of deep neural networks. It adopts neural networks as nonlinear transforms to extract binaries from images and restore them, while essential research problem is to handle the trade-off between rate and distortion [7]. Recent studies propose variational image compression and arrange above trade-off as a Lagrange multiplier for joint optimization [3, 4, 9, 25, 26]. They introduce univariate priors and hyperpriors to describe latent variables and make a breakthrough to control rate. We summarize advances in this task as a series of operational diagrams in Figs. 2(a) to 2(c).

To design an effective compressor in variational image compression, an appropriate prior that precisely describes quantized latent variables is needed [4, 9, 26]. Fig. 1(a) demonstrates observation of latent variables grouped by channels. This vectorized perspective reveals correlations of latents that help us to find a prior. Note that latent vector comes from a specific region of an image and represents this region's visual appearance, correlations between

*Corresponding author.



(a) Factorized prior [3]. (b) Hyperprior [4, 26, 29]. (c) Discretized Gaussians [9]. (d) Vectorized prior.

Figure 2. Operational diagrams of different methods. We generalize prior as a unified multivariate Gaussian mixture.

vectors can be summarized as inter-correlation and intra-correlation. Inter-correlation comes from facts that images have spatial redundancy [29] *i.e*. vectors extracted from visually-similar regions or patches are closed together. Meanwhile, similar regions still have differences in details, resulting in intra-correlation *i.e*. covariances. Two properties guide us to find a vectorized prior which could describe two correlations by means and covariances.

Univariate priors previous works adopt may not be sufficient to describe above observations, because they process each scalar value individually and lack a whole view over vectors. In other words, adopting a vectorized prior mainly has two impacts. Firstly, it treats latents as vectors along channels other than scalars, helping to summarize inter- and intra-correlations. Secondly, vectorized processing has the potential to speed up compression procedure. Therefore in this paper, we propose a novel vectorized prior for variational image compression. Specifically, a unified multivariate Gaussian mixture is proposed to describe latents. Then, a probabilistic vector quantization with cascaded estimation is designed to effectively and efficiently estimate means and covariances without context models involved. Multi-codebooks are further incorporated into quantization to reduce complexity and enable flexible rate control. The whole procedure is demonstrated in Fig. 2(d) and our contribution is summarized below:

1. We propose a new vectorized perspective for variational image compression. Unlike previous works, ours considers correlations between latent vectors and formulates a unified multivariate Gaussian mixture. We further propose a probabilistic vector quantization with cascaded estimation to estimate means and covariances.

2. A multi-codebook structure is further incorporated into quantization to reduce complexity and enable flexible rate control. Overall framework is able to perform effective and efficient compression with the help of vectorized prior.

3. Extensive experiments on benchmark datasets reveal impacts of vectorized prior. Compared to state-of-the-art, our method achieves better rate-distortion performance with an impressive $3.18\times$ speed up for compression latency. These results reveal possibility to provide practical variational image compression with vectorized prior.

2. Related Works

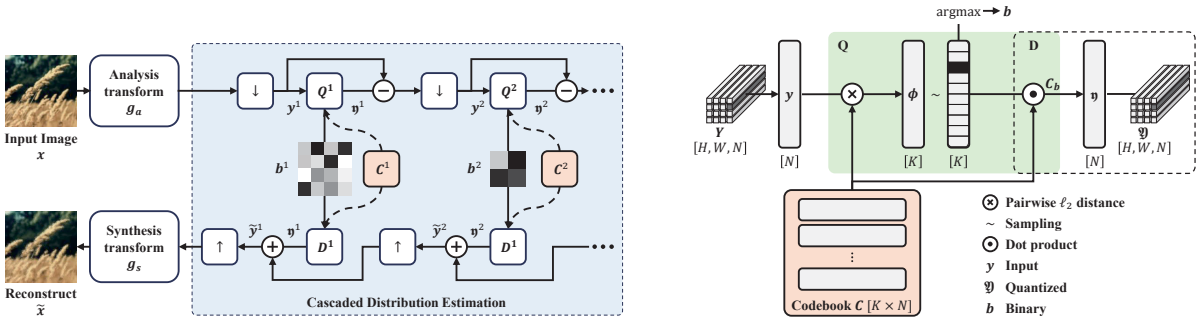
This paper focuses on variational image compression. Formally, this approach utilizes an auto-encoder to process latents in order to compress images. Studies focus on handling trade-off between rate and distortion. Specifically, latents are quantized by rounding to the nearest integer [3] or prototype [25] in order to perform entropy coding with *e.g*. range coder. Ballé *et al*. [3] propose an entropy model and train network end-to-end (Fig. 2(a)). Subsequently, Hyperprior model [4] performs variational inference by hyperprior prediction. Figs. 2(b) and 2(c) give two mainstream styles of hyperpriors. The first [4, 26] is under a shifted and scaled Gaussian distribution, while the second [9] generalizes distribution to Gaussian mixture. Both of them could take an auxiliary context model [9, 26, 29, 32] for precise estimation and further reduce compression rate.

Other than scalar quantization they adopt, a vector quantization (VQ) is adapted to our proposed vectorized prior. Studies on VQ for image compression have a long history early to 1980s [2, 15, 27]. The core problem of VQ to integrate into deep networks is to tackle the non-differentiable $\arg\max$ operation involved in quantization. Agustsson *et al*. [1] relax $\arg\max$ to $\mathrm{Softmax}$ and propose a soft-to-hard end-to-end quantization. Van den Oord *et al*. [36] and Esser *et al*. [13] instead utilize a straight-through estimator and directly pass quantized latents to decoder. Similar approaches are also applied to many other tasks [5, 14, 34].

3. Proposed Method

In this section, we firstly give preliminaries and overall demonstration of our proposed method.

Given an arbitrary image $\boldsymbol{x}$, variational image compression takes an analysis transform g_a to produce latent variable $\boldsymbol{y} = \mathrm{g_a}\left(\boldsymbol{x}\right)$, which will be quantized $\hat{\boldsymbol{y}} = \mathrm{q}\left(\boldsymbol{y}\right)$. A synthesis transform g_s restores $\hat{\boldsymbol{x}} = \mathrm{g_s}\left(\hat{\boldsymbol{y}}\right)$ from $\hat{\boldsymbol{y}}$. Distortion between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ is measured by a perceptual metric $\mathrm{d}\left(\boldsymbol{x}, \hat{\boldsymbol{x}}\right)$. Meanwhile, size of compressed $\hat{\boldsymbol{y}}$ is controlled by an entropy model $p_{\hat{\boldsymbol{y}}}$. Therefore, trade-off between rate:



(a) (b)

Figure 3. (a) Proposed network utilizes cascaded estimation with probabilistic vector quantization (Q) and reverse (D) to model vectorized prior. "↓", "↑" denotes down- and up-sampling blocks. (b) Proposed probabilistic vector quantization constructs Categorical distribution parameterized by $\boldsymbol{\phi}$ to sample $\boldsymbol{b}$ and quantize $\boldsymbol{y}$.

$$\min_{g_a, g_s} \mathcal{R} = \mathbb{E}_{\boldsymbol{x}} \left[-\log_2 p_{\hat{\boldsymbol{y}}} \left(\hat{\boldsymbol{y}} \right) \right] \tag{1}$$

and distortion:

$$\min_{g_a, g_s} \mathcal{D} = \mathbb{E}_{\boldsymbol{x}} \left[\mathrm{d} \left(\boldsymbol{x}, \hat{\boldsymbol{x}} \right) \right] \tag{2}$$

is the essential optimization objective. To enable end-to-end training, above compressed values $\hat{\cdot}$ are approximated by $\tilde{\cdot}$.

Framework. We put overall framework in Fig. 3(a). Specifically, analysis and synthesis transforms are similar as [9] where residual and attention blocks are involved. Then, to quantize and transmit latent variables, cascaded estimation adopts a series of down-sampling or up-sampling blocks followed by probabilistic vector quantization Q or dequantization D, respectively. Take a look at a single Q^ℓ at level ℓ, it accepts latent $\boldsymbol{y}^\ell \subseteq \mathbb{R}^{h^\ell \times w^\ell \times N}$ with N channels, $h^\ell \times w^\ell$ size, then produces intermediate latent $\boldsymbol{\mathfrak{y}}^\ell$ in same shape using codebook $\boldsymbol{C}^\ell \subseteq \mathbb{R}^{K \times N}$. Corresponding binary code $\boldsymbol{b}^\ell \subseteq \{0,1\}^{h^\ell \times w^\ell \times \log_2 K}$ is transmitted to the decoder side, and residual $\boldsymbol{y}^\ell - \boldsymbol{\mathfrak{y}}^\ell$ is passed to the next level.

D^ℓ does symmetrical thing. It restores $\boldsymbol{\mathfrak{y}}^\ell$ by $\boldsymbol{C}^\ell_{\boldsymbol{b}^\ell}$. Then, $\boldsymbol{\mathfrak{y}}^\ell$ and upper level $\tilde{\boldsymbol{y}}^{\ell+1}$ are added up to get $\tilde{\boldsymbol{y}}^\ell$. Therefore, the core pipeline of encoding and decoding is defined as following recursive functions:

$$\begin{cases} \left(\boldsymbol{\mathfrak{y}}^\ell,\ \boldsymbol{b}^\ell \right) = \mathrm{Q}^\ell \left(\boldsymbol{y}^\ell;\ \boldsymbol{C}^\ell \right), \\ \quad \boldsymbol{y}^{\ell+1} = \left(\boldsymbol{y}^\ell - \boldsymbol{\mathfrak{y}}^\ell \right)_\downarrow, \ 1 \le \ell \le L, \end{cases} \tag{3}$$

$$\begin{cases} \quad \boldsymbol{\mathfrak{y}}^\ell = \mathrm{D}^\ell \left(\boldsymbol{b}^\ell;\ \boldsymbol{C}^\ell \right), \\ \quad \tilde{\boldsymbol{y}}^\ell = \boldsymbol{\mathfrak{y}}^\ell + \left(\tilde{\boldsymbol{y}}^{\ell+1} \right)_\uparrow, \ 1 \le \ell < L, \end{cases} \tag{4}$$

where $(\cdot)_\downarrow, (\cdot)_\uparrow$ denote down-sampling and up-sampling.

Explaining these equations requires us to give definition of vectorized prior (Sec. 3.1.1), way to perform quantization and estimation (Sec. 3.1.2) and a generalization on prior by cascaded estimation (Sec. 3.1.3).

3.1. Unified Multivariate Gaussian Mixture

3.1.1 Vectorized Prior

An intuition to work with $\boldsymbol{y}^1$ is to group it by channels: $\boldsymbol{Y} = \{\boldsymbol{y}^1_j \subseteq \mathbb{R}^N, 1 \le j \le h^\ell w^\ell\}$ where j is the spatial location in latent feature map. For simplicity, we rearrange $\boldsymbol{y} \in \boldsymbol{Y}$ as a N-dim vector. Such arrangement helps to define $p_{\boldsymbol{Y}}(\boldsymbol{y})$ as a mixture of N-dim multivariate Gaussians:

$$\begin{gathered} p_{\boldsymbol{Y}}(\boldsymbol{y}) = \sum_{k=1}^{K} \boldsymbol{\Phi}_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \\ \textit{where } \boldsymbol{\Phi} \sim \mathrm{Categorical}(K, \boldsymbol{\phi}). \end{gathered} \tag{5}$$

Here, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are mean and covariance matrix of the k-th Gaussian component. $\boldsymbol{\Phi}$ represents a mixture parameterized by K-Categorical distribution with un-normalized log-probabilities $\boldsymbol{\phi}$.

The given vectorized prior is based on two kinds of correlations we summarize from $\boldsymbol{y}$. Fig. 1(a) reveals these by UMAP projection with $\boldsymbol{y}$ that directly extracted from backbone. Firstly, inter-correlations between $\boldsymbol{y}$s show similarities or visual redundancies *i.e.* extracted latent vectors are close if their original visual pattern are similar. This helps to cluster $\boldsymbol{y}$s into several distinct Gaussian components where cluster centroids are equivalent to means $\boldsymbol{\mu}_k$. Secondly, vectors clustered in a same component are not identical but have covariance $\boldsymbol{\Sigma}_k$, since they still have subtle differences. To further quantize vectors in $\boldsymbol{Y}$, a vector quantization to estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is needed.

3.1.2 Probabilistic Vector Quantization

We propose a learnable, probabilistic vector quantization that makes an *approximation* on above distribution, which is demonstrated in Fig. 3(b). Specifically, it maintains a codebook $\boldsymbol{C} \subseteq \mathbb{R}^{K \times N}$ consists of K codewords. Input $\boldsymbol{y}$ is quantized by assigning a specific codeword to it, which is

expressed as the following discrete conditional distribution:

$$p_{\mathfrak{Y}|\boldsymbol{Y}}(\mathfrak{y} \mid \boldsymbol{y};\ \boldsymbol{C}) = \prod_{k=1}^{K} \zeta(\boldsymbol{\phi})_k^{\mathbb{1}\{\mathfrak{y}=\boldsymbol{C}_k\}}, \quad \text{where } \boldsymbol{\phi}_k = -\left\|\boldsymbol{y} - \boldsymbol{C}_k\right\|_2^2,\ 1 \le k \le K. \tag{6}$$

Correspondingly, $\mathfrak{Y}$ is set of centroids $\mathfrak{y}$. $p_{\mathfrak{Y}|\boldsymbol{Y}}$ formulates a Categorical distribution where $\mathfrak{y}$ is assigned to the k-th codeword with probability $\zeta(\boldsymbol{\phi})_k$. ζ the $\mathrm{Softmax}$ function, $\boldsymbol{\phi}$ the negative Euclidean distance between $\boldsymbol{y}$ and codeword, $\mathbb{1}\{\cdot\}$ the characteristic function. To obtain $\mathfrak{y}$, we sample above distribution:

$$\mathfrak{y} \sim \mathrm{Q}(\boldsymbol{y};\ \boldsymbol{C}) = p_{\mathfrak{Y}|\boldsymbol{Y}}(\mathfrak{y} \mid \boldsymbol{y};\ \boldsymbol{C}) \tag{7}$$

which results in one-of-K codeword of $\boldsymbol{C}$. Intuitively, probability to choose $\boldsymbol{C}_k$ will be high if $\boldsymbol{y}$ is close to $\boldsymbol{C}_k$.

After a sample is drawn from $p_{\mathfrak{Y}|\boldsymbol{Y}}$, $\boldsymbol{b}$ is immediately obtained by index of picked codeword, which will be encoded into binary stream to transmit. On decoding side, D retrieves identical picked codeword by $\boldsymbol{C}_{\boldsymbol{b}}$ to restore $\mathfrak{y}$ since codebook $\boldsymbol{C}$ is a shared parameter between Q and D.

Above quantization defines a probabilistic model. By minimizing Eq. (2), codewords in $\boldsymbol{C}$ is derived to approximately estimate means of Gaussian components of $p_{\boldsymbol{Y}}$[1]:

$$\boldsymbol{C}_k \coloneqq\approx \mathbb{E}\{\boldsymbol{y} \in \boldsymbol{Y} \mid \boldsymbol{\Phi}_k = 1\} = \boldsymbol{\mu}_k \tag{8}$$

which automatically perform alignment between codewords and means. Compared to commonly used k-means, the proposed quantization chooses codeword stochastically other than directly pick the nearest one in a deterministic way. It models partial of $p_{\boldsymbol{Y}}(\boldsymbol{y})$ and aggregates into codebook. Moreover, introduced randomness may help network to escape the local optima during training.

3.1.3 Cascaded Estimation

It is worth noting that above proposed quantization is unable to estimate covariance matrix $\boldsymbol{\Sigma}$ according to previous derivation. Noticed that:

$$\boldsymbol{\Sigma}_k = \mathbb{E}\left\{(\boldsymbol{Y}_k - \boldsymbol{\mu}_k)(\boldsymbol{Y}_k - \boldsymbol{\mu}_k)^{\intercal}\right\}, \quad \text{where } \boldsymbol{Y}_k = \{\boldsymbol{y} \in \boldsymbol{Y} \mid \boldsymbol{\Phi}_k = 1\}. \tag{9}$$

An intuition is raised to tackle this by designing a residual connection, since:

$$\begin{aligned} \mathbb{E}\left\{\boldsymbol{y} - \mathfrak{y} \mid \boldsymbol{\Phi}_k = 1\right\} &= \mathbb{E}\left\{\boldsymbol{Y}_k - \boldsymbol{C}_k\right\} \\ &\approx \mathbb{E}\left\{\boldsymbol{Y}_k - \boldsymbol{\mu}_k\right\}. \end{aligned} \tag{10}$$

That is why Eqs. (3) and (4) are proposed. We take former level's $\boldsymbol{y} - \mathfrak{y}$ as inputs of latter level, and let latter level's neural network to predict $\boldsymbol{\Sigma}$. Fig. 1(b) tells us a trick to assume residuals on every level to be also under Gaussian mixture, helping us to expand Eq. (5) and give completed definition of the ***unified multivariate Gaussian mixture***:

$$p_{\boldsymbol{Y}^\ell|\boldsymbol{Y}^{\ell+1}}\left(\boldsymbol{y}^\ell \mid \boldsymbol{y}^{\ell+1}\right) = \sum_{k=1}^{K} \boldsymbol{\Phi}_k^\ell \mathcal{N}\left(\boldsymbol{\mu}_k^\ell, \boldsymbol{y}^{\ell+1}\right) \tag{11}$$

and model the compressed signal $\tilde{\boldsymbol{y}}$ by:

$$p_{\tilde{\boldsymbol{Y}}^\ell|\tilde{\boldsymbol{Y}}^{\ell+1}}\left(\tilde{\boldsymbol{y}}^\ell \mid \tilde{\boldsymbol{y}}^{\ell+1}\right) = \sum_{k=1}^{K} \boldsymbol{\Phi}_k^\ell \mathcal{N}\left(\boldsymbol{C}_k^\ell, \tilde{\boldsymbol{y}}^{\ell+1}\right). \tag{12}$$

We should emphasize that "$\boldsymbol{y}^{\ell+1}$", "$\tilde{\boldsymbol{y}}^{\ell+1}$" here are not strictly covariance matrices but are used to estimate covariance. Restoration of $\tilde{\boldsymbol{y}}$ starts from $\tilde{\boldsymbol{y}}^L$, and produces $\tilde{\boldsymbol{y}}^\ell$ level-by-level according to Eq. (4).

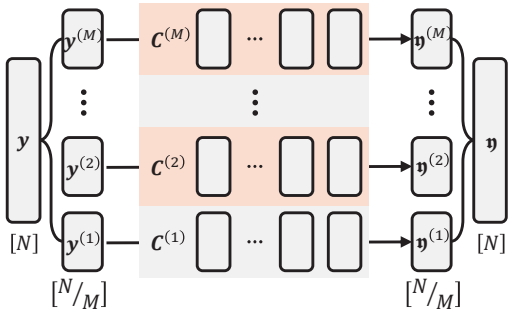


Figure 4. Multi-codebook structure. $\boldsymbol{y}$ is split into M groups and quantize them separately with sub-codebooks. Each sub-codebook parameterizes an individual distribution to model $\boldsymbol{y}^{(m)}$.

3.2. Reduce Complexity with Multi-Codebooks

We could handle above quantization by maintaining a codebook $\boldsymbol{C}^\ell$ on each level. If all of them have codebook size K, codebook size will be $L \cdot K \cdot N$ and output $\boldsymbol{b}$ for any vector has a maximum bit-length of $\log_2 K$. Unfortunately, K is not allowed to be extremely large otherwise network is unaffordable heavy. Considering trade-off between model complexity and compress ability, we further utilize multi-codebooks to generalize our method. As Fig. 4 shows, $\boldsymbol{y}^\ell$ is sliced into M groups along channels. Each piece $\boldsymbol{y}^{(\ell,m)}$ is quantized by individual sub-codebook $\boldsymbol{C}^{(\ell,m)}$ whose total size is still $L \cdot K \cdot M \cdot {}^N\!/_M = L \cdot K \cdot N$.

Introduced multi-codebook structure has several impacts. Firstly, since each part $\boldsymbol{y}^{(\ell,m)}$ has a choice out of K codewords, the set of all possible combinations of codebook $\boldsymbol{C}^{(\ell)}$ is a Cartesian product of sub-codebooks:

$$\boldsymbol{C}^{(\ell)} = \boldsymbol{C}^{(\ell,1)} \times \boldsymbol{C}^{(\ell,2)} \times \cdots \times \boldsymbol{C}^{(\ell,M)} \tag{13}$$

which makes the maximum bit length become $M \log_2 K = \log_2 K^M$ with significantly small size of codebook $M \cdot$

[1]The proof is placed in the supplementary materials.

K. Secondly, with multi-codebooks, we could generalize Eq. (11) to be a combination of several individual multivariate Gaussian mixture. $M = 1$ gives the original Eq. (11), while $M = N$ degenerates to univariate prior.

L, K, M are hyper-parameters for us to control rate. In practice, introducing multi-codebooks will not significantly downgrade performance with much smaller codebook size compared to $M = 1$ quantization under same bit-length.

3.3. Compression

At inference time, encoding and decoding are composed as follows: On encoder side, latents are quantized and binaries are rolled out by greedy assignments:

$$\boldsymbol{b}^{(\ell,m)} = \arg\max_{k} - \left\| \boldsymbol{y}^{(\ell,m)} - \boldsymbol{C}_{k}^{(\ell,m)} \right\|_2, \tag{14}$$

$$\mathfrak{y}^{(\ell,m)} = \boldsymbol{C}_{\boldsymbol{b}^{(\ell,m)}}^{(\ell,m)}, \tag{15}$$

which consumes $\mathcal{O}\left(K \cdot {}^{N}/_{M}\right)$ time complexity for encoding a single vector. $\boldsymbol{b}^{(\ell,m)}$ is compressed based on estimated occurrence frequency. As for decoder, restoration of $\mathfrak{y}^{(\ell,m)}$ only involves $\mathcal{O}\left(1\right)$ lookup according to Eq. (14). Last but not least, these operations are highly paralleled which is GPU-friendly, gives us ability to perform high-efficient encoding and decoding in actual developments.

4. Discussions

In this section, we handle a few questions about model design and compare our proposed method with other works.

Training. The model is trained in an end-to-end manner. However, to achieve this, our quantization (Sec. 3.1.2) utilizes stochastic computation graph for sampling, which is intractable to optimize. Fortunately, there are many studies to handle it. In our experiments, Gumbel reparameterization with straight-through estimator [18] has the best performance. Overall optimization is formulated as follows:

$$\mathcal{L} = \mathcal{D} = \mathrm{d}\left(\boldsymbol{x}, \tilde{\boldsymbol{x}}\right), \ \Theta \leftarrow \Theta - \eta \nabla_{\Theta} \mathcal{L}, \tag{16}$$

where Θ is the set of all trainable parameters in network and η is learning-rate. Such optimization can be done by any gradient-based optimizers.

Controlling the size of compressed binaries. The above objective only involves distortion but not rate. The reason is based on how we control size of compressed binaries, which is determined by $\boldsymbol{b}$. As aforementioned, the theoretical upper bound size of $\boldsymbol{b}$ is derived as $\sum_l M \cdot \log_2 K \cdot h^\ell \cdot w^\ell$ for all levels and all groups. Different from previous works, this upper bound is much smaller (which will be revealed in Sec. 5). We benefits from this to control bit rate by varying L, M, K or adjusting latent feature map size h^ℓ, w^ℓ. Then the rate of encoded binaries will gradually approach theoretical upper bound as training progresses without explicit objective to control it.

№	N	L	M	K	sup bpp
1	128	3	2	$[8192, 2048, 512]$	0.1274
2			6		0.3823
3	192		9		0.5098
4			12		0.7646
5			16		1.0195

Table 1. Model specifications target different rates. Empirically, we set $N = 128$ for small models while 192 for large. $L = 3$ and $K = [8192, 2048, 512]$ for all models achieves expected results with affordable model sizes. M is varied from 2 to 16 to control bpp. Theoretical upper bounds of bpp are in the last column.

Relations to hyperprior models. Proposed method has a strong relation to hyperprior models. Minnen *et al*. [26] and Cheng *et al*. [9] also model quantized latents as a Gaussian mixture while our approach extends it to N-dim multivariate. If we set $M = N$, then our prior is degenerated to univariate version. The key differences is: Firstly, our vectorized prior provide rich statistics by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to describe latents and summarize visual redundancies. Secondly, side information $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are automatically estimates by probabilistic vector quantization and cascaded estimation. In practice, they are sufficient to perform decoding without context model involved to give a speed up for compression.

Relations to other VQ-based generative models. There are a few works on compressing or generating images with help of VQ, *e.g*., SHVQ [1], VQ-VAE(-2) [31,36] and VQ-GAN [13]. Generally, they employ a k-means style quantizer which assigns the closest codeword to latent as we have discussed in Sec. 3.1.2. In order to perform end-to-end training, codebook is updated by two-stage E-M style algorithms or straight-through estimators. Nevertheless, ours includes covariance of latents while theirs could not handle. Furthermore, our framework generalizes quantization by multi-codebook structure other than a global codebook.

Proposed multi-codebook structure shares similar ideas with product quantization [19], group convolution [22] and multi-head attention [37]. They are widely applied to vision/language tasks for rich feature learning with low costs.

5. Experiments

We conduct extensive experiments to evaluate effectiveness and efficiency of our proposed method. Specifically, we first show R-D performance comparisons with other methods. Then, we measure encoder and decoder latency to demonstrate the network efficiency. Other analysis *i.e*. ablation study and visualization are further given.

5.1. Setup

Training datasets. The training dataset is a chosen subset of **ImageNet** [11] combined with **CLIC** [10] Professional training set. Specifically, we filter images from Im-

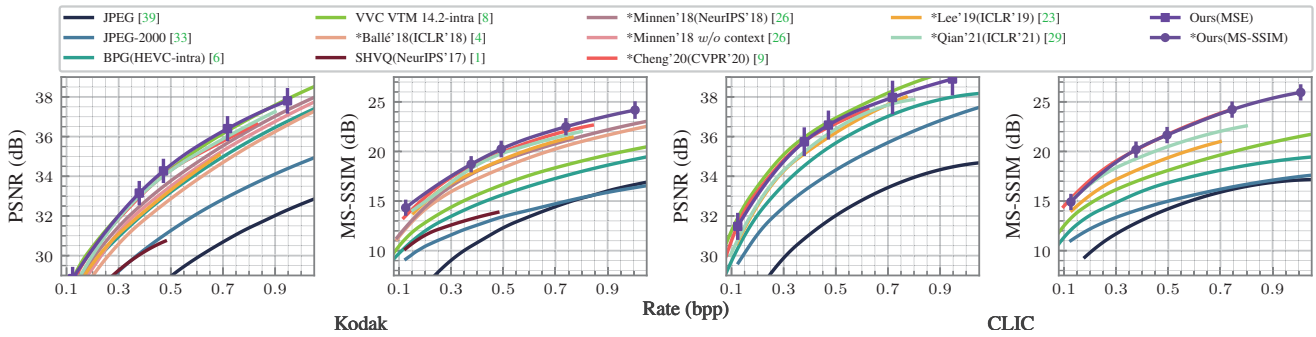


Figure 5. R-D curves on Kodak (left 2) and CLIC valid set (right 2). *: Models are optimized for MS-SSIM when with MS-SSIM metric.

Methods		Latency (ms) Encoder Abs	Encoder Rel	Decoder Abs	Decoder Rel
Ballé'18		30.66	1.09×	35.54	1.21×
Minnen'18	w/o	32.89	1.17×	36.24	1.24×
	→	2656.66	94.58×	1799.47	61.36×
	⚄	59.13	2.11×	40.40	1.38×
Cheng'20	→	2697.58	96.04×	1835.80	62.60×
	⚄	94.11	3.35×	88.04	3.00×
Ours		**28.09**	**1.00**×	**29.32**	**1.00**×

Table 2. Encoding and decoding latency comparisons for image size 768×512. For theirs, we test context-free (the firs two row) and context-enabled (row $3 \sim 6$) models. "→" means serial context model [26] while "⚄" denotes parallel [16]. Our model is № 5. Ours is the fastest model, with up to $79.32\times$ and $3.18\times$ speed up than two kinds of context-enabled models for whole compression, respectively. Ours is even faster than context-free models, since they need more than one passes to encode and decode latents.

ageNet to have more than one million pixels and randomly sample $7,415$ images from them. The whole CLIC training set with 585 images is merged ($8,000$ images in total).

Model Specs. Our method to be tested is model №$1 \sim 5$ targeting different rate by varying codebook sizes. The choices consider model complexity by tuning N, M, K and L, placed in Tab. 1. To train the model, we adopt LAMB optimizer [40]. Training images are random-cropped to 512×512 and batched into 8. Initial learning rate is set at 2×10^{-3} and annealed to 2×10^{-6} at end with cosine learning rate scheduler for $1,000$ epochs. All experiments are conducted with a single NVIDIA V100 GPU. The model is implemented with PyTorch [28].

5.2. Rate-Distortion Performance

To show rate-distortion performance, rate-distortion (R-D) points are observed and R-D curves are plotted. Specifically, to measure rate, bits-per-pixel (bpp) is calculated[2]. While for distortion, we adopt two perceptual metrics: **PSNR** and **MS-SSIM** (converted to decibels by $-10 \cdot \log_{10}(1 - value)$). Tests involve two image sets: **Kodak** [21] (24 images) and **CLIC** Professional valid set (41 images). Methods to compare include a few famous traditional standards: JPEG [39], JPEG 2000 [33], BPG [6], an upcoming new standard: VVC VTM 14.2 [8] and 6 deep image compression models: SHVQ [1], Ballé'18 [4], Minnen'18 [26], Lee'19 [23], Qian'21 [29] and Cheng'20 [9]. The R-D points are obtained from either public benchmarks or their paper[3]. For Minnen'18, both context-free and context-involved results are reported. Since two datasets have a few images, we adopt jackknife resampling and estimation strategy to report mean value and standard error on the plot by error bars [12]. More comparisons are provided in supplementary materials.

Results of Kodak and CLIC are shown in Fig. 5, respectively. For Kodak, ours outperforms state-of-the-art, while for CLIC, our model has nearly the same performance with the best deep method. Specifically, since ours adopts the same backbone as Cheng'20's, the key component to affect R-D performance is our proposed quantizers and cascaded estimators. From results we confirm our components do not hinder performance and show same or even better compress ability with state-of-the-art. Furthermore, ours achieve state-of-the-art performance without context model involved. This not only indicates effectiveness of the vectorized prior but also removes a bottleneck that slow down compression, which will be revealed in next section. Also, as rates increase, our model has a steady performance. This indicates introduced multi-codebooks are able to scale to large models by increasing M to provide satisfying performance with affordable codebook size.

5.3. Encoding and Decoding Latency

Evaluating encoding and decoding latency reveals model efficiency, which is important in actual developments. To

[2]They use various ways to control it, resulting in various bpp.

[3]https://github.com/tensorflow/compression. If not specified, models are trained using corresponding distortion metrics.

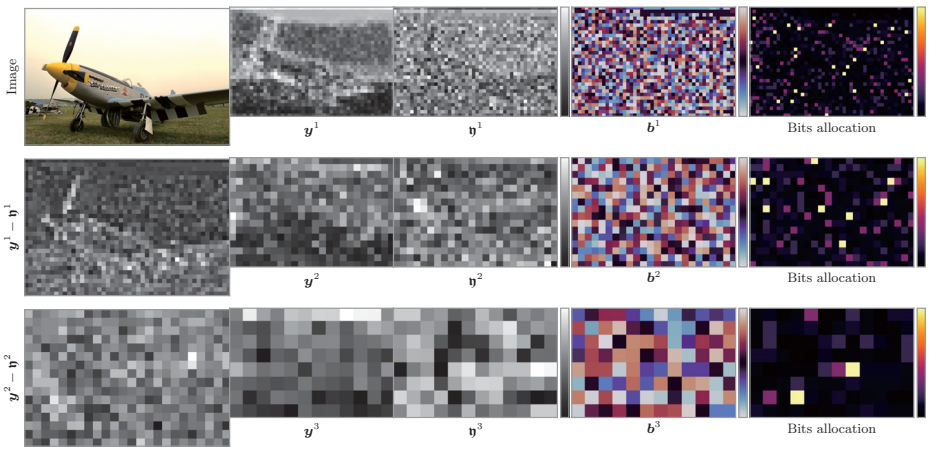


Figure 6. Visualization for a 3-level model. $\boldsymbol{y}$ is extracted latent, $\mathfrak{y}$ is quantized latents. By calculating $\left(\boldsymbol{y}^\ell - \mathfrak{y}^\ell\right)$, visual redundancies are removed. $\boldsymbol{b}$ is corresponding binary (index of picked codewords). Brighter pixels in the last column mean more bits allocation.

conduct such test, following models are adopted: Ballé'18, Minnen'18 ("w/o", "$\rightarrow$", "⚄"), and Cheng'20 ("$\rightarrow$", "⚄"). Specifically, "w/o" means no context model involved, and "$\rightarrow$", "⚄" are serial [32] and parallel [16] context model variants. Our model to be tested is № 5. To precisely measure the latency, we feed a batch of images from Kodak with size 768×512 and track the CUDA events of encoder and decoder separately. Measurements are based on their public models or reimplementations[4].

As Tab. 2 shows, our network is the fastest method among all other models. In particular, compared to models utilizing context, ours achieves up to $79.32\times$ faster than the serials and $3.18\times$ faster than the parallels for whole compression, respectively. This efficiency gap comes from our introduced cascaded estimation that do not need context model. Furthermore, our model is even faster than context-free models *i.e.* Ballé'18 and Minnen'18 (w/o), based on how we perform (de)quantization. Ours only involves $\mathcal{O}\left(K \cdot {}^{N}/_{M}\right)$ to quantize and $\mathcal{O}\left(1\right)$ to dequantize, and is highly paralleled running in GPU. Meanwhile, our encoders and decoders only require one forward pass, but they need two or more. In summary of Secs. 5.2 and 5.3, our model achieves better R-D performance with an impressive compression latency, enabling us the ability to perform practical image compression with our vectorized prior.

5.4. Ablation Study

To investigate impacts of proposed method, we conduct ablation study and report BD-rate w.r.t. original model (lower is better) and latency (Tab. 3):

Impacts of cascaded estimation. The level L reflects how much parameters involved in estimation, *e.g.*, $L = 1$ does not perform cascaded estimation ("w/o cascaded"),

[4]Tested latencies of [16] are slightly slower than their report.

Variants	BD-rate	Latency	
		Encoder	Decoder
w/o cascaded	8.87%	27.13	28.29
2-levels	2.33%	27.62	28.77
4-levels	−0.64%	28.93	30.27
one-codebook	24.40%		
cos-quantizer	4.64%		
[13]-quantizer	16.20%	28.09	29.32
[1]-quantizer	11.48%		
Ours	-		

Table 3. Ablation study on 6 variants where BD-rate w.r.t. original model (lower is better) and latency is reported. The first three row vary level L, "one-codebook" uses a global codebook. And the 5th~7th rows modify quantizers' formulation.

while $L = 4$ will add an extra residual compared to original $L = 3$ model ("4-levels"). The first three rows of Tab. 3 give results of three variants $L = 1, 2, 4$. As level increases, BD-rate continuously decreases. "2-levels" is much better than "w/o cascaded" while "4-levels" obtains nearly no improvement compared to original model. The former indicates introducing cascaded estimation actually has a positive effect, while the latter tells us setting $L = 3$ is sufficient otherwise model will be large and may hard to train. The 2nd column of table show latency between different models. Introducing more levels does not significantly slow model down, which indicates that cascaded estimation is not computational heavy for real scenario applications.

Impacts of multi-codebook structure. We use a global shared codebook as variant "one-codebook" to study impacts of multi-codebook structure. Results in the 3rd row of Tab. 3 shows significant performance downgrade when using global codebook. It proves the effectiveness of multi-

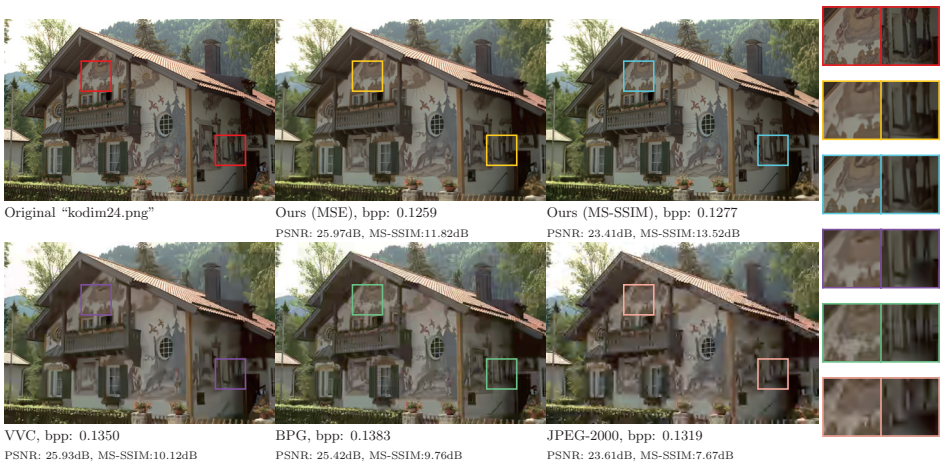


Figure 7. Visualization of "kodim24.png" for different codecs. Zoomed-in view on the right shows differences.

codebooks that model precise distributions since they adopt different parameters for different levels or groups.

Impacts of quantization. Quantization performance is affected in two way: a) Use a different similarity measure *e.g*. cosine similarity ("*cos*-quantizer") to define ϕ in Eq. (6), b) Use a deterministic quantizer *i.e*. same as [13] or [1] ("[13]-quantizer, [1]-quantizer"). The last three rows of Tab. 3 shows difference of three quantizers. "*cos*-quantizer" adopts cosine similarity which is not a distance metric since it breaks the triangle inequality. We find this may cause performance drop. When training "[13]-quantizer" or "[1]-quantizer", we find network is trapped in local-optima *i.e*. most of vectors are quantized to a few codewords, and some codes are never assigned. We think this makes two kinds of variants have performance gap with ours.

5.5. Visualization

We pick image from Kodak to show compression quality. Compared codecs are JPEG-2000, BPG and VVC. All methods are set to $bpp \approx 0.13$ while compression ratio is about $185:1$. As Fig. 7 shows, "kodim24.png" from Kodak dataset on the top-left is reference image. From zoomed-in view, we could find "Ours (MS-SSIM)" preserves more visual details, especially wall paintings and patterns. Meanwhile, it also achieves the highest MS-SSIM among all methods with the smallest bpp. Our MSE optimized model gives higher PSNR but is slightly blurred. It achieves comparable performance with VVC with a still small bpp. More perceptual measures and image comparisons are placed in supplementary materials.

We also give 2-d projection visualization of $\boldsymbol{y}^1, \boldsymbol{y}^2$ on a toy model trained with $N = 128, M = 1, L = 2, K = 32$, shown in Fig. 1. Specifically, latent vectors are extracted from 24 Kodak images and projected to 2-d points by UMAP [24]. They are colored by codewords, *i.e*., two points are with same color if they are assigned to same codeword. The visualization satisfy our vectorized prior. Latents can be clustered by these codewords (left), while residuals are under similar distribution (right). Therefore, we can induce all latents to a unified, vectorized prior.

6. Conclusion and Future Work

In this paper, we propose a novel vectorized prior for variational image compression. We demonstrate latent vectors are correlated and can be induced to a unified multivariate Gaussian mixture. To perform estimation, proposed cascaded estimation with probabilistic vector quantization effectively approximate means and covariances. Furthermore, multi-codebooks are incorporated into above components to give an efficient compression procedure. Extensive experiments confirm effectiveness and efficiency of our proposed method. Future work will focus on variable-rate control with our proposed vectorized prior.

Limitation and Broader Impacts. This work introduces a new perspective in neural image compression, which may inspire researchers to propose valuable future works. The high performance, low latency model may also benefit for real-life digital image storage or online multimedia contents. However, main limitations of our work are extra network parameters and computational resource requirement. Negative impacts involve vulnerability of model. We may give uncontrollable images under adversarial examples. Meanwhile, there seems to have no ethical issues or biases since the network is trained without supervision. However, training dataset does influence model with biased or sensitive images. Therefore, data should be checked to avoid potential issues.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Grant No. 62020106008, No. 62122018, No. 61772116, No. 61872064), Sichuan Science and Technology Program (Grant No.2019JDTD0005).

References

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *NeurIPS*, pages 1141–1151, 2017. 2, 5, 6, 7, 8

[2] R Aravind and Allen Gersho. Image compression based on vector quantization with finite memory. *Optical Engineering*, 26(7):267570, 1987. 2

[3] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *ICLR*, 2017. 1, 2

[4] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *ICLR*, 2018. 1, 2, 6

[5] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *ICLR*, 2022. 2

[6] Fabrice Bellard. Bpg image format. https://bellard.org/bpg/, 2014. 1, 6

[7] Toby Berger. Rate-distortion theory. *Wiley Encyclopedia of Telecommunications*, 2003. 1

[8] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Trans. Circuits Syst. Video Technol.*, 31(10):3736–3764, 2021. 1, 6

[9] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *CVPR*, pages 7936–7945, 2020. 1, 2, 3, 5, 6

[10] Workshop and challenge on learned image compression. http://compression.cc/, 2018. 5

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 5

[12] Bradley Efron and Charles Stein. The jackknife estimate of variance. *The Annals of Statistics*, pages 586–596, 1981. 6

[13] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, pages 12873–12883, 2021. 2, 5, 7, 8

[14] Lianli Gao, Xiaosu Zhu, Jingkuan Song, Zhou Zhao, and Heng Tao Shen. Beyond product quantization: Deep progressive quantization for image retrieval. In *IJCAI*, pages 723–729, 2019. 2

[15] Morris Goldberg, Paul R. Boucher, and Seymour Shlien. Image compression using adaptive vector quantization. *IEEE Trans. Commun.*, 34(2):180–187, 1986. 2

[16] Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *CVPR*, pages 14771–14780, 2021. 6, 7

[17] Abir Jaafar Hussain, Ali Al-Fayadh, and Naeem Radi. Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, 300:44–69, 2018. 1

[18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017. 5

[19] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011. 5

[20] J. Jiang. Image compression with neural networks - A survey. *Signal Process. Image Commun.*, 14(9):737–760, 1999. 1

[21] Kodak lossless true color image suite. http://r0k.us/graphics/kodak/, 1999. 6

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1106–1114, 2012. 5

[23] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *ICLR*, 2019. 6

[24] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 1, 8

[25] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *CVPR*, pages 4394–4402, 2018. 1, 2

[26] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*, pages 10794–10803, 2018. 1, 2, 5, 6

[27] Nasser M. Nasrabadi and Yushu Feng. Image compression using address-vector quantization. *IEEE Trans. Commun.*, 38(12):2166–2173, 1990. 2

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019. 6

[29] Yichen Qian, Zhiyu Tan, Xiuyu Sun, Ming Lin, Dongyang Li, Zhenhong Sun, Hao Li, and Rong Jin. Learning accurate entropy model with global reference for image compression. In *ICLR*, 2021. 2, 6

[30] AM Raid, WM Khedr, Mohamed A El-Dosuky, and Wesam Ahmed. Jpeg image compression using discrete cosine transform-a survey. *arXiv preprint arXiv:1405.6147*, 2014. 1

[31] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *NeurIPS*, pages 14837–14847, 2019. 5

[32] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017. 2, 7

[33] Athanassios Skodras, Charilaos A. Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Process. Mag.*, 18(5):36–58, 2001. 1, 6

[34] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic, and Heng Tao Shen. Unified binary generative adversarial

network for image retrieval and compression. *Int. J. Comput. Vis.*, 128(8):2243–2264, 2020. 2

[35] Gary J. Sullivan, Jens-Rainer Ohm, Woojin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Trans. Circuits Syst. Video Technol.*, 22(12):1649–1668, 2012. 1

[36] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, pages 6306–6315, 2017. 2, 5

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 5

[38] Gaurav Vijayvargiya, Sanjay Silakari, and Rajeev Pandey. A survey: various techniques of image compression. *arXiv preprint arXiv:1311.6877*, 2013. 1

[39] Gregory K. Wallace. The JPEG still picture compression standard. *Commun. ACM*, 34(4):30–44, 1991. 1, 6

[40] Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *ICLR*, 2020. 6