Lossy Image Compression with Quantized Hierarchical VAEs

Zhihao Duan[*] Ming Lu[†] Zhan Ma[†] Fengqing Zhu[*]
[*] Purdue University, West Lafayette, Indiana, U.S.
[†] Nanjing University, Nanjing, Jiangsu, China
duan90@purdue.edu, luming@smail.nju.edu.cn, mazhan@nju.edu.cn, zhu0@purdue.edu [*]

Abstract

Recent work has shown a strong theoretical connection between variational autoencoders (VAEs) and the rate distortion theory. Motivated by this, we consider the problem of lossy image compression from the perspective of generative modeling. Starting from ResNet VAEs, which are originally designed for data (image) distribution modeling, we redesign their latent variable model using a quantization-aware posterior and prior, enabling easy quantization and entropy coding for image compression. Along with improved neural network blocks, we present a powerful and efficient class of lossy image coders, outperforming previous methods on natural image (lossy) compression. Our model compresses images in a coarse-to-fine fashion and supports parallel encoding and decoding, leading to fast execution on GPUs. Code is made available online.

1. Introduction

Data (in our context, image) compression and generative modeling are two fundamentally related tasks. Intuitively, the essence of compression is to find all "patterns" in the data and assign fewer bits to more frequent patterns. To know exactly how frequent each pattern occurs, one would need a good probabilistic model of the data distribution, which coincides with the objective of (likelihood-based) generative modeling. This connection between compression and generative modeling has been well established, both theoretically and experimentally, for the *lossless* setting. In fact, many of the modern image generative models are also the best-performing lossless image coders [43, 56].

A similar connection can be drawn for the *lossy* compression setting. In particular, a popular class of image generative models, variational autoencoders (VAEs) [19], has been proved to have a rate distortion (R-D) theory interpretation [2, 54]. With a distortion metric specified, VAEs learn to "compress" data by minimizing a *tight* upper bound on

[*]Dr. Z. Ma is partially funded by Microsoft Research Asia.

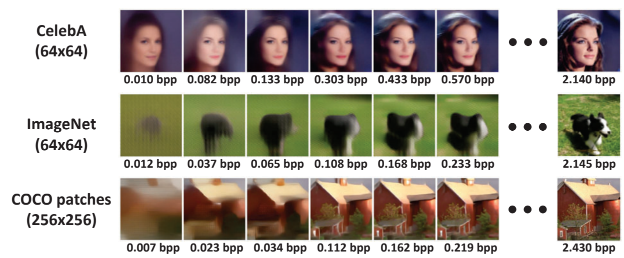


Figure 1: Our **QRes-VAE** (for quantized ResNet VAE) image coder learns a hierarchy of features and encode/decode images in a coarse-to-fine fashion. Better viewed by zooming in. Models are trained separately on each dataset.

their information R-D function [54], showing great potential for application to lossy image compression. However, existing best-performing VAEs [45, 11] employ continuous latent variables, which cannot be straightforwardly coded into bits, and thus cannot be used for practical image compression. Although several methods have been developed to turn VAEs into practical coders, *e.g.*, by communicating samples [14] and post-training quantization [53], neither achieved satisfactory R-D performance compared to existing lossy image coders.

Despite the lack of a practical coding algorithm, the potential of VAEs in lossy compression has also been reflected in the image coding community. Although independently developed from the perspective of *transform coding*, many state-of-the-art lossy image coders resemble a simple VAE in which the latent variables are first-order Markov [6]. Given that such simple VAEs are shown to be suboptimal in generative image modeling [39], we hypothesize that a more powerful VAE architecture, *e.g.*, *hierarchical VAEs*, would also achieve a better lossy compression performance.

Motivated by this, we adopt hierarchical VAE architectures that are originally designed for generative image modeling for lossy compression. We redesign the latent variable model to allow easy quantization and practical entropy coding, in a way similar to existing learned lossy image coders [6, 30]. Specifically, we start from a popular family of hierarchical VAEs, *ResNet VAEs* [20], and introduce modifications including 1) a uniform posterior, 2) a Gaussian convolved with uniform prior, and 3) revised network architectures. Our new model, **QRes-VAE** (for quantized ResNet VAE), achieves better R-D performance on natural image compression than existing state-of-the-art coders. Furthermore, our model compresses images in a coarse-to-fine manner (Fig. 1) thanks to its hierarchical architecture, while avoids the slow sequential encoding/decoding as experienced by (spatially) autoregressive image coders [30].

Our contributions are summarized as follows. We propose to use a quantization-aware latent variable model for modern hierarchical VAEs, making practical image coding feasible. We present a powerful and efficient lossy image coder that outperforms previous hand crafted and learned methods. Our method narrows the gap between image compression and generation, at the same time providing insights into designing better image compression systems.

2. Background and Related Work

In this section, we briefly summarize the preliminaries, introduce our notation, and review previous works.

2.1. Variational Autoencoder (VAE)

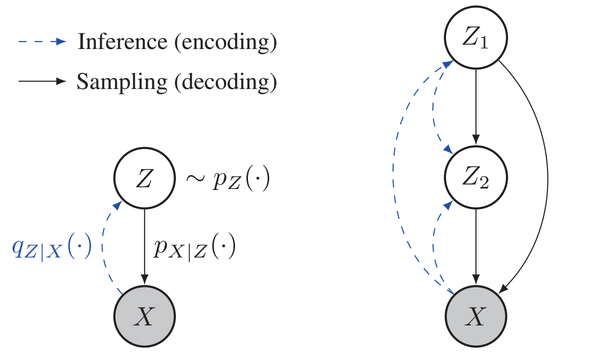
Let X be the random variable representing data (in our case, natural images), with an unknown data distribution $p_{\text{data}}(\cdot)$. VAEs [19] model the data distribution by assuming a latent variable model as shown in Fig. 2a, where Z is the assumed latent variable. In VAEs, there is a prior $p_Z(\cdot)$, a conditional data likelihood (or decoder) $p_{X|Z}(\cdot)$ for sampling, and an approximate posterior (or encoder) $q_{Z|X}(\cdot)$ for variational inference. Note that we omit the model parameters for simplified notation.

The training objective of VAE is to minimize a variational upper bound on the negative log-likelihood:

$$\begin{aligned}\mathcal{L} &= D_{\text{KL}}(q_{Z|x} \parallel p_Z) + \mathbb{E}_{q_{Z|x}}\left[\log \frac{1}{p_{X|Z}(x|Z)}\right] \\ &\geq -\log p_X(x),\end{aligned} \tag{1}$$

where x is a an image, $q_{Z|x}$ is a shorthand notation for $q_{Z|X}(\cdot|x)$, and the minimization of $\mathcal{L}$ is w.r.t. the model parameters of $q_{Z|X}(\cdot)$, $p_Z(\cdot)$, and $p_{X|Z}(\cdot)$.

Hierarchical VAEs: To improve the flexibility of VAEs, the latent variable is often divided into disjoint groups, $Z \triangleq \{Z_1, Z_2, ..., Z_N\}$ where N is the number of groups, to form a *hierarchical VAE*. Many best-performing VAEs [45, 11, 37] follow a ResNet VAE [20] architecture, whose probabilistic model is shown in Fig. 2b. During sampling, each latent variable Z_i is conditionally dependent on all previous variables $Z_{<i}$, where $i \in \{1, 2, ..., N\}$ is the index, and during inference, Z_i is also conditionally dependent on X



(a) VAE (b) 2-layer ResNet VAE [20]

Figure 2: **Probabilistic model of VAEs**, where X represents data, and Z denotes latent variable(s). In this work, we use a 12-layer ResNet VAE for image compression.

in addition to $Z_{<i}$. For notational simplicity, we define

$$\begin{aligned} q_i(\cdot) &\triangleq q_{Z_i|X,Z_{<i}}(\cdot) \\ p_i(\cdot) &\triangleq p_{Z_i|Z_{<i}}(\cdot) \end{aligned} \tag{2}$$

to denote the approximate posterior and prior distributions for Z_i, respectively. We also define $Z_{<1}$ to be an empty set, so we have $p_1(\cdot) \triangleq p_{Z_1}(\cdot)$ and $q_1(\cdot) \triangleq q_{Z_1|X}(\cdot)$.

Discrete VAEs: A number of works have been proposed to develop VAEs with a discrete latent space. Among them, vector quantized VAEs (VQ-VAE) [47, 34] are able to generate high-fidelity images, but they rely on a separately learned prior and cannot be trained end-to-end. Several works extend VQ-VAEs to form a hierarchy [51, 50], but they suffer from the same problem as VQ-VAE. Another line of work, Discrete VAEs (DVAE) [35, 46, 44], assumes binary latent variables and use Boltzmann machines as the prior. However, none of them are able to scale to high resolution images. In this work, we tackle these issues through test-time quantization.

Data compression with VAEs: VAEs have a very strong theoretical connection to data compression. When X is discrete, VAEs can be directly used for lossless compression [42, 21] using the bits-back coding algorithm [18]. When X is continuously valued, the VAE objective (Eq. 1) has a rate-distortion (R-D) theory interpretation [2] and has been used to estimate the information R-D function for natural images [54]. However, there lacks an entropy coding algorithm to turn VAEs into practical lossy coders. Various works [1, 14, 40, 15] have been conducted towards this goal, but they either require an intractable execution time or do not achieve competitive performance compared to existing lossy image coders. In this work, we provide another pathway for turning VAEs into practical coders, *i.e.*, by using a quantization-aware probabilistic model for latent variables.

2.2. Lossy Image Compression

Most existing lossy image coders follows the *transform coding* [16] paradigm. Conventional, handcrafted image codecs such as JPEG [48], JPEG 2000 [38], BPG [23], VVC intra [32] use orthogonal linear transformations such as discrete cosine transform (DCT) and discrete wavelet transform (DWT) to decorrelate the image pixels before quantization and coding. Learning-based coders achieve this in a non-linear way [3] by implementing the transformations and an entropy model using neural networks. To improve compression efficiency, most previous works either explore efficient neural network blocks, such as residual networks [10], self-attention [9], and transformers [28, 57], or develop more expressive entropy models, such as hierarchical [6] and autoregressive models [30, 31, 17].

Interestingly, the learned transform coding paradigm can be equivalently viewed as a simple VAE [41, 6], where the encoder (with additive noise), entropy model, and decoder in transform coding correspond to the posterior, prior, and conditional data likelihood in VAEs, respectively. Along this direction, we propose to use a more powerful hierarchical VAE architecture for lossy image compression.

3. Quantization-Aware Hierarchical VAE

In this section, we first present our overall network architecture in Sec. 3.1. Next, we describe our probabilistic model and loss function in Sec. 3.2. We detail how to implement practical image coding in Sec. 3.3. Finally, we provide discussion in Sec. 3.4, highlighting the relationship of our approach to previous methods.

3.1. Network Architecture

Our overall architecture is similar to the framework of ResNet VAEs [20, 45, 11], on which we made modifications to adapt them for practical compression. We briefly describe our network architecture in this section and refer readers to the Appendix 6.1 and our code for full implementation details.

Overall architecture: Fig. 3a overviews the general framework of ResNet VAE, which consists of a bottom-up path and a top-down path. Given an input image x, the bottom-up path produces a set of deterministic features, which are subsequently sent to the top-down path for (approximate) inference. In the top-down path, the model starts with a learnable constant feature and then pass it into a sequence of *latent blocks*, where each latent block adds "information", carried by the latent variables z_i, into the feature. At the end of the top-down path, a reconstruction $\hat{x}$ is predicted by an upsampling layer. Our latent blocks behaves differently for training, compression, and decompression, which we describe in details in Sec. 3.2 and Sec. 3.3.

Network components: We use patch embedding [12] (*i.e.*, convolutional layer with the stride equal to kernel size) for downsampling operations. We use sub-pixel convolution [36] (*i.e.*, 1x1 convolution followed by pixel shuffling) for upsampling, which can be viewed as an "inverse" of the patch embedding layer. Our choices for the downsampling/upsampling layers are motivated by their success in computer vision tasks [12, 36] as well as their efficiency compared to overlapped convolutions. The residual blocks in our network can be chosen arbitrarily in a plug-and-play fashion. We use the ConvNeXt [27] block as our choice, since we empirically found it achieves better performance than alternatives (see ablation study in Sec. 4.4).

3.2. Probabilistic Model and Training Objective

As we mentioned in Sec. 2, VAEs with continuous latent variables cannot be straightforwardly used for lossy compression (due to the lack of an entropy coding algorithm). We also note that a combination of a) uniform quantization at test time and b) additive uniform noise at training time allows easy entropy coding, as well as having an elegant VAE interpretation [5, 6]. Inspired by this, we redesign the probabilistic model in ResNet VAEs, using uniform posteriors to enable quantization-aware training. For the prior, we use a Gaussian distribution convolved with uniform distribution, which has enough flexibility to match the posterior [30]. Note that under this configuration, the probabilistic model in each latent block closely resembles the mean & scale hyperprior entropy model [30].

We now give formal description for our probabilistic model. In this section, we assume every variable is a scalar in order to simplify the presentation. In implementation, we apply all the operations element-wise.

Posteriors: The (approximate) posterior for the i-th latent variable, Z_i, is defined to be a uniform distribution:

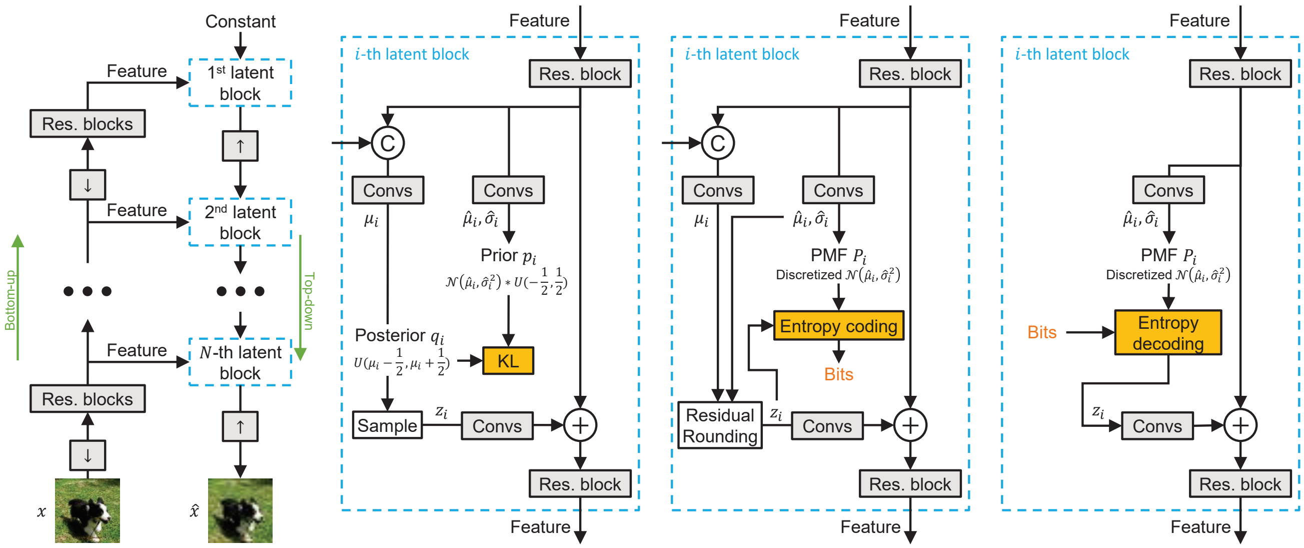
$$q_i(\cdot\,|z_{<i}, x) \triangleq U(\mu_i - \frac{1}{2}, \mu_i + \frac{1}{2}), \tag{3}$$

where μ_i is the output of the posterior branch in the i-th latent block (see Fig. 3b). From Fig. 3a and Fig. 3b, we can see that μ_i (and thus z_i) depends on the image x as well as previous latent variables $z_{<i}$.

Priors: The prior distribution for Z_i is defined as a Gaussian convolved with a uniform distribution [6]:

$$\begin{aligned} p_i(\cdot\,|z_{<i}) &\triangleq \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2) * U(-\frac{1}{2}, \frac{1}{2}) \\ \Leftrightarrow p_i(z_i|z_{<i}) &= \int_{z_i - \frac{1}{2}}^{z_i + \frac{1}{2}} \mathcal{N}(t; \hat{\mu}_i, \hat{\sigma}_i^2)\, dt \end{aligned} \tag{4}$$

where $\mathcal{N}(t; \hat{\mu}_i, \hat{\sigma}_i^2)$ denotes the Gaussian probability density function evaluated at t. The mean $\hat{\mu}_i$ and scale $\hat{\sigma}_i$ are predicted by the prior branch (see Fig. 3b) in the i-th latent block. Note that the prior mean $\hat{\mu}_i$ and scale $\hat{\sigma}_i$ are



(a) General framework (b) Latent block (training) (c) Latent block (compress) (d) Latent block (decompress)

Figure 3: **Overview of our quantized ResNet VAE (QRes-VAE)**. In (a), we show the general framework of ResNet VAEs [20]. In our implementation, we use a 12-layer (*i.e.*, 12 latent blocks) architecture for QRes-VAE. The gray blocks represent neural networks, ↑ and ↓ denote upsampling and downsampling operations, respectively, ⓒ denotes feature concatenation, and ⊕ denotes addition. The latent blocks behave differently for training, compression, and decompression, which are illustrated in (b), (c), and (d), respectively. See Sec. 3 for detailed description.

dependent only on $z_{<i}$, but not on image x. Sampling from $p_i(\cdot\,|z_{<i})$ can be done by:

$$z_i \leftarrow \hat{\mu}_i + t \cdot (\hat{\sigma}_i w + u), \tag{5}$$

where $w \sim \mathcal{N}(0,1)$ and $u \sim U(-\frac{1}{2}, \frac{1}{2})$ are random noise, and $t \in [0,1]$ is a temperature parameter to control the noise level. When $t = 1$, z_i is an unbiased sample from the prior $p_i(\cdot\,|z_{<i})$, and when $t = 0$, z_i is simply the prior mean.

Data likelihood: For lossy compression, the form of the likelihood distribution $p_{X|Z}(\cdot)$ depends on which distortion metric $d(\cdot)$ is used. In general, we define

$$p_{X|Z}(x|z) \propto e^{-\lambda \cdot d(\hat{x}, x)}, \tag{6}$$

where λ is a scalar hyperparameter that we can manually set, and $\hat{x}$ is the final output of the top-down path network (see Fig. 3a). Notice that $\hat{x}$ depends on all latent variables $z \triangleq \{z_1, ..., z_N\}$. In image compression, $d(\cdot)$ is often chosen to be the mean squared error (MSE), in which case the data likelihood forms a (conditional) Gaussian distribution.

Training objective: Given an image x, our training objective is then to minimize the loss function of VAE (Eq. 1), which can be written as:

$$\begin{aligned}\mathcal{L} &= D_{\mathrm{KL}}(q_{Z|x} \parallel p_Z) + \mathbb{E}_{q_{Z|x}}\left[\log \frac{1}{p_{X|Z}(x|Z)}\right], \\ &= \mathbb{E}_{q_{Z|x}}\left[\sum_{i=1}^{N} \log \frac{1}{p_i(z_i|z_{<i})} + \lambda \cdot d(x, \hat{x})\right] + \text{constant},\end{aligned} \tag{7}$$

where the expectation w.r.t. $Z \sim q_{Z|x}$ is estimated by drawing a sample at each training step. The detailed derivation of Eq. 7 is given in the Appendix 6.2. We can see that the first term in Eq. 7 corresponds to a continuous relaxation of the test-time bit rate (up to a constant factor $\log_2 e$), and the second term corresponds to the distortion of reconstruction, where we can tune λ to balance the trade-off between rate and distortion. Note that the model parameters should be optimized independently for each λ. That is, we use separately trained models for different bit rates.

3.3. Compression

In actual compression and decompression, the overall framework (*i.e.*, Fig. 3a) is unchanged, except the latent variables are quantized for entropy coding. Similar to the Hyperprior model [6], we quantize μ_i instead of sampling from the posterior, and we discretize the prior to form a discretized Gaussian probability mass function.

The compression process is detailed in Fig. 3c, in which the *residual rounding* operation is defined as follows:

$$z \leftarrow \hat{\mu}_i + \lfloor \mu_i - \hat{\mu}_i \rceil, \tag{8}$$

where $\lfloor \cdot \rceil$ is the nearest integer rounding function. That is, we quantize μ_i to its nearest neighbour in the set $\{\hat{\mu}_i + n \mid n \in \mathbb{Z}\}$, denoted by z_i, which we can encode into bits using the probability mass function (PMF) $P_i(\cdot)$:

$$P_i(n) \triangleq p_i(\hat{\mu}_i + n|Z_{<i}), n \in \mathbb{Z}. \tag{9}$$

It can be shown that $P_i(\cdot)$ is a valid PMF, or specifically, a discretized Gaussian [6]. Note that each of our latent block produces a separate bitstream, so a compressed image consists of N bitstreams, corresponding to the N latent variables $z_1, z_2, ..., z_N$. In the implementation, we use the range-based Asymmetric Numeral Systems (rANS) [13] for entropy coding.

Decompression (Fig. 3d) is done in a similar way. Starting from the constant feature, we iteratively compute $P_i(\cdot)$ for $i = 1, 2, ..., N$. At each step, we decode z_i from the i-th bitstream using rANS and transform z_i using convolution layers before adding it to the feature. Once this is done for all $i = 1, 2, ..., N$, we can obtain the reconstruction $\hat{x}$ using the final upsampling layer in the top-down decoder.

3.4. Relationship to Previous Methods

Our hierarchical architecture (same as ResNet VAEs) may seem similar to the Hyperprior [6] model at the first glance, but they are in fact fundamentally different in many aspects. The inference (encoding) order in the Hyperprior framework is the first order Markov, while our inference is bidirectional [20]. Similarly, the sampling order of latent variables in the Hyperprior model is also first order Markov, while in our model, $\hat{X}$ depends on all latent variables Z.

A nice property of the ResNet VAE architecture is that, if sufficiently deep, it generalizes autoregressive models [11], which are widely adopted in lossy image coders and gives strong compression performance. Note that this conclusion is not limited to *spatial* autoregressive models. Other types of autoregressive models, such as channel-wise [31] and checkerboard [17] models, can also be viewed as special cases of the ResNet VAE framework.

From the perspective of (non-linear) transform coding [3], our model transforms image x into $z \triangleq \{z_1, ..., z_N\}$ and losslessly code z instead, so z can be viewed as the transform coefficients (after quantization). Note that z contains a feature hierarchy at different resolutions, in contrast to many transform coding frameworks where only a single feature is used. Since z_i is only a quantized version of the posterior mean μ_i, we can also view $\{\mu_1, ..., \mu_N\}$ as the transform coefficients (before quantization). Unlike existing methods, there is no separate *entropy model* in our method, and instead, our top-down decoder itself acts as the entropy model for all the transform coefficients z.

4. Experiments

4.1. Implementation

We build two models based on our QRes-VAE architecture as described in Sec. 3.1, one with 34M parameters and the other with 17M parameters. We use the larger model, QRes-VAE (34M), for natural image compression, and we use the smaller one QRes-VAE (17M) for additional experiments and ablation analysis. Data augmentation, exponential moving averaging, and gradient clipping are applied during training. We list the full details of architecture configurations in the Appendix 6.1 and training hyperparameters in the Appendix 6.3.

4.2. Datasets and Metrics

COCO: We use the COCO [24] *train2017* split, which contains 118,287 images with around 640×420 pixels, for training our QRes-VAE (34M). We randomly crop the images to 256×256 patches during training.

Kodak: The Kodak [22] image set is a commonly used test set for evaluating image compression performance. It contains 24 natural images with 768×512 pixels.

CLIC: We also use the CLIC[1] 2022 test set for evaluation. It contains 30 high resolution natural images with around 2048×1365 pixels.

CelebA (64x64): We train and test our smaller model, QRes-VAE (17M), on the CelebA [26] dataset for ablation study and additional experiments. CelebA is a human face dataset that contains more than 200k images (182,637 for train/val and 19,962 for test), all of which we have resized and center-cropped to have 64×64 pixels.

Metrics: As a standard practice, we use MSE as the distortion metric $d(\cdot)$ during training, and we report the peak signal-to-noise ratio (PSNR) during evaluation:

$$\text{PSNR} \triangleq -10 \cdot \log_{10} \text{MSE}. \tag{10}$$

We measure data rate by bits per pixel (bpp). To compute the overall metrics (PSNR and bpp) for the entire dataset, we first compute the metric for each image and then average over all images. We also use the BD-rate metric [8] to compute the average bit rate saving over all PSNRs.

4.3. Main Results for Lossy Image Compression

We compare our method to open-source learned image coders from public implementations. We use VVC intra [32] (version 12.1), the current best hand crafted codec and a common lossy image coding baseline, as the anchor for computing BD-rate for all methods. Thus, our reported BD-rates represent average bit rate savings over VVC intra.

Evaluation results are shown in Fig. 4. We observe that at lower bit rates, our method is on par with previous best methods, while at higher bit rates, our approach outperforms them by a clear margin. We attribute this bias towards high bit rates to the architecture design of our model. We reconstruct the image from a feature map that is $4\times$ down-sampled w.r.t. the original image, which can preserve more high-frequency information of the image. In contrast, most previous methods aggressively predict the image

[1]http://compression.cc/

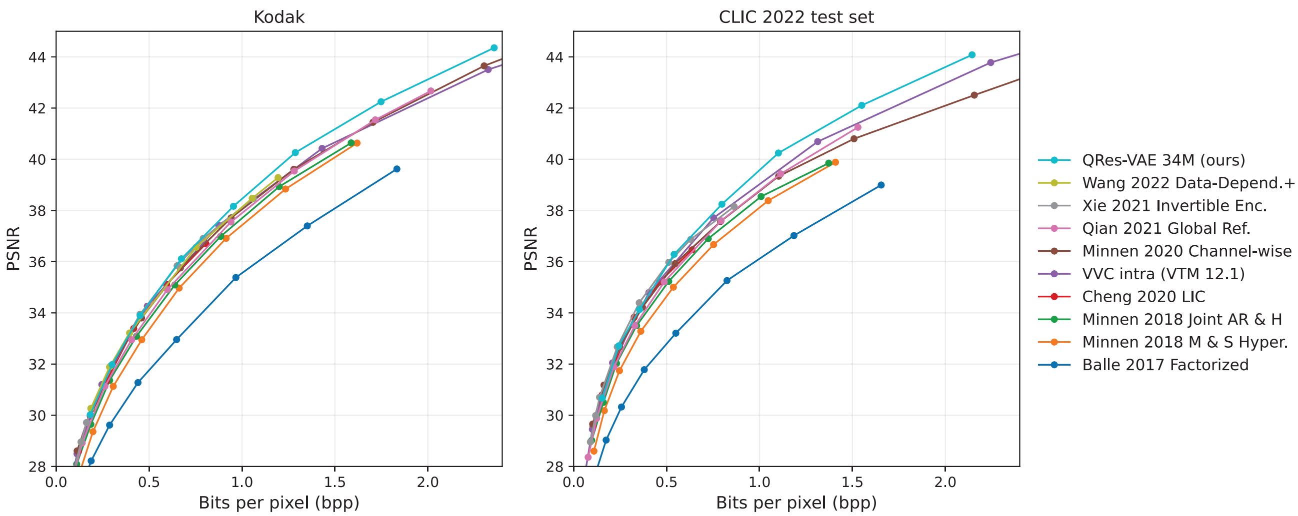


Figure 4: **Lossy compression performance on Kodak (left) and CLIC 2022 test set (right).** Our approach is on par with previous methods at low bit rates but outperforms them by a clear margin at higher bit rates. BD-rates are shown in Table 1.

	Impl.†	Parameters	Estimated end-to-end FLOPs	Latency (CPU)* Encode	Latency (CPU)* Decode	Latency (GPU)* Encode	Latency (GPU)* Decode	BD-rate ↓ Kodak	BD-rate ↓ CLIC
QRes-VAE (34M) (ours)	-	34.0M	**23.3B**	1.124s	0.558s	0.109s	0.076s	**-4.076%**	**-4.067%**
Wang 2022 Data-Depend. + [49]	Off.	53.0M	523B	-	-	-	-	-0.821%	-
Xie 2021 Invertible Enc. [52]	Off.	50.0M	68.0B	2.780s	6.343s	2.607s	5.531s	-0.475%	-2.433%
Qian 2021 Global Ref. [33]	Off.	35.8M	32.1B	341s	352s	35.3s	47.6s	6.034%	8.133%
Minnen 2020 Channel-wise [31]	TFC	116M	38.1B	0.524s	0.665s	0.136s	0.124s	0.580%	6.852%
Cheng 2020 LIC [10]	CAI	26.6M	60.7B	2.231s	5.944s	2.626s	5.564s	2.919%	3.300%
Minnen 2018 Joint AR & H [30]	CAI	25.5M	29.5B	5.327s	9.337s	2.640s	5.566s	10.61%	13.64%
Minnen 2018 M & S Hyper. [6, 30]	CAI	17.6M	28.8B	0.241s	0.442s	0.043s	0.033s	21.03%	27.06%
Balle 2017 Factorized [5]	CAI	**7.03M**	26.7B	**0.204s**	**0.411s**	**0.040s**	**0.032s**	67.81%	89.77%

*Latency is the time to code/decode the first Kodak image (W, H=768,512), including entropy coding. CPU is Intel 10700K, and GPU is Nvidia 3090.
†We use implementations from authors' official releases (Off.), TensorFlow Compression (TFC) [4], and CompressAI (CAI) [7].

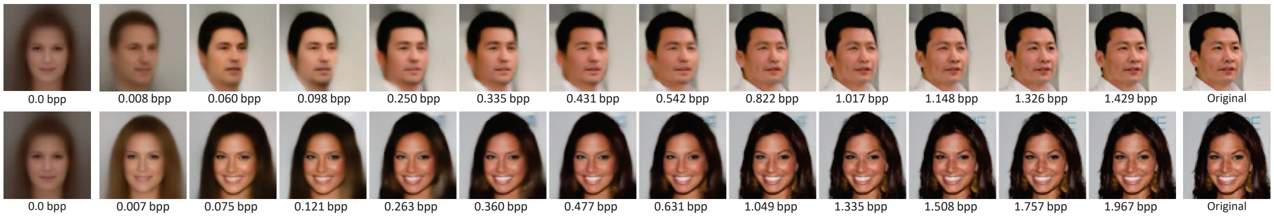
Table 1: **Computational complexity and BD-rate w.r.t. VVC intra [32] (VTM version 12.1).** FLOPs are for 256×256 input resolution. Note some values cannot be obtained using author released implementations. Our method outperforms previous methods in terms of BD-rate and at the same time maintains a relatively low computational complexity.

from a $16\times$ down-sampled feature map, which may be too "coarse" to preserve fine-grained pixel information.

In Table 1, we compare the BD-rate and computational complexity of our method against previous ones. For all methods, we use the highest bit rate model for measuring complexity, which represents the worst case scenario. Because parameter counts and FLOPs are poorly correlated with the actual encoding and decoding latency, we mainly focus on the CPU and GPU execution time for comparison.

Our method outperforms all previous ones in terms of BD-rate on both datasets. Furthermore, our model runs orders of magnitude faster than the spatial autoregressive models [30, 10, 33, 33], both on CPU and GPU. Since our approach is fully convolutional and can be easily parallelized, our encoding/decoding latency can be drastically reduced by $10\times$ when a powerful GPU is available, requiring less than 0.1 seconds to decode a 768×512 image. Note that there are indeed a few baseline methods, such as the Hyperprior [6] and Factorized model [5], execute faster than our method. However, this speed difference does not exceed an order of magnitude (*e.g.*, our 0.558s vs. 0.411s [5] for CPU decoding) and is acceptable given that our method achieves significant better compression efficiency (*e.g.*, our -4.076% vs. 67.81% [5] for the Kodak dataset in terms of BD-rate).

Another noticeable difference is that for most previous methods encoding is faster than decoding, while for our QRes-VAE (34M) this trend is reversed. Since in most image compression applications, encoding is done only once but decoding is performed many times, our ResNet VAE-based framework shows great potential for real-world deployment. For example, our model takes 1.124s/0.558s for encoding/decoding on CPU, in contrast to, *e.g.*, the Channel-wise model [31], which takes 0.524s/0.665s. This paradigm of higher encoding cost but cheaper decoding of



(a) QRes-VAE (17M), trained on CelebA, $\lambda = 1024$. Image resolution is 64×64.

(b) QRes-VAE (34M), trained on COCO patches, $\lambda = 1024$. Image resolution is 256×256.

Figure 5: **Examples of progressive decoding**. We decode only the low-dimensional latent variables from the bitstream, and we sample the remaining ones with temperature $t = 0$ (*i.e.*, simply using the prior mean). Note that the leftmost images (annotated by 0.0 bpp) can represent the "average image" in the training dataset. Better viewed by zooming in.

our model is due to the bidirectional inference structure of ResNet VAEs, in which the entire network (both the bottom-up and top-down path) is executed for encoding, but only the top-down path is executed for decoding.

4.4. Ablative Analysis

To analyze the network design, we train our smaller model, QRes-VAE (17M), with different configurations on the CelebA (64x64) training set and compare their resulting losses on the test set.

Number of latent blocks. We first study the impact of the number of latent blocks, N, by varying it from 4 to 12 and compare their testing loss. For fair comparison, we adjust the latent variable dimensions (*i.e.*, number of channels of z) to keep the same total number of latent variable elements. We also adjust the number of feature channels to keep approximately the same computational complexity. We train models with different N on CelebA (64x64) and compare their testing loss in Table 2. All models use $\lambda = 32$ (recall that λ is the scalar parameter to adjust R-D trade-off). We observe that with the same number of parameters and FLOPs, deeper model is better, but the improvement is negligible when $N \geq 10$. We conclude that $N = 12$ achieves a good balance between complexity and performance, which we choose as the default setting for our models.

Choice of residual network blocks: As we mentioned in Sec. 3.1, the choice of the residual blocks in our network is arbitrary, and different network blocks can be applied in a plug-and-play fashion. We experiment with three choices: 1) standard convolutions followed by residual connection as in Very Deep VAE [11], 2) Swin Transformer block [25], and 3) ConvNext block [27]. Again, for fair comparison, we adjust the number of feature channels to keep a similar computational complexity. We observe that at $\lambda = 32$, the three choices achieve about the same testing loss. However, when $\lambda = 4$, which corresponds to a lower bit rate, the ConvNext block clearly outperforms the alternatives. We thus use the ConvNext block as our residual block since it achieves a better overall performance.

N	4	6	8	10	12
# parameters	16.7M	16.9M	16.8M	16.9M	16.7M
FLOPs (64x64)	1.12B	1.20B	1.22B	1.24B	1.26B
Test loss ($\lambda = 32$)	0.2368	0.2186	0.2121	0.2079	**0.2078**
Δ	-	-0.0182	-0.0065	-0.0042	-0.0001

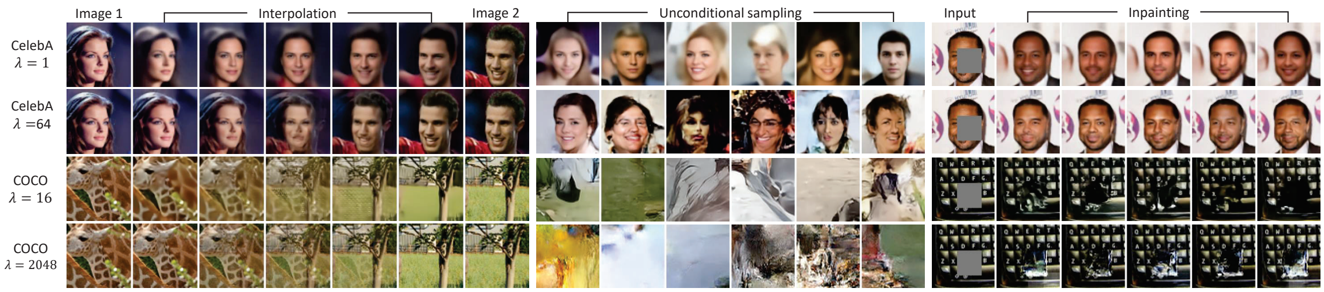
Table 2: Ablation analysis on the number of latent blocks N in QRes-VAE (17M), trained on CelebA64. Δ is the loss decrement by increasing N by 2. Deeper model is better, but the amount of improvement decreases as N increases.

Res. block choice	Convs [11]	Swin [25]	ConvNeXt [27]
# parameters	17.2M	17.0M	16.7M
FLOPs (64x64)	1.40B	1.38B	1.26B
Test loss ($\lambda = 4$)	0.0783	0.0790	**0.0757**
Test loss ($\lambda = 32$)	**0.2072**	0.2077	0.2078

Table 3: Ablation analysis on the choice of the residual block in QRes-VAE (17M), trained on CelebA64 . Overall, ConvNext [27] blocks give a better performance.

4.5. Additional Experiments

To further analyze our model, we provide a number of additional experiments in the Appendix, including bit rate distribution (Appendix 6.6), generalization to different



(a) Latent space interpolation (b) Unconditional sampling (c) Image inpainting

Figure 6: **Analysis of the latent space**. Image resolution is 64×64. A lower λ represents a model with lower bit rate. We note that our QRes-VAE learns a semantically meaningful latent space when trained on CelebA with a small λ while failed in other settings. Despite the weakness when considered as a generative model, our model takes a step towards unifying compression and generative modeling. See text for detailed discussion.

resolutions (Appendix 6.7), MS-SSIM [55] training (Appendix 6.8), and lossless compression (Appendix 6.9). We refer interested readers to these sections for more details.

4.6. Latent Representation Analysis

In this section, we show visualization results to analyze how QRes-VAE represents images in the latent space.

Progressive decoding: We show examples for progressive image decoding in Fig. 5. In Fig. 5a, we can observe that our method is able to reconstruct the gender, face orientation, and expression of the original face image given only a small subset of bitstreams (< 0.1 bpp), which correspond to low-dimensional latent variables. This suggests that the low-dimensional latent variables can encode high-level, semantic information, while the remaining high-dimensional latent variables mainly encode low-level, pixel information. We also note a similar fact for natural image patches (Fig. 5b): the low-dimensional latent variables encode the global image structure with only a small amount of bits. However, due to the complexity of the natural image distribution, semantic information is much harder to parse.

Latent space interpolation. We encode different images into latent variables (without additive uniform noise), linearly interpolate between their latent variables, and decode the latent interpolations. Results are shown in Fig. 6a. We observe that when trained on CelebA with $\lambda = 1$ (*i.e.*, the lowest bit rate), our model learns a semantically meaningful latent space, in the sense that a linear interpolation in the latent space corresponds to a semantic interpolation in the image space. However, when $\lambda = 64$, latent space interpolation becomes equivalent to pixel space interpolation, indicating that the latent representation mainly carries pixel information. A similar pixel space interpolation pattern can be observed from the results of our COCO model, suggesting that the COCO model does not learn to extract semantic information, but instead the pixel information.

Unconditional sampling. Our model can unconditionally generates images in the same way as VAEs, and the samples could give a visualization of how well our model learns the image distribution. If it learns well, the samples should look similar to real face images (for the CelebA model) or natural image patches (for the COCO model). The results are shown in Fig. 6b. We observe that the CelebA model generates blurred images when $\lambda = 1$ and artifacts when $\lambda = 64$, indicating that the model learns global structure but not the pixels arrangements. Similarly, the COCO model generates samples with some global consistency, but still being different from natural image patches.

Image inpainting. We also show results on image inpainting in Fig. 6c (the algorithm used is similar to [29] and is given in our code). The visualization of inpainting, again, reflects how well our model learns the image distribution. We obtain a similar conclusion as in previous experiments that our model succeeds in learning the image distribution when trained on the simpler, human face images, while the reconstruction quality degrades for natural image patches.

5. Conclusion

In this paper, we propose a new lossy image coder based on hierarchical VAE architectures. We show that with a quantization-aware posterior and prior model, hierarchical VAEs can achieve state-of-the-art lossy image compression performance with a relatively low computational complexity. Our work takes a step closer towards semantically meaningful compression and a unified framework for compression and generative modeling. However, like most learned lossy coders, our method requires a separately trained model for each bit rate, making it less flexible for real-world deployment. This could potentially be addressed by leveraging adaptive quantization as used in traditional image codecs, which we leave to our future work.

References

[1] Eirikur Agustsson and Lucas Theis. Universally quantized neural compression. *Advances in Neural Information Processing Systems*, 33:12367–12376, Dec. 2020.

[2] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a broken elbo. *Proceedings of the International Conference on Machine Learning*, 80:159–168, Jul. 2018.

[3] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. Hwang, and Toderici G. Nonlinear transform coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):339–353, Feb. 2021.

[4] Johannes Ballé, Sung Jin Hwang, and Eirikur Agustsson. TensorFlow Compression: Learned data compression. http://github.com/tensorflow/compression, 2022.

[5] J. Ballé, V. Laparra, and E. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. *Picture Coding Symposium*, pages 1–5, Dec. 2016.

[6] J. Ballé, D. Minnen, S. Singh, S. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. *International Conference on Learning Representations*, Apr. 2018.

[7] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, Nov. 2020.

[8] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *Video Coding Experts Group - M33*, Apr. 2001.

[9] T. Chen, H. Liu, Z. Ma, Q. Shen, X. Cao, and Y. Wang. End-to-end learnt image compression via non-local attention optimization and improved context modeling. *IEEE Transactions on Image Processing*, 30:3179–3191, Feb. 2021.

[10] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7936–7945, Jun. 2020.

[11] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *International Conference on Learning Representations*, Apr. 2021.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, May. 2021.

[13] Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, Dec. 2013.

[14] Gergely Flamich, Marton Havasi, and José Miguel Hernández-Lobato. Compressing images by encoding their latent representations with relative entropy coding. *Advances in Neural Information Processing Systems*, 33:16131–16141, Dec. 2020.

[15] Gergely Flamich, Stratis Markou, and José Miguel Hernández-Lobato. Fast relative entropy coding with a* coding. *Proceedings of the International Conference on Machine Learning*, Jul. 2022.

[16] V.K. Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, Sep. 2001.

[17] Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14766–14775, Jun. 2021.

[18] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, page 5–13, 1993.

[19] D. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, Apr. 2014.

[20] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 29, Dec. 2016.

[21] Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. *Proceedings of the International Conference on Machine Learning*, 97:3408–3417, Jun 2019.

[22] Eastman Kodak. Kodak lossless true color image suite. http://r0k.us/graphics/kodak/.

[23] J. Lainema, F. Bossen, W. Han, J. Min, and K. Ugur. Intra coding of the hevc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1792–1801, Dec. 2012.

[24] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. *Proceedings of the European Conference on Computer Vision*, pages 740–755, Sep. 2014.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *IEEE/CVF International Conference on Computer Vision*, pages 9992–10002, Oct. 2021.

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, Dec. 2015.

[27] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[28] Ming Lu, Peiyao Guo, Huiqing Shi, Chuntong Cao, and Zhan Ma. Transformer-based image compression. *Data Compression Conference*, pages 469–469, Mar. 2022.

[29] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, Jun. 2022.

[30] D. Minnen, J. Ballé, and G. Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in Neural Information Processing Systems*, 31:10794–10803, Dec. 2018.

[31] D. Minnen and S. Singh. Channel-wise autoregressive entropy models for learned image compression. *Proceedings of the IEEE International Conference on Image Processing*, pages 3339–3343, Oct. 2020.

[32] J. Pfaff, A. Filippov, S. Liu, X. Zhao, J. Chen, S. De-Luxán-Hernández, T. Wiegand, V. Rufitskiy, A. K. Ramasubramonian, and G. Van der Auwera. Intra prediction and mode coding in vvc. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3834–3847, Oct. 2021.

[33] Yichen Qian, Zhiyu Tan, Xiuyu Sun, Ming Lin, Dongyang Li, Zhenhong Sun, Li Hao, and Rong Jin. Learning accurate entropy model with global reference for image compression. *International Conference on Learning Representations*, May 2021.

[34] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in Neural Information Processing Systems*, 32, Dec. 2019.

[35] Jason Tyler Rolfe. Discrete variational autoencoders. *International Conference on Learning Representations*, Apr. 2016.

[36] W. Shi, J. Caballero, F. Huszár, J. Totz, A. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, Jun. 2016.

[37] Samarth Sinha and Adji Bousso Dieng. Consistency regularization for variational auto-encoders. *Advances in Neural Information Processing Systems*, 34, Dec. 2021.

[38] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, Sep. 2001.

[39] Casper Kaae Sø nderby, Tapani Raiko, Lars Maalø e, Søren Kaae Sø nderby, and Ole Winther. Ladder variational autoencoders. *Advances in Neural Information Processing Systems*, 29, Dec. 2016.

[40] Lucas Theis and Noureldin Y Ahmed. Algorithms for the communication of samples. *Proceedings of the International Conference on Machine Learning*, 162:21308–21328, Jul 2022.

[41] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *International Conference on Learning Representations*, Apr. 2017.

[42] James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *International Conference on Learning Representations*, May 2019.

[43] James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: lossless image compression with hierarchical latent variable models. *International Conference on Learning Representations*, Apr. 2020.

[44] Arash Vahdat, Evgeny Andriyash, and William Macready. Dvae#: Discrete variational autoencoders with relaxed boltzmann priors. *Advances in Neural Information Processing Systems*, 31, Dec. 2018.

[45] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, Dec. 2020.

[46] Arash Vahdat, William Macready, Zhengbing Bian, Amir Khoshaman, and Evgeny Andriyash. Dvae++: Discrete variational autoencoders with overlapping transformations. *International Conference on Machine Learning*, pages 5035–5044, Jul. 2018.

[47] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, Dec. 2017.

[48] G. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, Feb. 1992.

[49] Dezhao Wang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. Neural data-dependent transform for learned image compression. *arXiv preprint arXiv:2203.04963*, 2022.

[50] Matthew Willetts, Xenia Miscouridou, Stephen Roberts, and Chris Holmes. Relaxed-responsibility hierarchical discrete vaes. *Neural Information Processing Systems Workshop on Bayesian Deep Learning*, 2021.

[51] Will Williams, Sam Ringer, Tom Ash, David MacLeod, Jamie Dougherty, and John Hughes. Hierarchical quantized autoencoders. *Advances in Neural Information Processing Systems*, 33:4524–4535, Dec. 2020.

[52] Yueqi Xie, Ka Leong Cheng, and Qifeng Chen. Enhanced invertible encoding for learned image compression. *Proceedings of the ACM International Conference on Multimedia*, page 162–170, Oct. 2021.

[53] Yibo Yang, Robert Bamler, and Stephan Mandt. Variational bayesian quantization. *Proceedings of the International Conference on Machine Learning*, 119:10670–10680, Jul. 2020.

[54] Yibo Yang and Stephan Mandt. Towards empirical sandwich bounds on the rate-distortion function. *International Conference on Learning Representations*, Apr. 2022.

[55] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr. 2004.

[56] Mingtian Zhang, Andi Zhang, and Steven McDonagh. On the out-of-distribution generalization of probabilistic image modelling. *Advances in Neural Information Processing Systems*, 34:3811–3823, Dec. 2021.

[57] Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17492–17501, Jun. 2022.