6. Appendix

6.1. Detailed Architecture

We show the detailed architecture of QRes-VAE (34M) in Fig. 7, in which we assume the input resolution is 64×64 as an example. Detailed description of network blocks is provided in the caption.

Since our model is fully convolutional, the input image size is arbitrary as long as both sides are divisible by 64 pixels. This requirement is because our model downsamples an image by a ratio of 64 at maximum. In practical cases where the input resolution is not divisible by 64 pixels, we pad the image on the right and bottom borders using the edge values (also known as the *replicate padding*) to make both sides divisible by 64 pixels. When computing evaluation metrics, we crop the reconstructed image to the original resolution (*i.e.*, the one before padding).

Also note that our model has a learnable constant at the beginning of the top-down path (*i.e.*, on the right most side of Fig. 7a and Fig. 7b). The constant feature has a shape of $1 \times 1 \times C$, where C denotes the number of channels, for 64×64 images. When the input image is larger, say, 256×256, we simply replicate the constant feature accordingly, *e.g.*, $4 \times 4 \times C$ in this case.

6.2. Loss Function

Recall that the training objective of VAE is to minimize the variational upper bound on the data log-likelihood:

$$\begin{aligned} \mathcal{L} &= D_{\mathrm{KL}}(q_{Z|x} \parallel p_Z) + \mathbb{E}_{q_{Z|x}}\left[\log \frac{1}{p_{X|Z}(x|Z)}\right] \\ &= \mathbb{E}_{q_{Z|x}}\left[\log \frac{q_{Z|X}(Z|x)}{p_Z(Z)} + \log \frac{1}{p_{X|Z}(x|Z)}\right], \end{aligned} \tag{11}$$

where x is a given image in the training set. In the ResNet VAE, the posterior and prior has the form:

$$\begin{aligned} q_{Z|X}(z|x) &\triangleq q_{Z|X}(z_1, ..., z_N|x) \\ &= q_N(z_N|z_{<N}, x) \cdots q_1(z_1|x) \\ p_Z(z) &\triangleq p_Z(z_1, ..., z_N) \\ &= p_N(z_N|z_{<N}) \cdots p_1(z_2|z_1) p_1(z_1), \end{aligned} \tag{12}$$

Plug this into the VAE objective, we have

$$\mathcal{L} = \mathbb{E}_{q_{Z|x}}\left[\sum_{i=1}^{N} \log \frac{q_i(z_i|z_{<i}, x)}{p_i(z_i|z_{<i})} + \log \frac{1}{p_{X|Z}(x|Z)}\right]. \tag{13}$$

In our model, the posteriors q_i are uniform, so on the support of the PDF $q_{Z|x}$, we have $q_i(z_i|z_{<i}, x) = 1, \forall i$. Also recall our likelihood term $p_{X|Z}(x|Z) \propto e^{-\lambda \cdot d(\hat{x}, x)}$. Putting them altogether, we have

$$\mathcal{L} = \mathbb{E}_{q_{Z|x}}\left[\sum_{i=1}^{N} \log \frac{1}{p_i(z_i|z_{<i})} + \lambda \cdot d(x, \hat{x})\right] + \text{constant}. \tag{14}$$

	QRes-VAE (17M)	QRes-VAE (34M)
Training set	CelebA train	COCO 2017 train
# images	182,637	118,287
Image size	64x64	Around 640x420
Data augment.	-	Crop, h-flip
Train input size	64x64	256x256
Optimizer	Adam	Adam
Learning rate	2×10^{-4}	2×10^{-4}
LR schedule	Constant	Constant
Weight decay	0.0	0.0
Batch size	256	64
# epochs	200	400
# images seen	36.5M	47.3M
Gradient clip	5.0	2.0
EMA	0.9999	0.9999
GPUs	4 × 1080 ti	4 × RTX 6000
Time	20h	85h

Table 4. Training hyperparameters.

6.3. Training Details

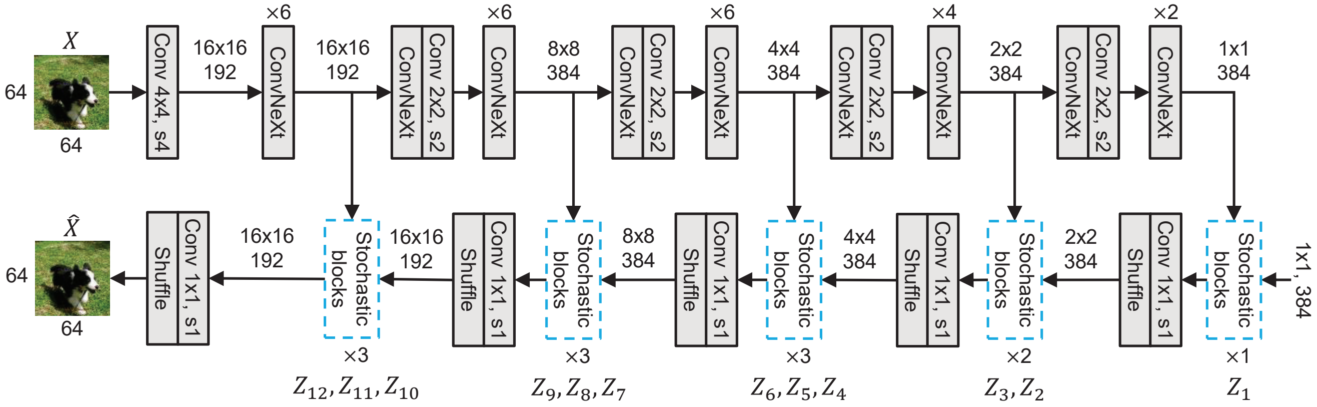
Detailed hyperparameters are listed in Table. 4, where "h-clip" denotes random horizontal flipping, and EMA stands for weights exponential moving averaging. In our experiments, we find that the learning rate and gradient clip should be set carefully to avoid gradient exploding, possibly caused by the large KL divergence (*i.e.*, high bit rates) when the prior fails to match the posterior.

6.4. Rate-Distortion Performance

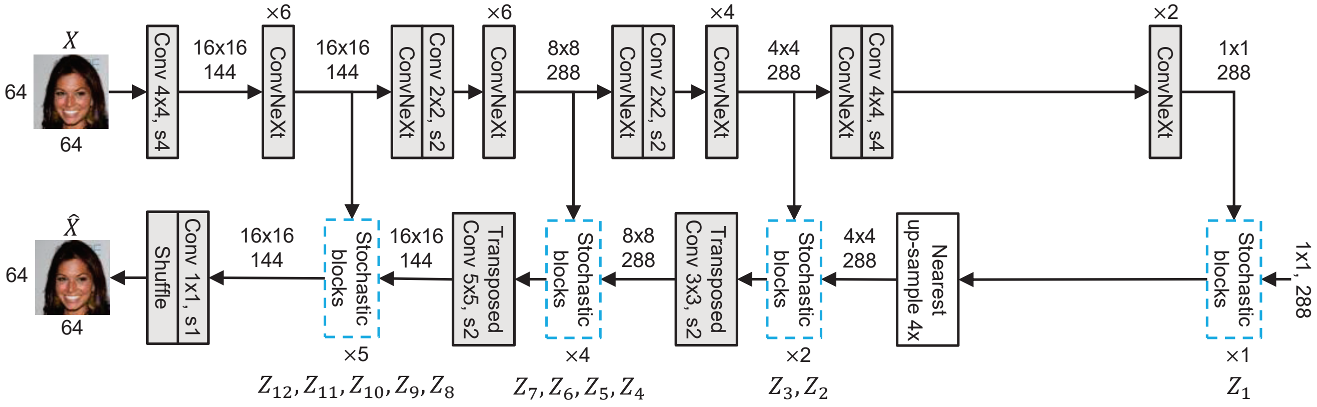
We provide the numbers of our rate-distortion curves in Table 5 and Table 6, where we show both the estimated, theoretical bit rate (computed from the KL divergence) as well as the actual bit rate (after entropy coding). We note that on Kodak images (Table 5), our actual bit rate results in an overhead when compared to the estimated bit rate. This overhead remains approximately a constant of about 0.003 bpp for all bit rates. In experiments, we find that this overhead is rooted in the entropy coding algorithm, which constantly uses extra 64 bits in each bitstream. Since our model produces 12 bitstreams for each image, for Kodak images (512×768) we obtain a constant bpp overhead of

$$\frac{12 \times 64 \text{ bits}}{512 \times 768} \approx 0.002 \text{ bits}, \tag{15}$$

which approximately matches the overhead we observed in Table 5. Note that as the image resolution increases, the



(a) QRes-VAE (34M). We train this model on COCO dataset and use it for natural image compression.



(b) QRes-VAE (17M). We train this model on CelebA dataset (human face images) and use it for ablation study. Note that in addition to *pixel shuffle*, we also use *nearest upsampling* and *transposed convolutions* for feature map upsampling. The choices of these upsampling operations are arbitrary and do not visibly impact the model performance. We adopt this combination in early development and did not further tune them.

Figure 7. **Detailed architecture of QRes-VAE** (64x64 input as an example). Numbers on the arrows denote the feature map dimension. For example, "16x16 192" means a feature map that is 16×16 in height and width, with 192 channels. *"Conv 4x4, s4"* means a convolutional layer with kernel size 4×4 and stride 4. *"ConvNeXt"* [27] is a modern residual block with depth-wise convolution, layer normalization, linear layers, and GeLU activation. *"Shuffle"* is the pixel shuffle operation [36] as often used in super resolution methods.

percentage of the extra bit rate caused by entropy coding will asymptotically decreases to zero. As we can observe in Table 6, on CLIC images (around 2048×1365), the actual bit rate is very close to the estimated bit rate, sometimes even smaller than estimated ones, for example when $\lambda = 2048$.

Note that the reported results of QRes-VAE in the main experiments are the actual bit rates, which is computed after entropy coding.

6.5. Comparison with theoretical bounds

As we introduced in the related works, VAEs can be used for computing upper bounds on the information R-D function of images. We compare our method with one such upper bound, which is computed by a ResNet VAE by Yang *et al.* [54], in Fig. 8. The blue curve in the figure is an upper bound of the (information) R-D function of Kodak images, and when viewed in the PSNR-bpp plane, it is a lower bound of the optimal achievable PSNR-bpp curve. We observe that although our approach improves upon previous method, it is still far from (a lower bound of) the theoretical limit, and further research is required to approach the limit of compression.

6.6. Bit Rate Distribution

Recall that our model produces 12 bitstreams for each image, where each bitstream corresponds to a separate latent variable. We visualize how the overall bit rate distributes over latent variables in Fig. 9. Results are averaged over the Kodak images. We also notice the posterior collapse, *i.e.*, VAEs learn to ignore latent variables, in our models. For example, Z_2 is ignored by our $\lambda = 64$ model. Posterior collapse is a commonly known problem is VAEs, and future work need to be conducted to address this issue.

λ	16	32	64	128	256	512	1024	2048
Bpp (estimated)	0.17960	0.29680	0.44780	0.66960	0.94993	1.28291	1.74430	2.35219
Bpp (entropy coding)	0.18352	0.30125	0.45200	0.67388	0.95406	1.28697	1.74814	2.35659
PSNR	30.0210	31.9801	33.8986	36.1126	38.1649	40.2613	42.2478	44.3549

Table 5. Rate-distortion performance of QRes-VAE (34M) on Kodak images.

λ	16	32	64	128	256	512	1024	2048
Bpp (estimated)	0.15370	0.24236	0.35435	0.54013	0.79785	1.10251	1.55179	2.14681
Bpp (entropy coding)	0.15405	0.24315	0.35457	0.54065	0.79773	1.10183	1.55027	2.14379
PSNR	30.6719	32.7126	34.1318	36.2879	38.2443	40.2436	42.1072	44.0814

Table 6. Rate-distortion performance of QRes-VAE (34M) on CLIC 2022 test set.

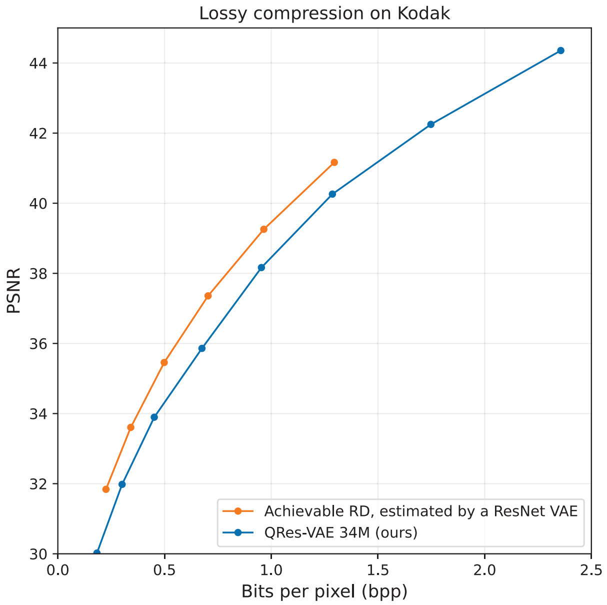


Figure 8. Comparing QRes-VAE (34M) with the an achievable PSNR-rate curve computed by ResNet VAE (blue line), taken from [54]. We can observe that there is still a large room for improvement (around 1dB at all bit rates) for lossy coders.

6.7. Generalization to Various Image Resolutions

In our main experiments on natural image compression, we can observe that our model behaves stronger on Kodak than on the CLIC 2022 test set, in the sense that our BD-rate on Kodak is more than 3% better than the Invertible Enc. model [52], while on CLIC this advantage reduces to around 1.5%. We hypothesize that this is because our model is trained on COCO dataset, which has a relatively low resolution (around 640×420 pixels) that better matches Kodak than CLIC. A better match between training/testing image resolution causes the probabilistic model optimized for the training set also match the testing set, leading to better compression efficiency.

To study this effect quantitatively, we resize the CLIC test set images, whose original resolutions are around $1,365 \times 2,048$, such that the longer sides of all images equal to r, which we choose from the following resolutions:

$$r \in \{192, 256, 384, 512, 768, 1024, 1536, 2048\}. \tag{16}$$

Then, we evaluate our model as well as two baselines, Minnen 2018 Joint AR & H [30] and Cheng 2020 LIC [10], at each resolution. Both of the two baselines are obtained from the CompressAI[2] codebase and have been trained on the same high resolution dataset.

Results are shown in Fig. 10. We observe that as the resolution r increases, the BD-rate of our model w.r.t. the baseline gets worse, from -24.6% at $r = 192$ to -16.2% at $r = 2048$. In contrast, the BD-rate between two baselines remains relatively unchanged, ranging from -8.6% to -10.0%. We thus conclude that QRes-VAE (34M) model, which is trained on COCO dataset, is stronger at lower resolutions than higher resolutions. How to design lossy coders that generalize to all resolution images is an interesting but challenging problem, which we leave to future work.

6.8. MS-SSIM as the distortion metric

Our model posit no constraints on the distortion metric $d(\cdot)$, so a different metric other than MSE could be used. For an example, we train our smaller model, QRes-VAE (17M), on the CelebA 64×64 dataset to optimize for MS-SSIM [55]. Since MS-SSIM is a similarity metric, we let

$$d(x, \hat{x}) \triangleq 1 - \text{MS-SSIM}(x, \hat{x}). \tag{17}$$

Results are shown in Fig. 11, where we also train the Joint AR & H model [30] as the baseline. We observe that our smaller model outperforms the baseline at all bit rates in terms of MS-SSIM, showing that our approach could generalize to using different distortion metrics.

6.9. Lossless Compression

Although our method is primarily designed for lossy compression, it can be easily extended to the lossless setting

---

[2] github.com/InterDigitalInc/CompressAI

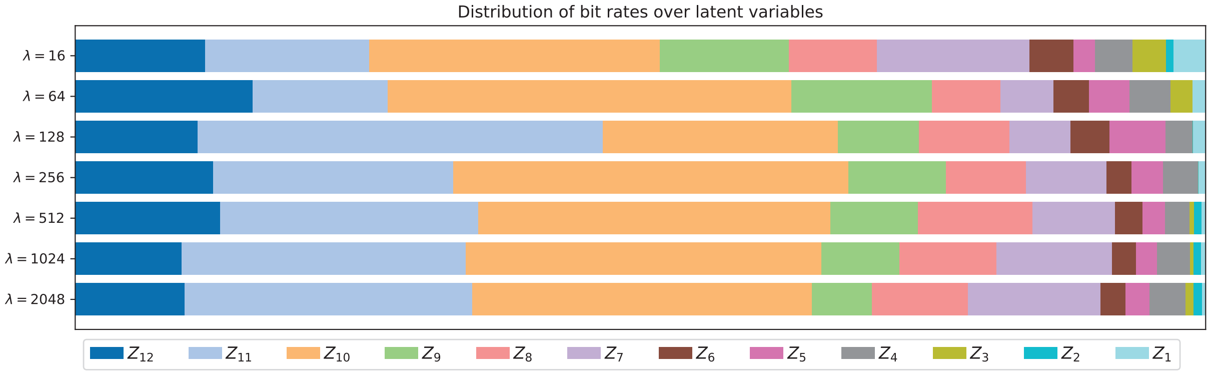


Figure 9. QRes-VAE (34M) bit rate distribution over latent variables. Rows correspond to models for different bit rates, *i.e.*, trained with different λ. $Z_1, Z_2, ..., Z_{12}$ are the latent variables, where Z_1 has the smallest (spatial) dimension (64× downsampled w.r.t. the input image) and Z_{12} has the largest dimension (4× downsampled w.r.t. the input image). See also Fig. 7a for correspondence. We observe a similar trend for all models: latent variables with higher dimension cost more bit rates.

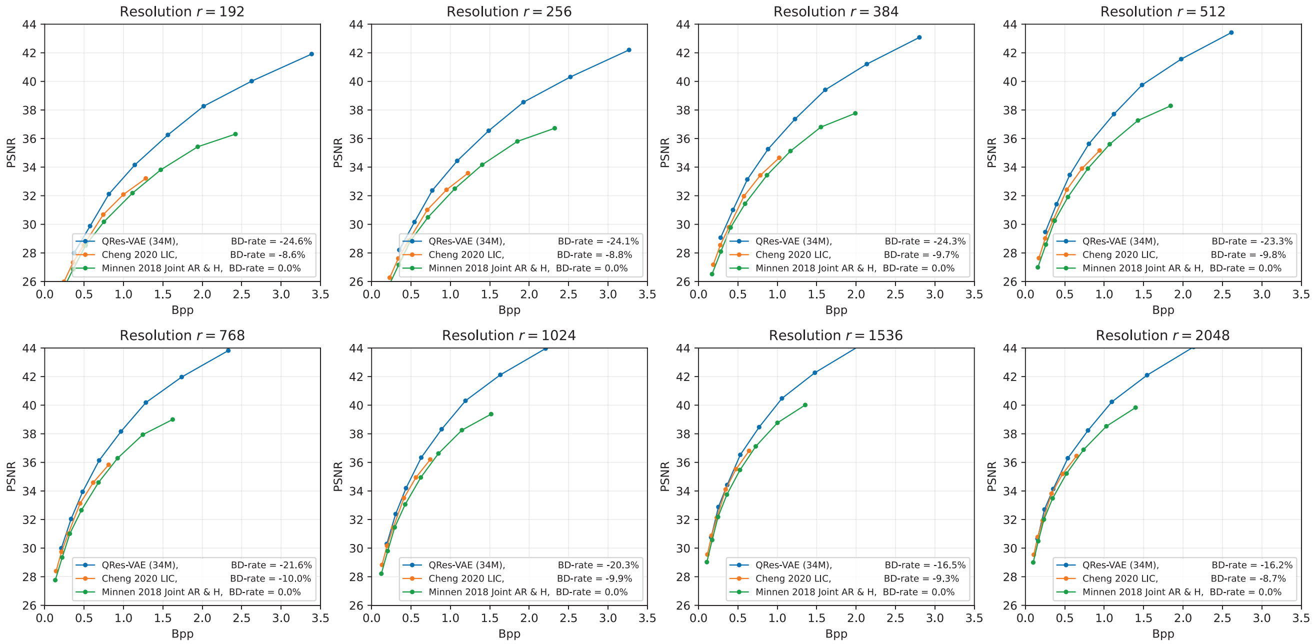


Figure 10. We resize the CLIC 2022 test set images such that their longer size equals to r (shown in each subplot), and we evaluate our approach as well as baseline methods on these downsampled images. At each resolution, BD-rate is computed w.r.t. the Joint AR & H model. We observe that our QRes-VAE (34M) is stronger at lower image resolutions.

by using a discrete data likelihood term $p_{X|Z}(\cdot)$. Specifically, we let $p_{X|Z}(\cdot)$ be a discretized Gaussian, which is the same as our prior distributions when at compression/decompression. Instead of predicting a reconstruction $\hat{x}$ as in lossy compression, we predict the mean and scale (*i.e.*, standard deviation) of the discretized Gaussian using a single convolutional layer at the end of the top-down path, which enables losslessly coding of the original image x.

We start from the QRes-VAE (34M) trained at $\lambda = 2048$, and we fine-tune for lossless compression for 200 epochs on COCO. Other training settings are the same as QRes-VAE (34M) shown in Table 4. We show the lossless compression results in Table 7. The QRes-VAE (lossless) is better than PNG but is still behind better lossless codecs such as WebP. We want to emphasize that our network architecture is not optimized for lossless compression, and this preliminary result nevertheless shows the potential possibility of a unified neural network model for both lossy and lossless image compression.

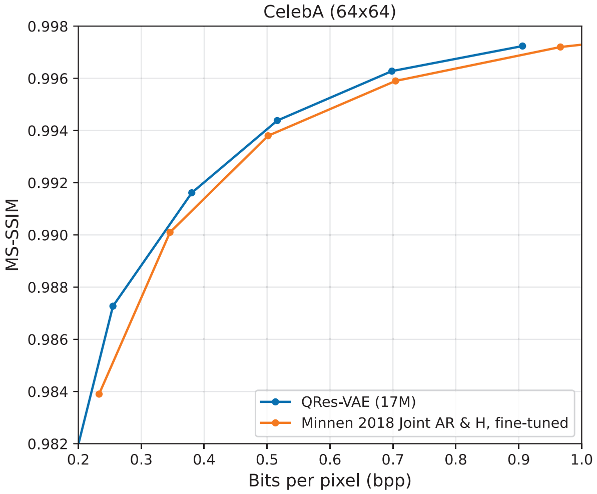


Figure 11. MS-SSIM results of QRes-VAE (17M) on CelebA. Our model is better than the baseline method.

	Kodak bpp
PNG	13.40
WebP	9.579
QRes-VAE (lossless)	10.37

Table 7. Results on lossless compression. Although not designed for lossless compression, our model achieves similar performance compared to common methods.