In this work we propose a novel lossy image compression technique based on MIRACLE [3], that is:

- **principled**: our method is based on the MDL principle, as we learn encoding and decoding distributions over a latent representation of images, which then allows us to use MIRACLE to compress a random latent sample.
- **efficient**: with our method we can compress images close to their information-theoretical limit (in the bits-back sense).
- **differentiable**: in contrast to previous work, our method does not require quantization (which is non-differentiable) for compression, hence our system can be trained end-to-end.

### Introduction

Based on earlier work on lossy image compression using VAEs by Ballé [1], we show that their architecture - when interpreted in the MIRACLE framework - corresponds to a Hierarchical VAE. We use the hierarchical structure reported in [1], but unlike them, we omit the quantization step and use diagonal Gaussians as the latent priors $p(z)$ and posteriors $q(z|x)$. We train on the CLIC 2018 dataset [4] with the $\beta$-ELBO for Gaussian likelihood as the loss:

$$L = -\mathbb{E}_{q(x)}[\log p(D|x)] + \beta RKL(p(z|D)||p(z)).$$

This is equivalent to optimizing for the PSNR as a perceptual metric.

For a single training example $x$, our encoding distribution $q(z|x)$ factorizes as $q(z_1|x)\cdots q(z_n|x)$ where

$$q(z_1|x) = N(z_1|\mu_{11}(x),\sigma_{11}^2(x)),$$

$$q(z_2|x) = N(z_2|\mu_{21}(z_1),\sigma_{21}^2(z_1)).$$

The generative model / decoding distribution $p(z|x)$ factorizes as $p(z_1)p(z_2)\cdots p(z_n)$ where

$$p(z_1|x) = N(z_1|\mu_1(x),\sigma_1^2(x),I),$$

$$p(z_2|x) = N(z_2|\mu_2(z_1),\sigma_2^2(z_1)), I).$$

The $\mu_i, \sigma_i$ are given by the layers of the network as in Figure 1. We use General Divisive Normalization [1] as the activation:

$$a_{i+1}^{(k+1)}(m,n) = u_{i+1}^{(k+1)}(m,n)$$

before the first stochastic layer. We use these as they have been shown to outperform other activations at the task of image compression / reconstruction [1].

### Coding

- Assume parties share random string $S$. (i.e. shared RNG with shared seed)
- Given the above, the following upperbound holds:

$$T[D:z] \leq H[D:z] + 2\log(D/z) + 1 + O(1)$$

where $T[D:z]$ is the communication cost [2].

The rejection sampling algorithm presented in [2] or [3] can hence be used to code images effectively.

To code image $x$:

- Pass it through the VAE so that we have $q(z|x)$ and $p(z|x)$.
- Using $p(z|x)$ as a proposal distribution, rejection sample from $q(z|x)$. The samples $\{z_i\}_{i=1}^c$ drawn from $p(z|x)$ should be driven by the shared random string $S$.
- If $z_i$ is accepted as a sample from $q(z|x)$, communicate $z_i$.

To decode:

- We simply take the $i$th sample $z_i$ from $p(z|x)$, where the sampling is driven by the same shared random string $S$.
- Pass $z_i$ through the decoder of the VAE to obtain the reconstructed image $x$.

### Results

Our architecture achieves close to state-of-the-art performance on the CLIC dataset (see Table 1). Given $c$ we can calculate the upper bound on the compressed size of an image by calculating

$$KL(q(z|x)||p(z)) + 2\log(\frac{D}{z}) + 1 + O(1)$$

where $c$ is a small constant. The bounds for Figures 1 and 3 are in Table 2.

### Challenges and Future Directions

While promising, coding the latents presents several challenges:

- Coding a single multivariate sample of the latent space is infeasible with rejection sampling, it would simply take too long (the number of latents is in the order of 10^4).
- It might be possible to code each individual latent using rejection sampling and then use arithmetic coding to compress a sequence of them.
- $A$-sampling could be adopted to this scenario to greatly speed up the rejection sampling step.

### References


