Compression without Quantization

Gergely Flamich, Marton Havasi, José Miguel Hernández-Lobato
{gf332,mh740,jmh233}@cam.ac.uk

17 June 2019

UNIVERSITY OF CAMBRIDGE

Objectives

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 Hieararchical 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 $p(\mathbf{z})$ and posteriors $q(\mathbf{z} \mid \mathbf{x})$. We train on the CLIC 2018 dataset [4] with the β-ELBO for Gaussian likelihood as the loss:

$$\mathcal{L} = \mathbb{E}_q[\log p(\mathcal{D} \mid \mathbf{z})] + \beta \mathrm{KL}\left(\, q(\mathbf{z} \mid \mathcal{D}) \mid\mid p(\mathbf{z}) \,\right). \tag{1}$$

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

For a single training example $\mathbf{x}$, our encoding distribution $q(\mathbf{z} \mid \mathbf{x})$ Factorizes as $q(\mathbf{z}_1 \mid \mathbf{x})q(\mathbf{z}_2 \mid \mathbf{z}_1)$ where

$$q(\mathbf{z}_1 \mid \mathbf{x}) = \mathcal{N}(\mathbf{z}_1 \mid \mu_1^{(e)}(\mathbf{x}), \sigma_1^{(e)}(\mathbf{x}))$$
$$q(\mathbf{z}_2 \mid \mathbf{z}_1) = \mathcal{N}(\mathbf{z}_2 \mid \mu_2^{(e)}(\mathbf{z}_1), \sigma_2^{(e)}(\mathbf{z}_1)).$$

The generative model / decoding distribution $p(\mathbf{z}, \mathbf{x})$ factorizes as $p(\mathbf{z}_1)p(\mathbf{z}_2 \mid \mathbf{z}_1)p(\mathbf{x} \mid \mathbf{z}_1)$ where

$$p(\mathbf{z}_2) = \mathcal{N}(\mathbf{z}_2 \mid \mathbf{0}, I)$$
$$p(\mathbf{z}_1 \mid \mathbf{z}_2) = \mathcal{N}(\mathbf{z}_1 \mid \mu_2^{(d)}(\mathbf{z}_2), \sigma_2^{(d)}(\mathbf{z}_2))$$
$$p(\mathbf{x} \mid \mathbf{z}_1) = \mathcal{N}(\mathbf{x} \mid \mu_1^{(d)}(\mathbf{z}_1), I).$$

The $\mu_i^{(\cdot)}(\cdot), \sigma_i^{(\cdot)}(\cdot)$ are given by the layers of the network as in Figure 1. We use General Divisive Normalization [1] as the activation

$$a_i^{(k+1)}(m,n) = \frac{u_i^{(k)}(m,n)}{\sqrt{\beta_i^{(k)} + \sum_j \gamma_j^{(k)} w_j^{(k)}(m,n)^2}} \tag{2}$$

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].

Figure 1: Original image (JPEG), 1599×777: New Court, St John's College

Figure 3: Original image (PNG), 1264×790: `thong-vo.png` from the CLIC 2018 validation set.

Coding

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

$$T[\mathcal{D} : \mathbf{z}] \leq \mathbb{I}[\mathcal{D} : \mathbf{z}] + 2\log(\mathbb{I}[\mathcal{D} : \mathbf{z}] + 1) + \mathcal{O}(1) \tag{3}$$

  where $T[\mathcal{D} : \mathbf{z}]$ 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 $\mathbf{x}$:

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

To **decode**:

1. We simply take the kth sample $\mathbf{z}$ from $p(\mathbf{z})$, where the sampling is driven by the same shared random string S.
2. Pass $\mathbf{z}$ through the decoder of the VAE to obtain the reconstructed image $\hat{\mathbf{x}}$.

Figure 2: Reconstructed image using MIRACLE. MS-SSIM=0.9751, PSNR=34.91, KL=209750 bits

Figure 4: Uncompressed using MIRACLE. MS-SSIM=0.9480, PSNR=25.53, KL=301033 bits

Results

	PSNR	MS-SSIM
CLIC Valid. Set	0.9667 ± 0.0001	32.49 ± 0.0054

Table 1: Performance of our model on the 41 validation images.

Our architecture achieves close to state-of-the-art performance on the CLIC dataset (see Table 1).

Given 3 we can calculate the upper bound on the compressed size of an image by calculating

$$\mathrm{KL}\left(\, q(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z}) \,\right) + 2\log(\mathrm{KL}\left(\, q(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z}) \,\right) + 1) + c \tag{4}$$

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

Image	Original Size	Compressed	Bits / Pixel
Figure 1	114 KB	26.2 KB	0.1688
Figure 3	1.8 MB	37.6 KB	0.3014

Table 2: Compression upper bounds on the presented images. The gain on Figure 1 is not that great, since the original image is already lossy compressed as a JPEG, although the bpp is good. The gain is much more significant on Figure 3 which is losslessly coded as a PNG, with a reasonable bpp.

Architecture

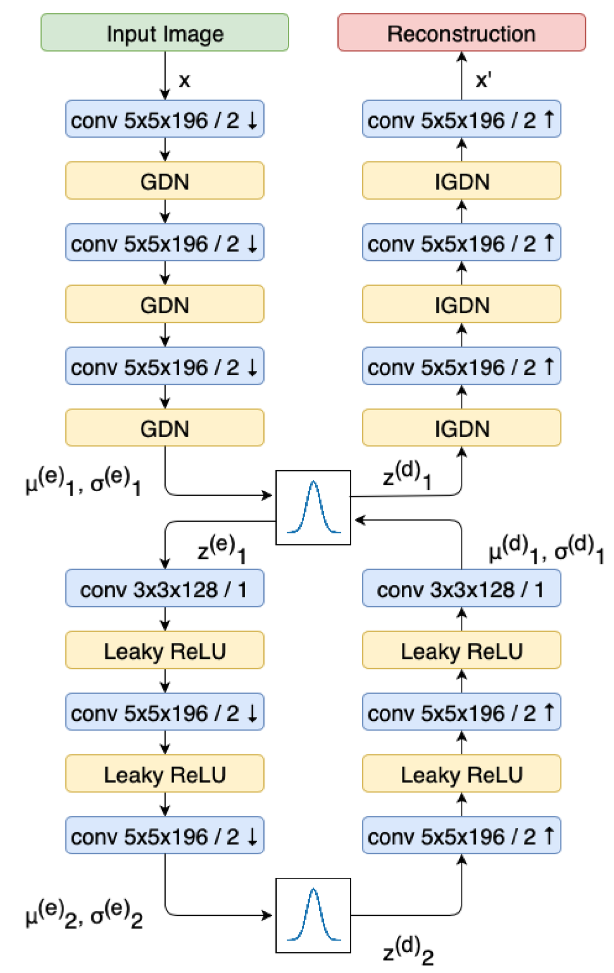

Figure 5: Our fully convolutional architecture. The $H \times W \times C/D$ convolutional blocks represent $H \times W$ sized kernels with C channels, with D times down/upsampling, indicated by the arrow.

To accommodate variable size images, we use a fully convolutional architecture, meaning we will have a variable size latent space. This is natural, as we would want a larger latent representation for larger images.

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 on the order of 10^6).
- 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

[1] Johannes Ballé et al. "Variational image compression with a scale hyperprior". In: *arXiv preprint arXiv:1802.01436* (2018).

[2] Prahladh Harsha et al. "The communication complexity of correlation". In: *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*. IEEE. 2007, pp. 10–23.

[3] Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. "Minimal Random Code Learning: Getting Bits Back from Compressed Model Parameters". In: *arXiv preprint arXiv:1810.00440* (2018).

[4] *Workshop and Challenge on Learned Image Compression*. https://www.compression.cc. Accessed: 2019-03-25.