Perceptually-Guided Lossy Image Compression

Alexander Lytchier, Jan Xu, Chris Finlay, Vira Koshkina, Ciro Cursio, Christian Besenbruch, Arsalan Zafar
Deep Render Ltd
{alex.lytchier}@deeprender.ai

Abstract

In this paper we present a lossy image compression framework that makes use of a novel loss function. The pipeline is based on a standard autoencoder architecture, augmented with a system of hierarchical hyperpriors, a powerful parallel-autoregressive context model based on PixelCNN++, and a perceptual loss based on mean opinion score (MOS) extracted from Mechanical Turk workers. Our model is trained for human satisfaction over fidelity to the original, achieving superior perceptual image quality.

1. Introduction

In many image-based tasks such as super-resolution, noise removal and image compression, there exists a trade-off between perceptual quality and distortion [7]. This is well-understood for traditional reconstruction metrics such as PSNR and MS-SSIM, which purely minimise distortion at the cost of perception: as a consequence, reconstructions are blurry or present artifacts, and thus are not visually pleasing to humans, who tend to assess reconstructions based on both similarity to the original image and aesthetic quality.

Multiple alternative losses have been developed to alleviate this problem. Among these:

- The VGG loss [13], based on intermediate feature maps of a trained VGG network [19];
- The Frechet Inception Distance (FID) [10], based on the Inception classifier [20];
- LPIPS [22], where a binary classifier is trained on human two-alternative forced choice (2AFC) between two types of distortions;
- VMAF [15], the fusion of multiple hand-crafted metrics with a support vector machine.

The LPIPS approach has been shown to be more closely correlated with real human opinion scores than other losses. Hence, we created our own dataset of distorted images, labelled with scores from a study with Amazon Mechanical Turk workers, and trained our model on this perceptual loss in conjunction with MSE and an adversarial loss.

2. Proposed Method

Our model is based on [16], with an autoencoder where the latent space is compressed using a joint hyperprior and a fast autoregressive context model based on PixelCNN++. We train this model using a combination of MSE, our own custom LPIPS and adversarial training.

2.1. Main Auto-Encoder

The architecture we use is loosely based on [16]. Our encoder network is composed of 6 convolutional layers with 384 5x5 filters each; we use GDN [3] as activation function, and Channel Normalisation [8] as our normalisation layer. The encoder downsamples twice by a factor of 2, resulting in a latent with dimensions $(12, H/4, W/4)$, where 12 is the number of channels, H is the height of the input image and W is the width.

We depart from the idea that the decoder architecture has to be symmetric with respect to the encoder. In fact, our decoder contains 6 residual blocks [9], in addition to 4 convolutional layers: we chose a more complex decoder because it helps us obtain higher quality images under adversarial training. Again, similarly to the encoder, we use GDN and ChannelNorm, and we upsample twice with bilinear upsampling operations.

2.2. Hyperprior Model

We use a hyperprior model based on [4]. As opposed to [4], our hyperprior is hierarchical, containing two stages with a hyperlatent and a hyper-hyperlatent.

The architecture of the hyperprior models is based on a 4-layer encoder and decoder. Similarly to the main autoencoder, we down/upsample twice, use GDN as the activation function, and ChannelNorm as the normalisation layer.

The probability distributions we use as priors are Laplacian. Furthermore, the prior on the highest hyperprior stage has fixed parameters learnt over the training set.

2.3. Context Model

Many of the top-performing learnt compression models include an autoregressive context model. These models are generally very powerful but extremely computationally expensive (the model executes a forward pass for every pixel in the latent space). We build on the context model used in our entry in last year's competition, PixelCNN++Lite, a modified version of PixelCNN++ [18], and apply further speed optimisation with the use of custom CUDA kernels. As in last year's entry, we only process the area equivalent to the receptive field of the network, we use less layers for each PixelCNN++ block, and use PReLU instead of the expensive Concat ELU activation function.

2.4. Quantisation Strategy

We add uniform noise to the latent spaces to simulate quantisation at training time. There are other strategies for quantisation, for example using a Straight-through Estimator (STE) [6], or universal quantisation [1]. In practice, we found that these alternative quantisation strategies do not result in significant gains over adding uniform noise.

2.5. Perceptual and Adversarial Loss

In addition to MSE, we use a custom perceptual loss based on LPIPS and an adversarial loss.

Our perceptual loss is based on the approach in [17], where the authors describe a gap in performance between the original LPIPS loss, with models that were trained on a dataset with a large variety of distortions, and a custom LPIPS loss with a model trained on compression-specific distortions. We generate a set of distortions using both traditional image codecs such as JPEG [21] and BPG [5], and learnt image compression models such as [4] and [16]; then we conducted a 2AFC task with Amazon Mechanical Turk workers to obtain binary labels to train our LPIPS backbone network.

In addition, we perform adversarial training of the decoder against a discriminator network, whose objective is to differentiate the reconstructed images against the original, uncompressed images.

We use vanilla GAN losses with the non-saturating modification for the decoder/generator, as below:

$$L_G = -\mathbb{E}_{\tilde{x} \sim p_G}[log(D(\tilde{x}))] \tag{1}$$

2.6. Training Details

We train a separate model for each of the 3 image compression tracks (0.075, 0.15 and 0.3 bpp), with different trade-offs (lambdas) between rate and distortion. Further, we train all models on randomised quantisation bin widths in a certain range: this is done to have greater control over the bpp achieved at inference time, which according to the challenge rules has to be below a specific value.

Our models are trained on random 256x256 crops of images from an in-house dataset of high-quality pictures. We use the Adam optimiser [12] with a step-down scheduler at 2, 4, 5 and 6 million iterations; we train for a total of 10 million iterations on a single Nvidia RTX Titan GPU.

2.7. Results

We beat the top 3 contestants from the CLIC 2020 competition on perceptual losses (FID and all LPIPS variants), while retaining the same bpp level. As expected from the distortion-perception tradeoff, we do not achieve the highest scores on the distortion-oriented MS-SSIM and PSNR, but we remain competitive against the past contestant. The full performance numbers are in Table 1, and we additionally show some reconstructed images in Figure 1.

2.8. Conclusion

We implemented a state-of-the-art pipeline for learnt image compression, with perceptual quality exceeding the best contestants from the CLIC 2020 competition.

Table 1. Performance numbers of our model, compared agains the top-3 contestants from last year's challenge. Data removed for peer review.

	bpp	PSNR	MS-SSIM	FID	LPIPS-AlexNet	LPIPS-VGG	LPIPS-SqueezeNet
EIC-PQE (1st place)	0.14995	30.9148	0.951199	113.910	0.135044	0.292312	0.103417
EIC-E2E-P (2nd place)	0.14998	29.9135	0.970126	117.524	0.141426	0.288893	0.107844
neuro (3rd place)	0.13762	28.0021	0.968609	135.204	0.173581	0.310259	0.138446
ours (mid-bpp)	0.14983	29.7014	0.952857	108.904	0.058786	0.246854	0.060158

Figure 1. Reconstructed 512x512 center crops of images from the test set of CLIC 2020 dataset. From left to right: neuro [2] (3rd place), EIC-E2E-P [14] (2nd place), EIC-PQE [11] (1st place), ours, original images.

References

[1] Eirikur Agustsson and Lucas Theis. Universally quantized neural compression, 2020.

[2] H. Akutsu, A. Suzuki, Z. Zhong, and K. Aizawa. Ultra low bitrate learned image compression by selective detail decoding. In *3rd Challenge on Learned Image Compression*, 2020.

[3] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. Density modeling of images using a generalized normalization transformation. *CoRR*, 2015.

[4] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[5] Fabrice Bellard. Bpg (better portable graphics)image forma. `https://bellard.org/bpg/`. Accessed: 2021-03-15.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

[7] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. *CoRR*, abs/1711.06077, 2017.

[8] Zhenwei Dai and Reinhard Heckel. Channel normalization in convolutional neural network avoids vanishing gradients, 2019.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[11] Y. Kim, S. Cho, J. Lee, S.-Y. Jeong, J. S. Choi, and J. Do. Towards the perceptual quality enhancement of low bit-rate compressed images. In *3rd Challenge on Learned Image Compression*, 2020.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[14] J. Lee, D. Kim, Y. Kim, H. Kwon, J. Kim, and T. Lee. A training method for image compression networks to improve perceptual quality of reconstructions. In *3rd Challenge on Learned Image Compression*, 2020.

[15] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. `https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652`. Accessed: 2021-03-15.

[16] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. *CoRR*, abs/1809.02736, 2018.

[17] Yash Patel, Srikar Appalaraju, and R. Manmatha. Saliency driven perceptual image compression, 2020.

[18] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[21] Gregory K. Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34(4):30–44, Apr. 1991.

[22] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.