Variational Autoencoder for Low Bit-rate Image Compression

Lei Zhou[1,2]*, Chunlei Cai[1,3]*, Yue Gao[1], Sanbao Su[1], Junmin Wu[1]
[1]Tucodec Inc, [2] University of shanghai for science and technology,[3] Shanghai Jiao Tong University
{zhoulei,caichunlei,gaoyue,susanbao,wujunmin}@tucodec.com *

Abstract

We present an end-to-end trainable image compression framework for low bit-rate image compression. Our method is based on variational autoencoder, which consists of a nonlinear encoder transformation, a uniform quantizer, a nonlinear decoder transformation and a post-processing module. The prior probability of compressed representation is modeled by a Laplacian distribution using a hyperprior autoencoder and it is trained jointly with the transformation autoencoder. In order to remove the compression artifacts and blurs for low bit-rate images, an effective convolution based post-processing module is proposed. Finally, a rate control algorithm is applied to allocate the bits adaptively for each image, considering the bits constraint of the challenge. Across the experimental results on validation and test sets, the optimized framework trained by perceptual loss generates the best performance in terms of MS-SSIM. The results also indicate that the proposed post-processing module can improve compression performance for both deep learning based and traditional methods, with the highest PSNR as 32.09 at the bit-rate of 0.15.

1. Introduction

Recently, machine learning methods have been applied for lossy image compression and promising results have been achieved using autoencoder [3, 4, 11, 12, 7, 2]. A typical neural network based image compression framework is composed of modules such as autoencoder, quantization, prior distribution model, rate estimation and rate-distortion optimization. Autoencoders are designed to transform image pixels x to data in code space y, which is composed of an encoder f_e and a decoder f_d. A vector of image intensities x $\in R^N$ is transformed to code space via encoder $y = f_e(x)$. After it, with quantization function $\hat{y} = Q(y)$, we yield a discrete-valued vector $\hat{y}$ by processing representation y. Then entropy coding methods such as arithmetic coding [8] is used to compress $\hat{y}$ lossless and the bit streams for transmission are generated. After receiving the bit streams, the quantized representation after being entropy decoding is transformed back to the image space $\hat{x}$ using a decoder $\hat{x} = f_e(\hat{y})$.

It is apparent that the prior probability model $p_{\hat{y}}(\hat{y})$ (also known as entropy model) of representation $\hat{y}$ is crucial for arithmetic coding. The actual marginal distribution of $\hat{y}$ which depends on the distributions of images is unknown. So we estimate it by the prior distribution. The prior distributions can be formulated by a parametric model and the parameters are learnt to fit the data. Given the entropy model, the lower bound of the rate is determined by the entropy of discrete prior distribution of $\hat{y}$. The actual rates achieved by a properly designed entropy code are only slightly larger than the entropy: $R = \sum_i [-log_2 p_{\hat{y}_i}(Q(f_e(x)))]$.

The role of rate-distortion optimization is to make the trade-off between the code length R and the distortion D between original image x and reconstructed image $\hat{x}$. D can be modeled using mean squared error (MSE) $D = ||x - \hat{x}||_2^2$ or the measure of perceptual distortion such as MS$-$SSIM [13]. It is obvious that if $\hat{y}$ is more centralized, then entropy R is smaller, but the representation ability of network is deteriorated and D may be increased. So a weighted sum of the rate and distortion that measures $R + \lambda D$ is optimized in an end-to-end way. We can conclude that the joint optimization of prior model $p_{\hat{y}}(\hat{y})$ and quantization is the most important technique in an effective compression system. For one thing, accurate estimation of the prior distribution of quantized representation $\hat{y}$ is beneficial for constraining the real marginal distribution of $\hat{y}$ and R. For another, an accurate prior model can make the adaptive arithmetic coding more effective in the encoding and decoding procedure.

The proposed image compression framework is based on previous methods [3, 4]. Different from these methods, an autoencoder with pyramidal encoder and more effective convolution structures is designed to improve the compression performance in our architecture. Moreover, the prior probability of the compressed representation is modeled precisely using a parameterized zero-mean Laplacian

*The first two authors share first-authorship distribution, whose parameters are learned by a hyperprior autoencoder. Considering the observation that the reconstruction will suffer from blurs and will not be appealing to human eyes if the networks are only learned by maintaining per-pixel similarity at low bit-rate. We use an effective $MS-SSIM$ based loss function to measure the perceptual loss and to train a compression codec for perceptual quality. Finally, a convolution based post-processing module is used to improve the reconstruction quality. Considering the 0.15 bpp constraints in compressing test and validation images in the challenge, a rate control algorithm is designed to select the best compression parameter for each image.

2. The Proposed Framework for Image Compression

2.1. Encoder and Decoder

Our compressive autoencoder with unbalanced structure is shown in Figure 1. The encoder f_e and decoder f_d are composed of convolutions and GDN/IGDN nonlinearities. The GDN/IGDN implement a type of local divisive normalization transformation that has been proven to be particularly suitable for density modeling and images compression [3, 4]. In the encoder, a pyramidal feature fusion structure is proposed to learn optimal, nonlinear features for each scale. The features of intermediate layers with $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ of the original size are downsampled to $\frac{1}{16}$ via convolutions. Then the downsampled features are concatenated and a 1×1 convolution is applied to generate the encoded representation y. In order to decrease the model parameters and reduce the computational costs, we have replaced all 5×5 convolutions used in [3, 4] with two 3×3 convolutions. As described in [9], replacing the 5×5 convolutions with two layers of 3×3 convolutions can reduce the parameter count by sharing the weights between adjacent tiles. We also found this replacement can improve the PSNR by 0.5 to 1 dB and reduce the computation costs.

2.2. Quantization

Assuming the transformations are powerful enough, the quantization representing values are at the center of the integer bins, that is $\hat{y}_i = round(y_i)$. The marginal density of $\hat{y}_i$ can be represented as a discrete probability mass with weights equal to the probability mass function [3]:

$$P_{\hat{y}_i}(\hat{y}_i = n) = \int_{n-0.5}^{n+0.5} P_{y_i}(t)dt \tag{1}$$

However, the gradient descent of round quantization function is ineffective because the derivatives of the round quantization function are zero almost everywhere. So in the training process, the quantizer is replaced with an additive uniform noise, that is $\tilde{y}_i = y_i + \epsilon$, where ϵ is random

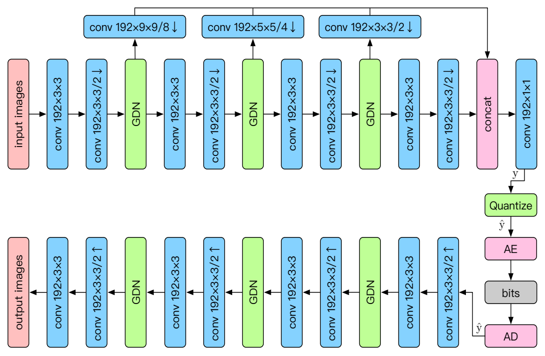

Figure 1. Illustration of the variational autoencoder architecture used in this paper. Convolution parameters are denoted as number of filter × kernel height × kernel width/ down or upsampling stride, where ↓ indicates downsampling and ↑ indicates upsampling. AE, AD represent arithmetic encoder and arithmetic decoder.

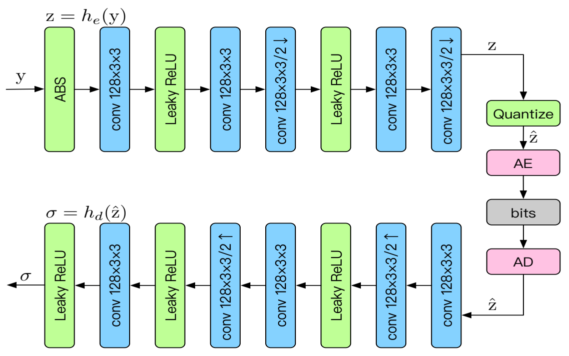

Figure 2. Illustration of the autoencoder architecture for hyperprior autoencoder.

noise. It is obvious that the entropy of $\tilde{y}$ can be used as an approximation of the entropy of $\hat{y}$. Hence we can use $\hat{y}_i = round(y_i)$ as the quantization operation in the test stage and the rates can be estimated precisely as well.

2.3. Prior Distribution and Rate Estimation

As shown in [6] that the gradients of the natural image are commonly considered as following Laplacian distribution, so we model the probability $p_{\tilde{y}}$ of each feature $\tilde{y}_i$ as a zero-centered Laplacian distribution with the standard deviation σ_i in our framework:

$$p_{\tilde{y}|\hat{z}}(\tilde{y}|\hat{z}) = \prod_i (Laplacian(0, \sigma_i^2) * \mu(-\frac{1}{2}, \frac{1}{2}))(\tilde{y}_i), \tag{2}$$

where $\sigma = h_d(\hat{z})$ and $h_d(\hat{z})$ is the decoder of hyper prior. σ_i is taken as a hyperprior to capture the spatial dependencies between elements in $\tilde{y}$ [4] and it is learned

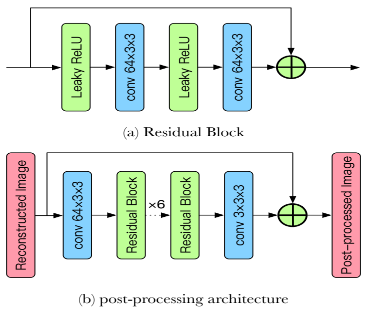

Figure 3. (a) The residual block. (b) The post-processing architecture is composed of two convolutional layers and 6 residual blocks.

by an autoencoder as well. As illustrated in Figure 2, the compressed representation $\tilde{y}$ is fed into the hyperprior encoder which summarizes the distribution of standard deviations in $z = h_e(y)$. z is then quantized $\hat{z} = Q(z)$, compressed and transmitted as side information. The decoder estimates the parameter $\sigma = h_d(\hat{z})$ and σ is used to form the Laplacian distribution for rate estimation and for compressing the quantized representation $\hat{y}$ using entropy coding. Empirically, we found that modeling the distribution using Laplacian distribution performs better than modeling it as a Gaussian distribution as Balle did [4] (0.1dB improvement at 0.15bpp). This performance gain demonstrates that Laplacian distribution is closer to the practical distribution of pixels in the natural image compared to a Gaussian distribution. As to the distribution of $\hat{z}$, because there is no prior knowledge for $\hat{z}$, we model it as a non-parametric and fully factorized density model, similar to the strategy used in [4]:

$$p_{\hat{z}|\psi}(\hat{z}|\psi) = \prod_i (P_{z_i|\psi_i}(\psi_i) * \mu(-\frac{1}{2}, \frac{1}{2}))(\hat{z}_i), \quad (3)$$

where the vector ψ_i represents the parameters of each univariate distribution $P_{z_i|\psi_i}$. Finally, the compression rates are composed of two part: rate R_y of compressed representation $\hat{y}$ and rate R_z of compressed side information $\hat{z}$. These rates are defined as follows:

$$\begin{aligned} R_y &= \sum_i -log_2(p_{\hat{y}_i|\hat{z}}(\hat{y}|\hat{z})), \\ R_z &= \sum_i -log_2(p_{\hat{z}_i|\psi}(\hat{z}|\psi)) \end{aligned} \quad (4)$$

2.4. Post-processing

A typical drawback of reconstructed images with low bit-rate is that there exists compression artifacts and smoother texture details. In order to improve the quality of reconstructed images with low bit-rate, an effective post-processing module is designed (as illustrated in Figure 3). The proposed architecture applies residual blocks as the backbone, which has been widely used in low-level applications such as super-resolution [10] and denoising [14]. It is noted that as post-processing works on full-resolution images, increasing the depth of network can increase the computational costs significantly. So we only use six residual blocks for cascaded details enhancement in our implementation. We believe the compression performance can be further improved by deeper post-processing architecture.

2.5. Optimized Rate Control

Rate-Distortion optimization is a common strategy in algorithms such as HEVC and JPEG2000. Considering the bits constraint, a rate control optimization problem is defined to allocate the bits more effectively for each image:

$$min_{j \in M} \sum_{i=1}^{N} D_j(x_i, \hat{x_i}) \;\; st. \; \sum_i R_j^i < R_{max}, \quad (5)$$

where D represents the distortion between original image x_i and the reconstructed image $\hat{x}_i$. M is the vector set which contains all possible quality configurations for the set of images. N is the image number. D_j and R_j are the distortions and rates under configuration j. The best quality configuration is selected for each image via optimizing Eq (5) in our implementation.

3. Experimental Results

1500 high-quality images licensed under creative commons were downloaded from flickr.com. These images were downsampled to 2000×2000 pixels and saved as lossless PNGs to avoid compression artefacts. From these downloaded images, we extracted 0.5 million patches with size 256×256 to train the network. We use two kinds of distortion measures: mean square error and perceptional loss to train the autoencoders with loss function as:

$$L = \lambda D + R_y + R_z, \quad (6)$$

where $D = ||x - \hat{x}||_2^2$ for MSE and $D = 0.2 \times ||x - \hat{x}||_2^2 + 0.8 \times (1 - L_{msssim})$ for perceptional loss where L_{msssim} is as defined in [13]. The results for validation set which contains 102 images are reported in Table 1. For distortion as MSE loss, eight models with $\lambda = 96$, 144, 192, 210, 230, 384, 512 and 768 are trained. Then, compressions with three settings are compared on the validation set. A single model setting (CNN+MSE) trained with $\lambda = 144$ achieves PSNR of 29.08. Rate control with eight models (CNN+MSE+RC) can improve the performance significantly and allow us to obtain a PSNR of 30.37. Setting

	PSNR	MS-SSIM	bit rate	Decoding Time
CNN+MSE (chunlei)	29.08	0.941	0.113	2776120
CNN+MSE+RC	30.37	0.952	0.15	3486646
CNN+MSE+RC+POST (tucodecTNGcnn)	30.47	0.953	0.149	20781843
CNN+MS-SSIM+RC (tucodecTNGcnn4p)	28.75	0.968	0.15	4486646
BPG	30.85	0.948	0.149	88043
H266	31.66	0.957	0.147	602012
BPG+POST	31.45	0.953	0.15	16383240
H266+POST (tucodecTNG)	32.09	0.959	0.15	16514127

Table 1. Evaluation results on CLIC 2018 validation dataset.

	PSNR	MS-SSIM	bit rate	Decoding Time
tucodecTNGcnn4p	27.67	0.964	0.15	15412641
tucodecTNGcnn	29.17	0.948	0.15	64602017
tucodecTNG	30.76	0.955	0.15	46535029

Table 2. Evaluation results on CLIC 2018 test dataset.

with post-processing (CNN+MSE+RC+POST) can achieve another 0.1 dB gain for PSNR. For distortion as perceptional loss, six models with $\lambda = 4,6,8,10,16$ and 32 are trained, the setting with rate control (CNN+MS-SSIM+RC) can generate an MS-SSIM of 0.968 on the validation set, which is the highest in the leaderboard. Empirically, we found that higher PSNR and MS-SSIM metrics can be obtained when the weighted sum of per-pixel loss (MSE) and structural loss is used rather than using only structural loss. To evaluate the role of post-processing module more comprehensively, we compared the results of two baseline image compression algorithms BPG [5] and our modified version of H266 [1], the results for multiple QPs are generated for rate control. It can be seen that post-processing can improve the PSNR and MS-SSIM metrics for both algorithms. In the test stage, we have submitted three results, tucodecTNGcnn, tucodecTNGcnn4p and tucodecTNG for compressing 286 images in the test set and the final results are reported in Table 2.

4. Conclusion

In this work, a novel autoencoder based low bit-rate image compression framework is presented. The compressed representation y is generated by a pyramidal encoder for fusing features of multiple scales and the distribution of quantized representation $\hat{y}$ is modeled using a zero-mean Laplacian distribution, which leads to better performance than when modeling as a Gaussian distribution as Balle did [4]. We observed that an end-to-end trained autoencoder has the advantage of being optimized for arbitrary metrics, such as PSNR or MS-SSIM. Training with perceptual loss enables us to achieve the best performance, in terms of MS-SSIM compared with methods listed in the leaderboard. Notably, compression performance can be improved significantly by the proposed post-processing module using simple convolutional architecture, which is proved to be effective, as can be seen by the increased PSNR by 0.2 to 0.6 dB. However high computation cost is a drawback of the convolution based framework. Existing codecs often benefit from specific hardware and optimized implementation. In future work, we would like to explore a more effective compressive autoencoder by optimizing network architecture and developing applications on hardware chips optimized for convolutional neural networks.

References

[1] H266 (https://de.wikipedia.org/wiki/h.266/), 2018. 4

[2] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017. 1

[3] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 1, 2

[4] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 1, 2, 3, 4

[5] F. Bellard. Bpg image format (http://bellard. org/bpg/), 2017. 4

[6] A. K. Jain. Fundamentals of digital image processing, 1989. 2

[7] O. Rippel and L. Bourdev. Real-time adaptive image compression. *arXiv preprint arXiv:1705.05823*, 2017. 1

[8] J. Rissanen and G. Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23, 1981. 1

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 2

[10] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2017. 3

[11] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017. 1

[12] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5435–5443. IEEE, 2017. 1

[13] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402. Ieee, 2003. 1, 3

[14] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 3